Universidad Nacional de Ingeniería

Facultad de Ingeniería Civil

Unidad de Posgrado

Modelación Matemática – Computacional del Sistema de Transporte como Herramienta para su Planificación

Tesis

Para Optar el Grado de Maestro en Ciencias con Mención en Ingeniería de Transportes

Elaborado por:
Adolfo Linares Flores

Asesor:
Dr. Santiago Esteban Contreras Aranda

Lima - Perú

2015

Dedicatoria

Para Adolfo Elías y Camila Silvia, en quienes tengo muchas esperanzas.

En homenaje a Alan Turing, quién rompió el Código Enigma de los nazis.

Agradecimientos

Al Dr. Ing. Santiago Esteban Contreras Aranda, por su excelente asesoría y su compresión por el plazo de tiempo largo que me ha dedicado en este proceso de la elaboración de la tesis, al Dr. Ing. José Carlos Matías León, nuestro guía y líder en estos años de estudio de la maestría, a la M.Sc. Rocio Espinoza Ventura, por su acertada crítica en la revisión del borrador; a los doctores y maestros de la sección de Postgrado de la “Universidad Nacional de Ingeniería” que nos han formado en el espíritu libre y crítico.

Universidad Nacional de Ingeniería
Lima, Marzo de 2015
Adolfo Linares Flores

Índice General

Resumen

La simulación del tráfico siempre ha sido un gran problema no resuelto, debido a la gran complejidad que un sistema de transporte en una ciudad significa, tanto por los diversos intereses y motivos de transportes de las personas o bienes, como de las diversos modos de transporte; incluso hay autores que creen que la modelación del sistema de transporte es una actividad esotérica, debido a la gran dificultad que esto ha acarreado, debido a que los conceptos en muchos casos son grafos cíclicos, que de alguna manera hay que romper; para salvar estas dificultades se crean **clases** del sistema de transportes para una adecuada planificación, cercana a la realidad, tanto como se pueda, en el Lenguaje de Programación C++: las clases en C++ son abstracciones del mundo real o imaginario que simulan el estado y el comportamiento de los objetos, que son particularidades de las clases, se usa el paradigma de la Programación Orientada a Objetos, y otros paradigmas del C++, que es un lenguaje multiparadigma. Se pretende construir un modelo matemático computacional que pueda simular el transporte y que sirva para su planificación. Se construyen modelos matemáticos generales (leyes) a partir de unas funciones bases y un conjunto de datos.

Palabras clave: Sistema, transporte, planificación, programa, algoritmo, clases, objetos, C++, simulación, modelo, grafos, redes.

Abstract

The traffic simulation has always been a major unsolved problem due to the complexity that a transport system in a city means, therefore the various interests and motives transport of persons or goods, as the various modes of transport ; even some authors believe that the modeling of the transport system is an esoteric activity, due to the great difficulty that this has brought, because the concepts in many cases are cyclic graphs, which somehow must be broken; To overcome these problems class transport system for proper planning, closer to reality, as far as possible, in the Programming Language C++ are created: **classes** in C++ are abstractions of real world or imaginary simulating the state and the behavior of objects, which are characteristics of the classes, the paradigm of OOP, and other paradigms of C++, which is a multi-paradigm language is used. It aims to build a computational mathematical model that can simulate the transport and that works for your schedule. General mathematical models (laws) is constructed from a base functions and a set of data.

Keywords: System, transportation, planning, program, algorithm, classes, objects, C++, simulation, model, graphs, networks.

Introducción

Un sistema de transporte es bastante complejo interactúan personas, conductores o peatones, con diversas características sicológicas y biológicas, empresarios y usuarios del transporte, gobierno y pueblo, con diversos intereses, a veces antagónicos; diversos vehículos motorizados y no motorizados, vías con diversos tratamientos y materiales, reglamentos, señales de tránsito, etc. Su modelación también es compleja, y hay autores que sostienen incluso que la modelación en transportes es algo esotérico.

Sin embargo, el avance de la informática y de las matemáticas discretas, permiten que se puedan modelar con éxito algunas partes del sistema de transportes para la mejor compresión del propio sistema y para dar solución a la amplísima variedad de los problemas del transporte.

El sistema de transporte se nos presenta como algo dado y que ha surgido en muchísimo tiempo, las vialidades en muchas ciudades son de hace cientos de años; pero a su vez los desarrollos tecnológicos recientes se tienen que incorporar y sólo nos queda observar y obtener nuestros datos sin que podamos modificar por mucho esa realidad. Es así que surge la necesidad de comprender el transporte y modelarlo; pero que modelo puede servir para el transporte, no podría ser un modelo físico, ya que sería aún más complejo que la propia realidad, queda por tanto sólo un modelo matemático computacional para poder comprender los problemas del transporte y darles una solución, que no puede ser total y para siempre, porque el transporte es algo muy vivo y en constante evolución/revolución.

La construcción de modelos es una de las tareas esenciales de la labor científica, que permite interrelacionar el conocimiento teórico con la realidad, en vista que un modelo es la representación parcial de la realidad y permite entender a las leyes científicas en sus relaciones haciendo abstracciones de partes no relevantes de la realidad en el modelo, así se puede observar mejor las características esenciales de la realidad. Se pretende crear modelos de una variedad de problemas del sistema de transportes, empleando una de las características intelectuales más poderosas de la actualidad de los lenguajes de programación orientada a objetos: las clases, que son abstracciones de objetos del mundo real (sistema de transportes de una ciudad) o imaginario (matemáticas). Esta herramienta, junto con la jerarquía de clases, herencia, polimorfismo y otros paradigmas de la programación, permite crear modelos, por tanto se crearán modelos para solucionar algunos problemas del sistema del transporte.

En la construcción de una herramienta, en este caso algunos modelos matemáticos –

computacionales para resolver algunos de los problemas del sistema de transporte, se sigue una metodología particular de la investigación científica, que se llama el modelo unificado del ciclo de vida del software, los ciclos denominados fases e iteraciones se muestran a lo largo del eje horizontal, mientras que las actividades denominadas flujos de trabajo se muestran sobre el eje vertical. Las cuatro fases son la concepción, la elaboración, la construcción y la transición (cambio al nuevo sistema). El tiempo se mueve sobre el eje horizontal de la iteración 1 a la n, y en cada ocasión se trata de una minicascada. Las cinco actividades (Requerimientos, Análisis, Diseño, Implementación y Prueba) son identificadas a aquellas en el modelo de cascada simple. Las áreas sombreadas por debajo de las curvas cerca de cada actividad tienen la intención de mostrar el esfuerzo relativo que se utiliza en una actividad en particular durante cada iteración. Por ejemplo durante la fase de concepción (iteraciones 1 y 2) la mayor parte del esfuerzo se aplica en la especificación de los requerimientos. De hecho, la especificación de los requerimientos es la única actividad que se lleva a cabo durante la iteración 1. Durante la iteración 2 una parte del esfuerzo se dedica al análisis, y un poco al diseño y a la implementación (codificación). En la medida en que los desarrolladores de software avanzan hacia la fase de elaboración, siguen trabajando sobre los requerimientos y el análisis, pero también invierten más tiempo en el diseño y la implementación, en modo especial en la fase final de la elaboración. Las pruebas se realizan desde las primeras fases de la construcción del software, las pruebas lo hacen normalmente los programadores, otros miembros del equipo de software que no codificaron el módulo que se prueba y los usuarios finales del producto. Por supuesto que en cada parte del modelo en que se pueda realizar hipótesis relevantes, se harán, para demostrar el conocimiento del candidato a maestro de la metodología tradicional de la investigación.

La investigación científica tiene tres funciones, describir, explicar y predecir; la predicción implica una traslación temporal de la explicación establecida desde un intervalo de tiempo pasado y conocido hasta otro intervalo futuro y por conocer. Para predecir, se tiene que explicar y para explicar se tiene que describir.

Los ideales teórico y operativo de una ciencia son dos: La axiomatización por una lado, que se pretende alcanzar con la matematización de los enunciados y por otro es la algoritmatización, que se alcanza con el Lenguaje de Programación C++.

Una de las características de las investigaciones científicas contemporáneas estriba en combinar la línea modelo teórica general con conceptos y métodos de las ciencias cognitivas y de programas computacionales. También se nota la proliferación de estudios de casos; esta tesis no escapa a estos atributos.

Capítulo I. PLANTEAMIENTO DEL PROBLEMA Y DISEÑO DE LA INVESTIGACIÓN

1.1. Antecedentes bibliográficos

Existe amplísima bibliografía del Lenguaje de Programación C++, que sirve de herramienta en la que pensamos, analizamos y solucionamos muchos problemas de diversos campos de la ciencia y la vida práctica, de Estructuras de Datos y Algoritmos. Por supuesto, también existen muchos libros sobre Ingeniería de Transportes y Carreteras, sobre Modelación del Transporte, sobre Econometría, etc.; la bibliografía es amplísima; existen también paquetes comerciales de software de Transportes, como el TransCAD, etc.; pero no se tiene, aunque deberá de haber, un enfoque sistémico que interrelacione estas tecnologías que muestren por un lado el desarrollo teórico, su modelación matemática – computacional, en algún lenguaje de programación, como el Lenguaje de Programación C++, el lenguaje de la ciencia, según Brian P. Flannery, Gerente de Ciencia, Estrategia y Programas de la Exxon Mobil Corporation; y muestren los resultados. Para el tratamiento de los datos también se cuenta con una extensa colección bibliográfica relacionada con la Estadística y el Cálculo de Probabilidades.

El TransCAD como muchos otros programas comerciales no muestran ni el código fuente, ni los algoritmos, ni las ayudas de cómo se los ha desarrollado, estas compañías sólo dan los manuales de usuario de cómo usar esos programas; ahora como sistemas complejos abarcan mucho y se concentran mucho en la parte gráfica y eso es una parte de su debilidad, eso que es bueno para los usuarios neófitos, pero realmente no sirve para dar información relevante; ya que el programa abarca mucho y los manuales crecen mucho, por ejemplo el manual del TransCAD es de 4 tomos, de más de 2000 páginas; ahora bien las funciones de utilidad que el usuario debe dar, también están en C++, específicamente en Visual C++; es decir, el usuario debe saber C++, para que obtenga información valiosa, sino sólo jugara con los bits de colores de los gráficos, problema que he contemplado en muchísimos ingenieros y mucho más en personas ajenas a la profesión; para poder superar estos inconvenientes es mejor realizar programas pequeños pero bien diseñados.

1.2. Descripción de la realidad problemática

El sistema de transporte es una totalidad compleja, que para su entendimiento por lo menos de una parte significativa requiere de modelos. Estos modelos pudieran ser

físicos, matemáticos, computaciones u otro tipo de modelo; sin embargo algunos modelos como el físico se descartan, por ahora, por su dificultad o imposibilidad práctica de llevarse a cabo. Los modelos matemáticos por sí solos no bastan, ya que solamente podríamos resolver problemas de juguete, tal como se muestran en casi toda la bibliografía de la Ingeniería de Transportes; la solución como en otros campos de la ciencia es una solución matemática – computacional, para poder modelar la problemática del transporte, la matemática como el lenguaje de programación sirven entonces como herramientas del pensamiento para resolver los problemas de la Ingeniería del Transporte. En la bibliografía sobre Ingeniería de Transporte hay un vacío en lo referente al tratamiento de los datos y los errores que éstos conllevan, la axiomatización a través de la matemática no es muy fuerte; y en mucha menor medida su algoritmatización, sin embargo estos problemas, generalmente de tamaño mediano a pequeño pueden resolverse no muy difícilmente, usando programas como Excel u otros programas estadísticos. Sin embargo en la planificación del transporte los esfuerzos para desarrollar modelos matemáticos – computacionales que consigan representar una realidad tan compleja ocupan una proporción considerable del esfuerzo durante los últimos años. Estas herramientas ampliamente desarrolladas y corroboradas en otros campos de la ciencia y la ingeniería, pueden considerarse en la ingeniería del transporte como herramientas en evolución.

Gran parte de los desarrollos realizados en modelos de transporte se pueden inscribirse dentro del marco del modelo de cuatro etapas: generación, distribución, reparto modal y asignación. Estas cuatro etapas son complejas cada una. En la etapa de generación de viajes el enfoque es, dados unas muestras que pueden ser de tipo de preferencias declaradas u datos directamente observados, sacar algún modelo que las pueda explicar y más aún predecir, ya que es muy onerosa la recolección de datos de una manera continua. Lamentablemente la bibliografía que se tiene a la mano no muestra o no enseña como obtuvieron, generalmente las regresiones lineales múltiples que expliquen una variable en función de otras. Otro problema encontrado en estos modelos de generación de viajes es que sólo consideran regresiones lineales entre las variables; qué pasaría si la variable que consideramos dependiente no solamente es una combinación lineal, sino de tipo polinómico o exponencial o de una función trascendental, de las variables que consideramos independientes; en este caso, nuestro modelo en el límite no valdría.

1.3. Formulación del problema

Uno de los problemas de la planificación del transporte es encontrar modelos

matemáticos que puedan explicar y predecir algunas variables que son muy difíciles o muy costosas de conseguir a partir de un conjunto de datos obtenidos de observaciones directas o de cuestionarios específicamente elaborados, por ejemplo, en la generación de viajes el problema es en términos generales el siguiente: se tiene una muestra con n variables, de ésas una variable, llamada dependiente, se tiene que explicar mediante las otras n - 1 variables, aquí también está el problema de la linealidad, generalmente los modelos que encontramos, son sólo combinaciones lineales de esas variables, limitando así a los modelos y queriendo adecuar la realidad a la teoría y no al revés. Será posible encontrar modelos matemáticos que con un alto grado de confianza puedan reflejar a la población o realidad, tal que esos modelos se hayan obtenido de muestras confiables y económicas de la misma población. Esto es lo que pretende la tesis.

1.4.1. Problema general

¿Se pueden crear clases, en el sentido del C++, que puedan modelar algunos de los problemas de la Ingeniería de Transportes para su resolución? ¿Algunas de estas clases nos pueden servir para realizar ingeniería inversa en el sentido de que aportando un conjunto de datos de alguna población y algunas funciones básicas, mostrarnos una ley general que la gobierna y poder predecir o inferir conocimientos acerca de esta población?

1.3.2. Problemas específicos

¿Usted tiene n variables independientes y una variable dependiente, designados por usted, por supuesto; puede usted sacar un modelo matemático que explique la variable dependiente en función de las otras n variables independientes o su combinación casi arbitraria de estas n variables, con una alta probabilidad de confianza, o con un nivel mínimo de error, para mejorar el planeamiento del sistema de transporte?

¿Puede aplicar el modelo anterior al estudio de casos, de problemas de transporte, por ejemplo al problema sobre la generación de viajes, o a la obtención de los parámetros de las funciones BPR volumen – tiempo de demora u otros, con una alta probabilidad de confianza, o con un nivel mínimo de error?

1.4. Justificación e importancia de la investigación

La investigación se justifica y es importante por las siguientes razones:

1. **Conveniencia.** Los modelos en la ciencia sirven para poder entender y simular

los problemas con muchos objetivos. Estos modelos que se realizan, (cada clase en C++ es un modelo conciso de alguna realidad) sirven para poder entender los problemas de la Ingeniería de Transporte, analizarlos, diseñarlos y resolverlos (programarlos). Programar es comprender, es a partir de ahí que se siguen las otras etapas, hasta su resolución. El hecho que se pueda realizar modelos que expliquen los datos obtenidos de muestras de alguna parte de la realidad, no solamente tiene un valor cognoscitivo, sino que además, puede generar economías significativas.

2. **Relevancia social.** Los resultados de la investigación deberían ser útiles a los gobiernos, empresas, universidades, instituciones privadas, ONGs, investigadores, profesionales y estudiantes que se interesen por los transportes y su modelación. Por lo que a mí respecta es muy valioso.

3. **Implicaciones prácticas**. Las clases – modelos obtenidos, tienen una aplicación práctica inmediata, se puede aplicar directamente a resolver algunos problemas del transporte, predicción de viajes, problemas de velocidad, ajustes de datos de alguna muestra, de diversa índole, generación de viajes, funciones BPR volumen – demora, etc. A alguna función o combinación de funciones sin importar los problemas de complejidad (cosas como que las funciones tienen que ser únicamente combinaciones lineales, etc. Que limitan la realidad)

4. **Valor teórico.** El que se puedan predecir algunas leyes, a partir de datos observados, tiene de hecho un indudable valor teórico, ya que ese es el objetivo de toda investigación, producir leyes que expliquen los fenómenos y los pronostiquen.

5. **Utilidad Metodológica.** Estos modelos resuelven y predicen problemas de la Ingeniería del transporte. La metodología aplicada muestra la transición de la descripción modelo-teórico al modelo computacional.

1.5. Objetivos

1.5.1. Objetivos Generales

Diseñar modelos matemáticos computacionales en C++ que resuelven algunos problemas de la Ingeniería del Transporte. Diseñar modelos matemáticos que para datos observados de la realidad puedan obtener alguna ley que las explique y que no esté

limitada por restricciones arbitrarias de los diseñadores, analistas, planificadores, etc. Del tránsito.

1.5.2. Objetivos Específicos

- Obtener un modelo matemático que explique una variable dependiente en función de una combinación lineal de variables independientes y sus casi arbitrarias combinaciones (lineales o no) o funciones de ellas, con un alto grado conocido de probabilidad de confianza.

- Profundizar en el conocimiento de los modelos de transporte de la generación de viajes.

- Resolver algunos estudios de caso de la ingeniería de transporte, utilizando el software desarrollado.

1.6. Hipótesis

1.6.1. Hipótesis general

Las clases en C++ (modelos) pueden resolver los problemas de la Ingeniería de Transportes. Se pueden obtener modelos matemáticos de algunos problemas complejos de la Ingeniería del Transporte como la predicción de viajes, o la obtención de los parámetros de las funciones BPR volumen – tiempo de demora, etc., con ayuda de las computadoras personales para mejorar el sistema de transporte a partir de modelos matemáticos computacionales y funciones bases. Se encontrará algunas leyes (modelos matemáticos) para unas muestras de datos correspondientes, es decir se hará generalizaciones, o ingeniería inversa, de que partiendo de datos y algunas funciones bases se encontrarán leyes (modelos matemáticos) que gobiernen a toda la población, es decir, expliquen y predizcan variables (que dependerá del problema concreto tratado) para escenarios futuros o cambios de las variables independientes.

1.6.2. Hipótesis particulares

- Se generan modelos matemáticos generales, para la predicción o generación de viajes, u otros problemas de la ingeniería de tránsito, con una alta probabilidad de confianza o con una mínima probabilidad de error, usando C++.

- Obtener parámetros de máxima verosimilitud que expliquen o modelen matemáticamente una variable dependiente en función de una o más variables dependientes, o sus combinaciones arbitrarias, con un grado alto y conocido de probabilidad de confianza.

1.7. Variables e indicadores

En vista que la tesis pretende encontrar un modelo matemático general que predizca o explique una variable en función de otras, es de absoluta responsabilidad del usuario de la tesis o del software definir cuál será su variable dependiente y cuál o cuáles son sus variables independientes. Note que ni damos las cualidades de las variables ni sus cantidades. La tesis no puede sustituir la inteligencia y el conocimiento del ingeniero de transportes, quién debe de dar justamente las funciones de base, que él considere relevantes para el fenómeno que desea describir, explicar y predecir. Como variables de la ingeniería de transportes podemos tener, variables demográficas, socio-económicas, precios, niveles de servicio, tasa de motorización, etc., etc., en algún problema concreto, podrían ser:

- Población: cantidad de personas de algún espacio bien definido.
- Densidad: Relación o ratio entre la población y el área geográfica que ocupa.
- Ingresos: Ingresos monetarios de las personas, probablemente encajados o por clases socio – económicas.
- Precios de los combustibles.
- Consumo de combustible por los vehículos.
- Viajes realizados en sistemas públicos de transporte.
- Tenencia de autos o coches.
- Grado de industrialización del país.
- Valor del tiempo, etc.

1.8. Unidad de Análisis

La unidad de análisis no se restringe a la generación de los viajes de la planificación del transporte, aunque es ahí donde se concentra. Ya que los modelos que se generarían soldrían ser de tipo general, pueden haber varias unidades de análisis, dependiendo del caso concreto que creamos modelar.

1.9. Tipo y Nivel de Investigación

En vista que la tesis dará soluciones a problemas prácticos concretos, es por su propósito una investigación aplicada, tecnológica y de desarrollo; y por su función es predictiva.

El nivel de la investigación, en vista de que describe algunos conceptos de la ingeniería de transporte, y explican por qué es así, es decir, implican a las investigaciones descriptivas y explicativas parciales, y responden a la pregunta: si la realidad es así por estas razones o causas, y si hago este cambio, ¿qué pasaría?; es predictiva, es decir de tercer nivel.

El tesista considera que en vista que se obtienen leyes (modelos matemáticos) que explican y pronostican escenarios futuros, el nivel de investigación es de segundo nivel.

1.10. Periodo de Análisis

Parece una pregunta fácil de responder, pero en este caso no es así; según Bjarne Stroustrup, el creador del Lenguaje de Programación C++, en su Homepage, en internet, para ser un astronauta se necesitan aproximadamente 6 meses de preparación, y para ser un programador productivo en C++, se necesitan aproximadamente 3 años, la tesis desarrolla un par de clases en C++ para generar modelos matemáticos generales de datos, que pueden usarse en ingeniería de transportes para resolver o predecir variados problemas, pero que se basa en una muy numerable clases en C++, básicamente de la Estadística y del Cálculo de Probabilidades. Entonces, cuál ha sido el periodo de análisis de la presente tesis, obviando lo anterior aproximadamente 6 meses.

1.11. Fuentes de Información e Instrumentos Utilizados

En vista que la tesis versa fundamentalmente en la construcción de una herramienta para

poder generar modelos matemáticos generales de un conjunto de datos; las fuentes de información han sido la bibliografía nombrada en la parte casi final de la tesis; de esa bibliografía consultada se han sacado colecciones de diversas estructuras de datos que me ha servido para probar la tesis; algunos datos han sido elaborados a propósito por el tesista para poder mostrar didácticamente algún modelo desarrollado. Esos datos están mostrados en la tesis y se han realizado las pruebas correspondientes del grado de confianza alcanzado o el algunos casos de contrastación de hipótesis tanto en el desarrollo del marco metodológico o del capítulo final de la contrastación de las hipótesis.

Entre los instrumentos utilizados, tenemos fundamentalmente el compilador de software C++ Builder XE5, en su versión profesional, original, que tiene como objetivo producir código de máquina, nativo ejecutable para realizar millones de operaciones por segundo, que en este caso den modelos matemáticos de los datos.

1.12. Técnicas de Recolección y Procesamiento de Datos

Hubo diversas técnicas de recolección de datos, de acuerdo a los problemas por resolver de los diversos investigadores de cuyos datos me he aprovechado. En vista que la tesis para poder generar esos modelos matemáticos necesita de una amplia gama de algoritmos de procesamiento de datos, se ha elaborado propiamente clases que simulen toda la estadística descriptiva y gran parte de la estadística inferencial; no ha sido reinventar la rueda, sino que era necesario hacerlo así, ya que por ejemplo para medir el error de los parámetros del modelo generado por la computadora, se necesitan las matrices de varianzas y covarianzas; para poder ver el grado de aproximación de la función generada con los datos, necesitamos del chi – cuadrado, y así por el estilo. Por tanto el procesamiento de datos se ha realizado con software propio, con ese fin, arriba indicado. Si se quiere optimizar algo, hay que hacer casi todo desde los cimientos, es así también si se quiere descubrir, hay que ir a la base del conocimiento y volver a mirar, desde una perspectiva diferente.

Capítulo II. MARCO TEÓRICO Y MARCO CONCEPTUAL, DESARROLLO DE LA TEORÍA.

No se desarrolla la variable, en vista de la generalidad de la tesis, que consiste en construir una herramienta que genere modelos matemáticos de diversos conjuntos de datos, las variables aparecerán en cada caso específico de esos conjuntos de datos.

2.1 Características del Conductor, del Peatón, del Vehículo y del Camino

2.1.1. Clase persona

Como una introducción al desarrollo de una clase en C++, desarrollaremos en primer lugar el concepto de persona (clase), que es lo que tienen de común un conductor y un peatón, ambos son personas; luego realizando el proceso de derivación de clases o herencia desarrollaremos las particularidades de las clases conductor y peatón. Una clase cuenta con datos miembro, que son las variables o características de la clase, es lo que muestra su estructura interna o estado de la clase; además las clases cuentan con lo que se llaman sus funciones miembro o métodos que le dan su funcionalidad o dinamismo a las clases.

Una persona tiene un nombre o nombres, apellidos, el número de identificación personal o DNI, año de nacimiento y muchísimos otras características, que sin embargo, a nuestro modelo no nos importa; a esto se llama abstracción, en C++ también existe el mecanismo de la abstracción de datos u ocultación de la información. Además una persona tiene acciones que hacer, a esas acciones en el lenguaje C++ se les llama funciones miembro.

Hay funciones miembro especiales que son los constructores, son funciones miembro que tienen el mismo nombre que su clase, no devuelven ningún valor, y establecen los datos iniciales a los objetos de las clases. Los objetos son las instanciaciones de las clases, las clases son una abstracción del mundo, no existen, lo que existen en este caso son las personas como usted o como yo, pero una generalización de estas personas concretas de carne y hueso, es el concepto de persona que en C++ se representa como una clase. Entonces un constructor construye objetos de su clase, garantizando que los valores de sus datos miembros sean válidos u correctos, una clase por tanto debe tener por lo menos un constructor; puede haber muchos constructores, de acuerdo a cada necesidad de las clases, esto se realiza por el mecanismo del polimorfismo en C++,

varias funciones pueden tener el mismo nombre, sin embargo el compilador podrá reconocer a que función se ha llamado, de acuerdo al número y tipo de los argumentos de la función, que si deben ser diferentes, por lo menos en el número de sus argumentos.

Otro tipo de funciones especiales son las funciones modificadoras del estado de un objeto, o funciones “set” (del inglés), no devuelven nada, pero cambian algún valor de los datos miembro del objeto. También existen las funciones cuya única finalidad es devolver el valor de un dato miembro (o propiedad), son las llamadas funciones “get”, devuelven un valor del tipo del dato miembro y así otras funciones como las funciones destructoras, una clase tiene una y sola una función destructora, que controla el tiempo de vida de un objeto.

En el capítulo III, se encuentran la codificación en C++ de las clases que se desarrollan en este capítulo, por ejemplo, para esta clase estará en el párrafo 4.1.1.

2.1.2. Clase conductor

La característica relevante del conductor que se adiciona a la de una persona es la del **proceso de percepción - reacción**. El proceso por medio del cual un conductor, un ciclista o un peatón evalúan y reaccionan ante un estímulo puede dividirse en cuatro subprocesos:

Percepción: el conductor ve un dispositivo de control, una señal de advertencia, o un objeto en el camino.

Identificación: el conductor identifica el objeto o el dispositivo de control y de esta manera comprende el estímulo.

Emociones: el conductor decide qué acción tomar como respuesta al estímulo; por ejemplo, pisar el pedal de freno, pasar, virar, o cambiar de carril.

Reacción o resolución: el conductor ejecuta en realidad la acción durante el subproceso de las emociones.

El tiempo que transcurre desde el inicio de la percepción hasta el final de la reacción es el tiempo total requerido para la percepción, identificación, emociones y resolución; comúnmente llamado el tiempo de percepción – reacción.

Triggs y Harris describieron detalladamente el fenómeno. Ellos observaron que el tiempo de frenado del percentil 85, obtenido para diferentes situaciones, varía de 1.26 segundos a más de 3 segundos. Las recomendaciones hechas por la Sociedad Estadounidense de Funcionarios Estatales del Transporte y las Carreteras (American Association of State Highway and Transportation Officials AASHTO) estipulan 2.5 segundos para las distancias de paro visual.

Por tanto derivamos la clase conductor de la clase persona, para adicionarle la propiedad del tiempo de percepción – reacción, teniendo en cuenta toda la teoría al respecto.

2.1.3. Clase peatón

Una característica adicional de los peatones aparte del tiempo de percepción – reacción tratada en la clase conductor, es la de la velocidad de caminata, que puede variar desde 1.97 pies/segundo hasta 3.66 pies/segundo en las personas con discapacidad. Para personas normales, las observaciones han indicado que las velocidades de caminata varían entre 3.0 hasta 8.0 pies/segundo. También se han observado diferencias entre las velocidades de caminata femenina y masculina. El Manual de Capacidad de Carreteras (The Highway Capacity Manual HCM) sugiere un uso de un valor más conservador de 4.0 pies / segundo.

Sin embargo para personas de la tercera edad, el valor de 4.0 pies/segundo puede ser muy elevado; entonces deberá de considerarse este factor en los diseños de señales cuando se espere una gran cantidad de peatones de edad avanzada. Con estas consideraciones se tiene la clase peatón que se deriva también de la clase persona.

2.1.4. Clase vehículo

Los criterios para el diseño geométrico de las carreteras se basan parcialmente en las características estáticas, cinemáticas y dinámicas de los vehículos. Las características estáticas consideran el peso y el tamaño de los vehículos; las características cinemáticas comprenden el movimiento del vehículo; y las características dinámicas toman las fuerzas que causan el movimiento de los vehículos.

CARACTERISTÍCAS ESTÁTICAS

Las carreteras alojan una diversidad de tipos de vehículos, desde automóviles particulares como tránsito de camiones, es esencial que los criterios de diseño

consideren las características de los distintos tipos de vehículos. Un conocimiento completo de estas características va a ayudar a los ingenieros de carreteras y de tránsito para el diseño de carreteras y de sistemas de control de tránsito, que permita una operación segura y sin contratiempos del vehículo en movimiento, especialmente durante las maniobras básicas de paso, alto total y dar vuelta. Por tanto el diseño de una carretera incluye la selección del ***vehículo de diseño***, cuyas características cubran las relacionadas con la mayor parte de los vehículos que se espera usen la carretera.

Anteriormente los estados prescribían límites para el peso y los tamaños de los vehículos, ahora estas restricciones ya nos la pueden dar los estados, lo que implica que camiones más pesados y más largos y anchos atravesarán las carreteras.

El tamaño del vehículo de diseño para una carretera es un factor importante en la determinación de los estándares de diseño de varios componentes físicos de la carretera. Estos incluyen el ancho del carril, ancho de cuneta, longitud y ancho de las bahías de estacionamiento, y la longitud de las curvas verticales. El peso en los ejes de los vehículos (en espera) sobre la carretera es importante para determinar el peralte del pavimento y la pendiente máxima.

Por tanto, es necesario que se clasifiquen todos los vehículos, de modo que puedan proporcionarse características estáticas representativas dentro de una clase específica para propósito de diseño. La AASHTO ha seleccionado tres categorías generales de vehículos: automóviles de pasajeros, camiones y autobuses/vehículos recreativos. Se ha seleccionado un total de 15 vehículos de diseño para representar en las tres categorías los distintos tipos de vehículos.

Vehicle Type	Symbol	Dimensions in Feet: Wheel Base	Overhang: Front	Overhang: Rear	Overall: Length	Overall: Width	Height	Minimum Turning Radius Outside Front Wheel
Passenger	P	11	3	5	19	7.0	--	24
Single-unit truck	SU	20	4	6	30	8.5	13.5	42
Single-unit bus	BUS	25	7	8	40	8.5	13.5	42
Semitrailer combination								
Intermediate	WB40	13 + 27 = 40*	4	6	50	8.5	13.5	40
Large	WB50	20 + 30 = 50*	3	2	55	8.5	13.5	45
Full trailer combination	WB60	9.7 + 20 + 9.4** + 20.9 = 60	2	3	65	8.5	13.5	45

Figura 1. Dimensiones de algunos vehículos de diseño

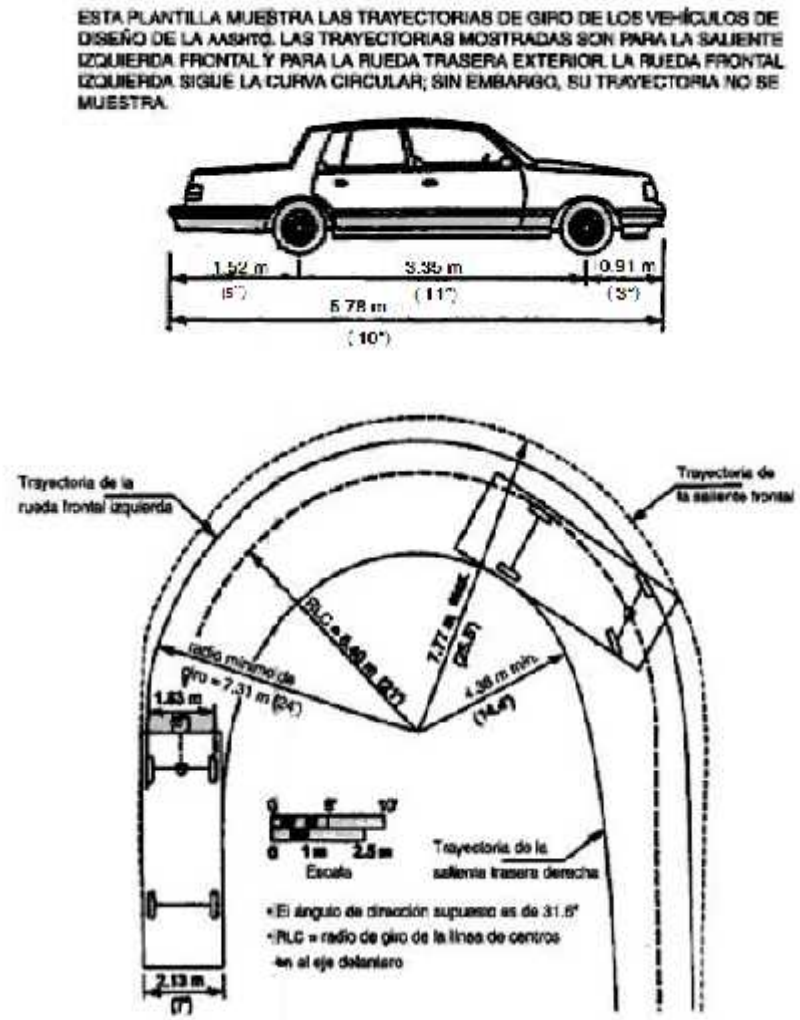


Figura 3.2 Trayectoria mínima de giro para el vehículo de pasajeros de diseño.

FUENTE: Departamento de Carreteras y de Transporte Público del estado de Texas, reimpreso en *A Policy on Geometric Design of Highways and Streets* (Una política sobre el diseño geométrico de autopistas y calles), Sociedad Estadounidense de Funcionarios Estatales de Carreteras y de Transporte (American Association of State Highway and Transportation Officials), Washington, D.C., 2001. Utilizado con autorización.

Figura 2. Radio de giro mínimo, para una velocidad de 10 millas/hora [7]

CARACTERISTÍCAS CINEMÁTICAS

El elemento principal de las características cinemáticas es la capacidad de aceleración del vehículo. Por tanto, un estudio de las características cinemáticas del vehículo, incluye principalmente un estudio de cómo influye la tasa de aceleración a los elementos del movimiento, tales como la velocidad y distancia. En esta sección se revisa la relación matemática entre aceleración, velocidad, distancia y tiempo.

Se considera que un vehículo se mueve a una distancia x a lo largo de una línea recta, desde el punto o hasta el punto m en un plano de referencia T. El vector de posición del vehículo después del tiempo t, puede expresarse como

$$\mathbf{r}_0 = x * \mathbf{i} \qquad (2.1.1)$$

Donde $\mathbf{r}_0$ es el vector de posición para m en T.

$\mathbf{i}$ vector unitario paralelo a la línea om.

x distancia a lo largo de la línea recta.

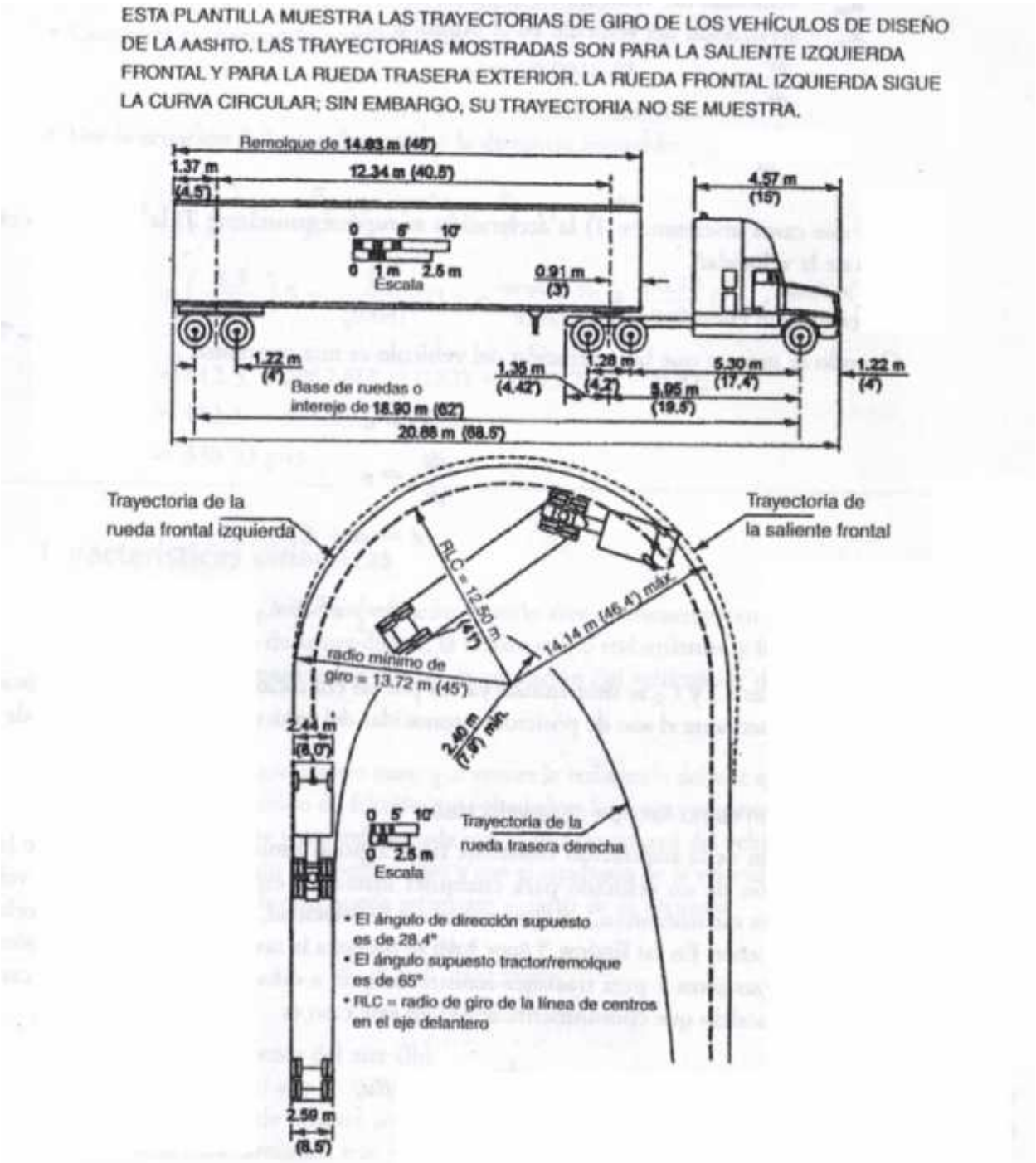


Figura 3.3 Trayectoria de giro mínima para el vehículo de diseño WB-62.

FUENTE: Departamento de Carreteras y de Transporte Público del estado de Texas, reimpreso en *A Policy on Geometric Design of Highways and Streets* (Una política sobre el diseño geométrico de autopistas y calles), Sociedad Estadounidense de Funcionarios Estatales de Carreteras y de Transporte (American Association of State Highway and Transportation Officials), Washington, D. C. Utilizado con autorización.

Figura 3. Radio de giro mínimo para una velocidad de 10 millas/hora [7]

La velocidad y la aceleración para m pueden expresarse simplemente como

$$\mathbf{u}_m = \dot{\mathbf{r}}_O = \dot{x}_t \qquad (2.1.2)$$

$$\mathbf{a}_m = \ddot{\mathbf{r}}_O = \ddot{x}_t \qquad (2.1.3)$$

Donde $\mathbf{u}_m$= velocidad del vehículo en el punto m
$\mathbf{a}_m$ = aceleración del vehículo en el punto m.

$$\dot{x} = \frac{d}{d}$$

$$\ddot{x} = \frac{d^2x}{d^{\,2}}$$

Hay dos casos interesantes: 1) la aceleración se supone constante, 2) la aceleración es una función de la velocidad.
La aceleración del vehículo se supone constante,

$$\ddot{x}_t = a$$

$$\frac{d\dot{x}}{d} = a \qquad (2.1.4)$$

$$\dot{x} = a * t + C_1 \qquad (2.1.5)$$

$$x = \frac{1}{2} * a * t^2 + C_1 * t + C_2 \qquad (2.1.6)$$

Las constantes C_1 y C_2 se determinan ya sea por las condiciones iniciales de la velocidad y de la posición o mediante el uso de posiciones conocidas del vehículo para los instantes de tiempo diferentes.
La aceleración es una función de la velocidad, la suposición de la aceleración constante tiene algunas limitaciones, debido a que la capacidad de aceleración de un vehículo para cualquier instante t está relacionada con la velocidad del vehículo para ese instante ($\mathbf{u}_t$). Entre menor sea la velocidad, es mayor la tasa de aceleración que se puede obtener. La Figura 5, ha servido para realizar la interpolación lineal, en la clase vehículo, entre las tasas máximas de aceleración versus la relación peso/potencia (lb/H.P.) para diferentes rangos de velocidades en millas por hora de dos tipos de vehículos.

Un modelo que comúnmente se usa en este caso es

$$\frac{du_t}{d} = \alpha - \beta * u_t \qquad (2.1.7)$$

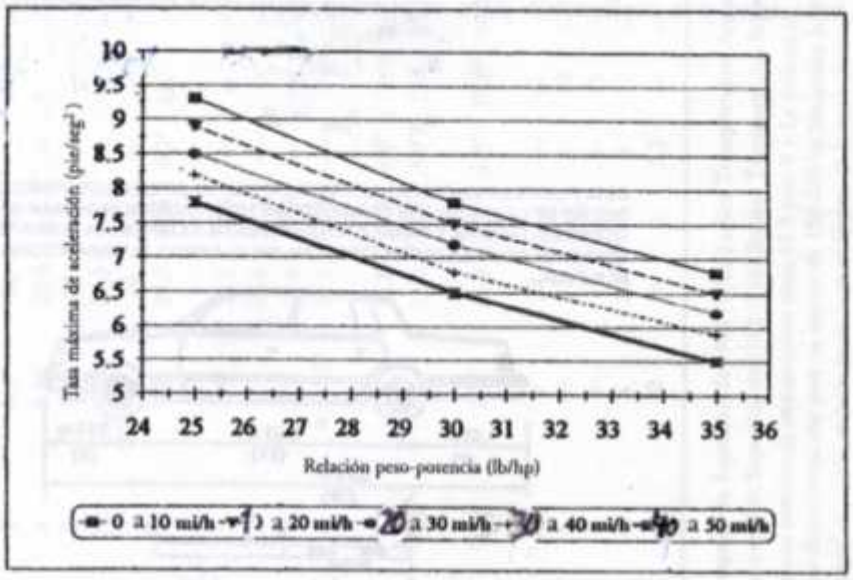
Donde α y β son constantes.
En este modelo, la tasa máxima de aceleración que teóricamente puede alcanzarse es α, lo que significa que tiene unidades de aceleración. β tiene dimensiones del recíproco

del tiempo.

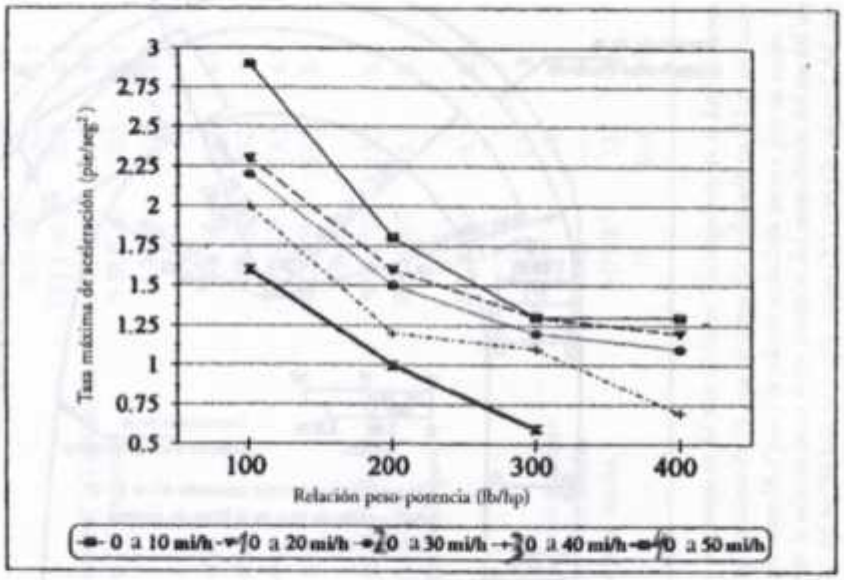
La integración de (2.1.7) da la velocidad en función del tiempo

$$u_t = \frac{a}{\beta} + \left(\frac{-a}{\beta} + u_0\right) * e^{-\beta(t-t_0)} \qquad (2.1.8)$$

Donde t_0 es el tiempo inicial, u_0 es la velocidad inicial.



(a) automóviles



(b) tractor-semirremolque

Figura 3.4 Capacidad de aceleración de automóviles de pasajeros y de tractores-semirremolques en un cambio horizontal.

FUENTE: Modificado de *Traffic Engineering Handbook* (Manual de ingeniería de tránsito), 5a. ed., Instituto de Ingenieros del Transporte (Institute of Transportation Engineers), Washington, D.C., 1999.

Figura 4. Tasas máximas de aceleración para dos tipos de vehículos. [7]

La integración de la ecuación (2.1.8) da la distancia recorrida en función del tiempo:

$$x = x_0 + \frac{a}{\beta} * (t - t_0) - \frac{a}{\beta^2} * \left(1 - e^{-\beta(t-t_0)}\right) + \frac{u_0}{\beta} * \left(1 - e^{-\beta(t-t_0)}\right) \qquad (2.1.9)$$

Donde x_0 es la posición inicial del vehículo.

CARACTERISTÍCAS DINÁMICAS

Varias fuerzas actúan sobre un vehículo cuando éste se encuentra en movimiento: la resistencia del aire, la resistencia de la pendiente, la resistencia al rodamiento, y la resistencia a la curva. El grado hasta el cual estas fuerzas afectan la operación del vehículo se discute a continuación:

Resistencia del aire: Un vehículo en movimiento tiene que vencer la resistencia del aire que tiene enfrente, así como la fuerza debida a la fricción de su alrededor. La fuerza requerida para vencerlas se conoce como resistencia del aire y está relacionada con el área transversal del vehículo, en una dirección perpendicular a la dirección del movimiento y con el cuadrado de la velocidad del vehículo. Claffey ha demostrado que esta fuerza puede estimarse a partir de la fórmula

$$R_a = 0.5 * \frac{(2.1 * p * C_D * A * u^2)}{g} \qquad (2.1.10)$$

Donde

R_a = fuerza resistente del aire (lb)

p = densidad del aire (0.0766 lb/pie³) al nivel del mar; menor para elevaciones mayores.

A = área transversal frontal (pie²)

u = velocidad del vehículo en (millas/hora)

g = aceleración de la gravedad (32.2 pies/seg²)

Resistencia a la pendiente: Cuando un vehículo sube por una pendiente, hay un componente del peso que actúa hacia abajo, a lo largo del plano de la carretera. Esto crea una fuerza que actúa en dirección opuesta al movimiento. Esta fuerza es la resistencia a la pendiente. Por tanto, un vehículo que sube por una pendiente tiende a perder velocidad a menos que se aplique una fuerza de aceleración.

Rp = peso * pendiente de la vía, la pendiente en decimales (2.1.11)

Resistencia al rodamiento: Existen fuerzas dentro del vehículo mismo que ofrecen resistencia al movimiento. Estas fuerzas son debidas principalmente al efecto de fricción en las partes móviles del vehículo, pero también incluyen el deslizamiento por fricción entre la superficie del pavimento y las llantas. El efecto acumulado de estas fuerzas sobre el movimiento se conoce como resistencia al rodamiento. La resistencia al rodamiento depende de la velocidad del vehículo y del tipo de pavimento. Las fuerzas de rodamiento son relativamente menores en pavimentos lisos que en pavimentos rugosos. La fuerza de resistencia al rodamiento para los automóviles de pasajeros sobre un pavimento liso puede determinarse a partir de la relación

$$R_t = (C_r + 2.15 * C_r * u^2) * W \quad (2.1.12)$$

Donde

R_t = fuerza de resistencia al rodamiento (lb)
C_r = constante (comúnmente 0.012 para automóviles)
C_r = constante (comúnmente 0.65*10^-6 seg²/pie² para automóviles)
u = velocidad del vehículo (millas / hora)
W = peso bruto del vehículo (lb)

Para camiones, la resistencia al rodamiento puede obtenerse a partir de

$$R_r = (C_a + 1.47 * C_b * u) * W \quad (2.1.13)$$

Donde

R_r = fuerza de resistencia al rodamiento (lb)
C_a = constante (comúnmente 0.02445 para camiones)
C_b = constante (comúnmente 0.00044 seg/pie para camiones)
u = velocidad del vehículo (millas / hora)
W = peso bruto del vehículo (lb)

Resistencia de la curva: Cuando se maniobra un automóvil para tomar una curva, hay fuerzas externas que actúan sobre las llantas delanteras del vehículo. Estas fuerzas tienen componentes que ejercen un efecto retardatario sobre el movimiento del vehículo hacía adelante. El efecto acumulado de estos componentes constituye la resistencia de la curva. Esta resistencia depende del radio de la curva, del peso bruto del vehículo y de la velocidad a la cual se mueve el vehículo. Puede determinarse como

$$R_c = 0.5 * \left(\frac{2.1 * u^2 * W}{g * R}\right) \quad (2.1.14)$$

Donde

R_c = fuerza de resistencia a la curva (lb)

g = aceleración de la gravedad (32.2 pies/seg²)

R = radio de la curva (pies)

u = velocidad del vehículo (millas / hora)

W = peso bruto del vehículo (lb)

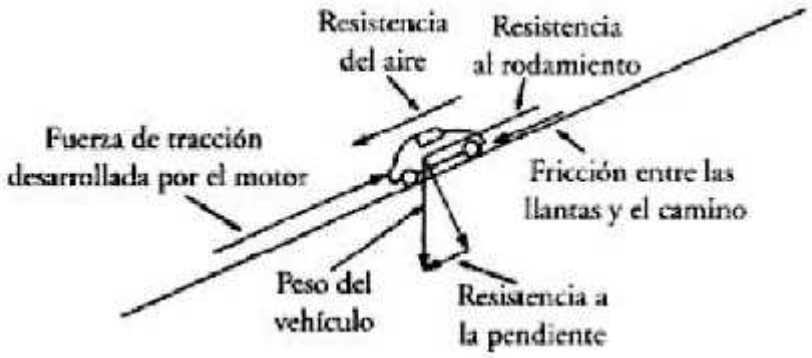


Figura 5. Fuerzas que actúan sobre un vehículo en movimiento. [7]

Requerimientos de potencia: La potencia es la tasa a la cual se realiza un trabajo. Generalmente se expresa en caballos de potencia, donde 1 H.P. = 550 pies-libra/seg. La capacidad de rendimiento de un vehículo se mide en términos de los caballos de potencia que el motor puede producir para vencer las fuerzas de resistencia del aire, de la pendiente, de la curva, y de fricción para poner el vehículo en movimiento. La potencia desarrollada por el motor es

$$P = \frac{1.4 * R * u}{5} \quad (2.1.15)$$

Donde

P = potencia generada (H.P.)

R = suma de la resistencia al movimiento (lb)

u = velocidad del vehículo (millas/hora)

Frenado

Considere un vehículo que viaja cuesta abajo con una velocidad inicial de u, en millas/hora, como se muestra en la Figura 7. Sea

W = peso del vehículo
f = coeficiente de fricción entre las llantas y el pavimento del camino
y = ángulo entre la pendiente y la horizontal
a = desaceleración del vehículo cuando se aplican los frenos
D_b = componente horizontal de la distancia recorrida durante el frenado (es decir, desde el momento en que se aplican los frenos hasta el momento en que el vehículo queda en reposo)

Observe que la distancia denominada como distancia de frenado es la distancia horizontal y no la distancia inclinada x. La razón es que la medición de distancias en topografía es horizontal y, por tanto, la medición de distancias en el diseño de carreteras se hace con base en un plano horizontal.

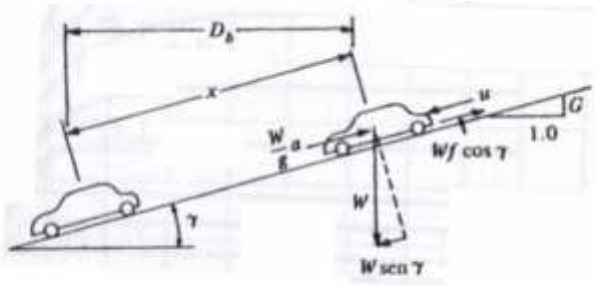
Fuerza de fricción sobre el vehículo = W*f*cos y

La fuerza que actúa sobre el vehículo debido a la desaceleración es W*a/g, donde g es la aceleración de la gravedad. La componente del peso del vehículo es Wsen y. Al sustituir en $\sum f = m * a$, se obtiene

$$W * s \quad - W * f * c \quad = \frac{W*a}{g} \qquad (2.1.16)$$

La desaceleración que lleva al vehículo a una posición estacionaria puede encontrarse en términos de la velocidad inicial u como a = -u²/(2*x) (suponiendo una desaceleración uniforme). Entonces la ecuación (2.1.16) se puede escribir como

$$W * s \quad - W * f * c \quad = -\frac{W*u^2}{2*g*x} \qquad (2.1.17)$$



W = peso del vehículo
f = coeficiente de fricción
g = aceleración de la gravedad
a = aceleración del vehículo
u = velocidad cuando se aplican los frenos
D_b = distancia de frenado
γ = ángulo de inclinación
G = tan γ (% en pendiente/100)
x = distancia recorrida por el vehículo a lo largo del camino durante el frenado

Nota: El valor absoluto de G que se usa como la pendiente descendente está considerada en el desarrollo de la ecuación.

Figura 3.7 Fuerzas que actúan sobre un vehículo que frena en una pendiente descendente.

Figura 6. Fuerzas que actúan en un vehículo que frena. [7]

Sin embargo $D_b = x * c$, por tanto se obtiene

$$D_b = \frac{u^2}{2*g*(f-t)} \qquad (2.1.18)$$

Sin embargo, observe que tany = -G, considerando la pendiente G, en tanto por uno, cuesta arriba como positiva, y negativa cuando el vehículo está yendo para abajo. Entonces la ecuación (2.1.18) se puede escribirse así:

$$D_b = \frac{u^2}{2*g*(f+G)} \qquad (2.1.19)$$

La AASHTO representa el coeficiente de fricción f como a/g, e hizo la observación de que para asegurar que el pavimento tenga y mantenga estos coeficientes debe diseñarse para que se cumpla con los criterios establecidos en los Guidelines for Skid Resistance Pavement Design de la AASHTO (Lineamientos para el diseño de pavimentos a prueba de derrape). La AASHTO también recomienda que se use una tasa de desaceleración de 11.2 pies/seg², ya que es una tasa de desaceleración cómoda para la mayoría de los conductores. Entonces la ecuación (2.1.19) se transforma en

$$D_b = \frac{u^2}{2*g*(\frac{a}{g}+G)} \qquad (2.1.20)$$

En forma similar, puede demostrarse que la distancia horizontal recorrida para reducir la velocidad de un vehículo de u_1 a u_2 en millas/hora durante una maniobra de frenado está dada por

$$D_b = \frac{u_1^2 - u_2^2}{2 * g * (\frac{u}{g} + G)} \qquad (2.1.20)$$

La distancia recorrida por un vehículo entre el momento en el cual el conductor observa un objeto en la trayectoria del vehículo y en el momento en el que el vehículo realmente se detiene es mayor que la distancia de frenado, ya que incluye la distancia recorrida durante el tiempo de percepción – reacción, que se denomina como la distancia visual S de paro y está dada por

$$S(f\ ) = 1.47 * u * t + \frac{u^2}{2 * g * (\frac{u}{g} + G)} \qquad (2.1.21)$$

Donde el primer término de la ecuación (2.1.21) es la distancia recorrida en pies durante el tiempo de percepción – reacción t (segundos).

Radio mínimo de una curva circular

Cuando un vehículo se mueve siguiendo una curva circular, la inercia hace que el vehículo continúe por la tangente de la curva, con objeto de que el vehículo no siga por la tangente, el camino en la curva tiene un peralte hacia el centro. La inclinación del camino hacia el centro de la curva se conoce como sobreelevación. La aceleración centrípeta depende de la componente del peso del vehículo a lo largo de la superficie inclinada del camino y de la fricción lateral entre las llantas y la superficie del camino. En la Figura 8 se muestra la acción de estas fuerzas sobre un vehículo que se mueve siguiendo una curva circular.

El radio mínimo R de una curva circular para un vehículo que viaja a u millas/hora puede determinarse considerando el equilibrio del vehículo respecto a su movimiento ascendente o descendente por el declive. Si α es el ángulo de inclinación de la carretera, la componente del peso descendiendo por el declive es W*senα, y la fuerza de fricción que también actúa descendiendo por el declive es W*f*cosα. La fuerza centrípeta Fc es:

$$F\ = \frac{W * a_c}{g} \qquad (2.1.22)$$

Donde

a_c = aceleración para un movimiento curvilíneo = u²/R (R = radio de giro de la curva)

W = peso del vehículo

g = aceleración de la gravedad

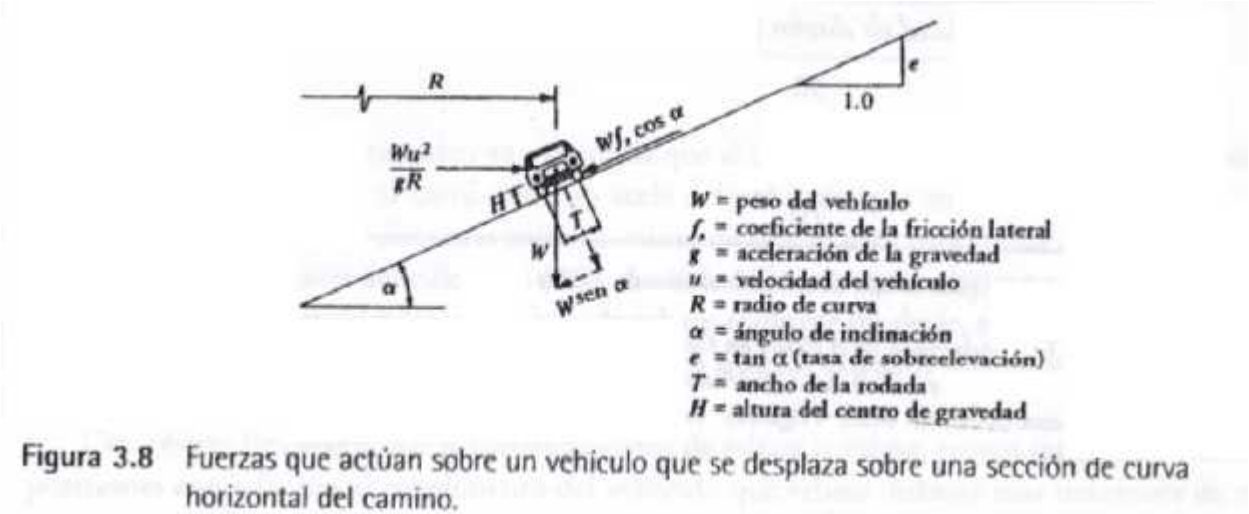


Figura 3.8 Fuerzas que actúan sobre un vehículo que se desplaza sobre una sección de curva horizontal del camino.

Figura 7. Fuerzas que actúan sobre un vehículo que se desplaza sobre una curva horizontal. [7]

Cuando el vehículo está en equilibrio respecto al declive (es decir, el vehículo se mueve hacia adelante pero no asciende ni desciende por el declive), se puede establecer una ecuación para las tres fuerzas relevantes y obtener

$$\frac{W*u^2}{g*R} * c_i = W * s_i + W * f_s * c_i \qquad (2.1.23)$$

Donde f_s=coeficiente lateral de fricción y (u²/g) = R*(tanα + f_s), lo que da

$$R = \frac{u^2}{g*(t_i + f_s)} \qquad (2.1.24)$$

La tanα, tangente del ángulo de inclinación del camino, se conoce como la tasa de sobreelevación e, por tanto la ecuación (2.1.24) se puede escribir como

$$R = \frac{u^2}{g*(e+f_s)} \qquad (2.1.25)$$

Tabla 2.1 Coeficiente de fricción lateral para diferentes velocidades de diseño. [7]

velocidades de diseño (millas/hora)	coeficiente de fricción lateral f*
30	0.16
40	0.15
60	0.12
70	0.10

*Valores límite para la casa de sobreelevación de .008 y 0.10

FUENTE Adaptado de Policy on Geometric Desing of Highways and Streets (Una policía sobre el diseño geometrico de autopistas y calles), Sociedad de Estadounidenses Estatales de Carreteras y de Transporte (American Association of State Highway and Transportation Officials). Washington, D.C. 2001. Utilizado con autorización

Con todo este tratamiento teórico mostramos enseguida la clase vehículo, hacemos notar que algunos conceptos como el radio de giro o la resistencia a la pendiente, necesitan de la clase camino, ya que la pendiente y la sobreelevación son características del camino.

2.1.5. Clase automóvil

La clase automóvil se deriva por herencia directamente de la clase vehículo, un automóvil es un vehículo, pero el recíproco no siempre es cierto. Mostramos de inmediato la clase automóvil que prácticamente hereda casi todo de la clase vehículo desarrollado anteriormente, excepto por algunas características como la tasa máxima de aceleración α, que depende de los tipos de los vehículo.

2.1.6. Clase semirremolque

Se deriva también por herencia de la clase vehículo, se adicionan o modifican las propiedades o datos miembros de la clase base, clase vehículo, para adecuarlo a un tractor semi – remolque.

2.1.7. Clase camino

La clase camino, se ha implementado con algunas características resaltantes que tiene un camino o una carretera. Estos datos miembros son la pendiente, el ancho, la longitud, el radio de giro, la sobreelevación, el coeficiente de fricción y la velocidad de diseño. Se entiende que la longitud se refiere a un tramo del camino (un carril) que tenga

la pendiente y el ancho constantes en ese tramo. Se considera la velocidad de diseño como una de las características de la clase camino, ya que es una cantidad que sirve para calcular el radio de giro de las curvas, propiedad básica de todo camino o carretera; sin embargo se advierte que estos conceptos están un poco confundidos en los autores de libros sobre ingeniería de tránsito o carreteras; cuando las ideas se quieren transformar en programas que funcionen, estas dependencias un tanto cíclicas se tienen que romper, mejorando así la comprensión de los problemas y logrando clasificarlos correctamente.

Uno de los problemas que se tenía en los libros sobre carreteras era la del radio de giro de una línea recta, que como se sabe es el infinito; había que tratar entonces con su recíproco, el grado de curvatura, que es cero para la línea recta. Sin embargo eso no es un problema para el C++, ya que define simbólicamente a esa cantidad del infinito; por tanto, no necesitamos del concepto del grado de curvatura, ya que diremos simplemente que un tramo recto de un camino tiene un radio de giro infinito, directamente en el programa.

2.2. Estudios de Ingeniería de Tránsito

2.2.1. Estudios de velocidad en el sitio

Los estudios de velocidad en el sitio se realizan para estimar la distribución de la velocidad de los vehículos en un flujo vehicular y en un lugar específico de una carretera.

Las características de velocidad que se determinen en el sitio pueden usarse para:

- Establecer parámetros para la operación y el control del tránsito, tales como zonas de velocidad (se usa la velocidad percentil 85 como el límite de velocidad en un camino), o las restricciones de paso.

- Evaluar la efectividad de los dispositivos de control de tránsito, tales como los señalamientos de mensajes variables en las zonas de trabajo.

- Verificar el efecto de los programas en vigor que monitorean la velocidad, tales como el uso del radar sonoro y de límites diferenciados de velocidad para automóviles y camiones.

- Evaluar y/o determinar lo adecuado de las características geométricas de la carretera, tales como los radios horizontales de las curvas y las longitudes verticales de las mismas.

- Evaluar el efecto de la velocidad en la seguridad de las carreteras mediante el análisis de los datos de accidentes para diferentes características de velocidad.

- Determinar las tendencias de velocidad.

- Determinar si son válidas las quejas acerca de incidentes de exceso de velocidad.

Prácticamente todos estos problemas arriba mencionados tienen que ver aparte de la recolección de datos de velocidad en el campo, la observación; que no es parte de la tesis; con su tratamiento estadístico. Si uno revisa la bibliografía acerca de los estudios dinámicos de la ingeniería de transportes, como son los estudios de la velocidad o los estudios de volúmenes de tránsito, veremos que se trata prácticamente de un caso especial o particular de la Estadística Descriptiva e Inferencial y del Cálculo de Probabilidades; es el tratamiento de datos para obtener estadísticos que puedan resumir esa colección de datos o probar hipótesis nulas o alternativas con un grado conocido de

confianza o del grado de error que somos capaces de aceptar, por tanto daremos una importancia capital a todo el desarrollo matemático logrado en este campo para producir programas que resuelven estos problemas de la Ingeniería de Transportes.

Por tanto, tenemos que estar capacitados con todas las herramientas del análisis estadístico y del cálculo de las probabilidades, así como de los modelos estadísticos que mejor se adapten a los datos. Sin más preámbulos, nos abocamos a esto.
Antes de entrar de lleno a las funciones estadísticas, tenemos que realizar un estudio completo de unas funciones especiales que nos servirán para desarrollar con éxito nuestro objetivo.

2.2.2. Función Gamma, Función Beta, Factoriales, Coeficientes Binomiales.

La función gamma es definida por la integral

$$\Gamma(z) = \int_0^\infty t^{z-1} * e^{-t} d \qquad (2.2.2.1)$$

Cuando el argumento z es un entero, la función gamma es sólo la función factorial familiar, pero compensado por 1:

$$n! = \Gamma(n+1) \qquad (2.2.2.2)$$

La función gamma satisface la relación de recurrencia

$$\Gamma(z+1) = z * \Gamma(z) \qquad (2.2.2.3)$$

Si la función es conocida para los argumentos z > 1 o, más generalmente, en el medio plano complejo Re (z) > 1, puede ser obtenida para z < 1 o Re (z) <1 por la fórmula de reflexión

$$\Gamma(1-z) = \frac{\pi}{\Gamma(z) * \sin(\pi * z)} = \frac{\pi * z}{\Gamma(1+z) * \sin(\pi * z)} \qquad (2.2.2.4)$$

Observe que $\Gamma(z)$ tiene un polo en z = 0 y en todos los valores enteros negativos de z. Hay una variedad de métodos en uso para el cálculo de la función $\Gamma(z)$ numéricamente, pero ninguno es tan limpio como la aproximación derivada por Lanczos. Este esquema es totalmente específica para la función gamma, al parecer sacado de la nada.

No vamos a intentar derivar la aproximación, sólo indicamos la fórmula resultante: Para

ciertas opciones del racional y y el entero N, y para ciertos coeficientes $c_1, c_2, \ldots, c_n$, la función gamma está dada por

$$\Gamma(z+1) = \left(z+\gamma+\frac{1}{2}\right)^{z+\frac{1}{2}} * e^{-\left(z+\gamma+\frac{1}{2}\right)} * \sqrt{2*\pi} * \left[c_0 + \frac{c_1}{z+1} + \frac{c_2}{z+2} + \cdots + \frac{c_N}{z+N} + \epsilon\right] \quad (z > 0)$$

(2.2.2.5)

Se puede ver que esto es una especie de despegue en la aproximación de Stirling, pero con una serie de correcciones que tengan en cuenta los primeros polos en el plano complejo izquierdo. La constante c_0 es casi igual a 1. El término de error es parametrizada por ϵ. Para N = 14, y un cierto conjunto de c y de γ (calculado por P. Godfrey), el error es menor que $|\epsilon| < 10^{-1}$. El evento más impresionante es el hecho de que, con estas mismas constantes, la fórmula (2.2.2.5) se aplica para la función gamma complejo, en todas partes en el medio plano complejo Re (z)> 0, logrando casi la misma precisión que en la recta real.

Es mejor implementar $ln\,\Gamma(x)$ que $\Gamma(x)$, ya que el desbordamiento superior de la última se da para un valor bastante modesto de x. A menudo, la función gamma se utiliza en los cálculos donde los grandes valores de $\Gamma(x)$ se dividen por otros grandes números, con el resultado de un valor normal y corriente. Estas operaciones normalmente se codifican como la resta de los logaritmos. Con (2.2.2.5) en la mano, podemos calcular el logaritmo de la función gamma con dos llamadas a un logaritmo y unas pocas docenas de operaciones aritméticas. Esto hace que no sea mucho más difícil que otras funciones incorporadas que damos por sentado, como sen x o e^x:

¿Cómo vamos a escribir una rutina para la función factorial n!? Generalmente la función factorial será llamada para valores enteros pequeños, y en la mayoría de las aplicaciones el mismo valor entero será llamado por muchas veces. Evidentemente, es ineficiente llamar a exp (gammln (n + 1.)) Para cada factorial requerido. Mejor es inicializar una tabla estática en la primera llamada, y hacer una búsqueda rápida en llamadas posteriores. El tamaño fijo 171 para la tabla es debido a que 170! es representable como un valor de precisión doble IEEE, pero 171! Ya se desborda. A veces también es útil saber que los factoriales hasta 22! Tienen representaciones exactas de doble precisión (52 bits de mantisa, sin contar las potencias de dos que son absorbidos en el exponente), mientras que el 23! y mayores se representan sólo aproximadamente.

Más útil en la práctica es una función que devuelve el logaritmo de un factorial, que no tiene problemas de desbordamiento. El tamaño de la tabla de logaritmos es lo que usted puede permitirse el lujo en el espacio y el tiempo de inicialización. El valor NTOP = 2000 se debe aumentar si sus argumentos enteros son a menudo más grandes.

El coeficiente binomial es definido por

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \qquad 0 \le k \le n \qquad (2.2.2.6)$$

Si su problema requiere una serie de relaciones entre los coeficientes binomiales, una buena idea es recurrir a las relaciones de recurrencia, por ejemplo,

$$\binom{n+1}{k} = \frac{n+1}{n-k+1} * \binom{n}{k} = \binom{n}{k} + \binom{n}{k-1}$$

$$\binom{n}{k+1} = \frac{n-k}{k+1} * \binom{n}{k} \qquad (2.2.2.7)$$

Por último, apartándose de las funciones combinatorias con argumentos de valores enteros, llegamos a la función beta,

$$B(z,w) = B(w,z) = \int_0^1 t^{z-1} * (1-t)^{w-1} d \qquad (2.2.2.8)$$

que está relacionada con la función gamma por

$$B(z,w) = \frac{\Gamma(z) * \Gamma(w)}{\Gamma(z+w)} \qquad (2.2.2.9)$$

2.2.3. La Función Gamma Incompleta y la Función de Error

La función gamma incompleta es definida por

$$P(a,x) \equiv \frac{\gamma(a,x)}{\Gamma(a)} = \frac{1}{\Gamma(a)} \int_0^x e^{-t} * t^{a-1} d \qquad (a > 0) \qquad (2.2.3.1)$$

Tiene los valores límite

P(a, 0) = 0 y P(a, ∞) = 1 (2.2.3.2)

La función gamma incompleta P (a, x) es monótona y (para a mayor que uno o así) se levanta de "casi cero" a "cerca-uno" en un rango centrado de x en alrededor de a - 1, y de ancho acerca de $\sqrt{a}$. Vea la figura 8.

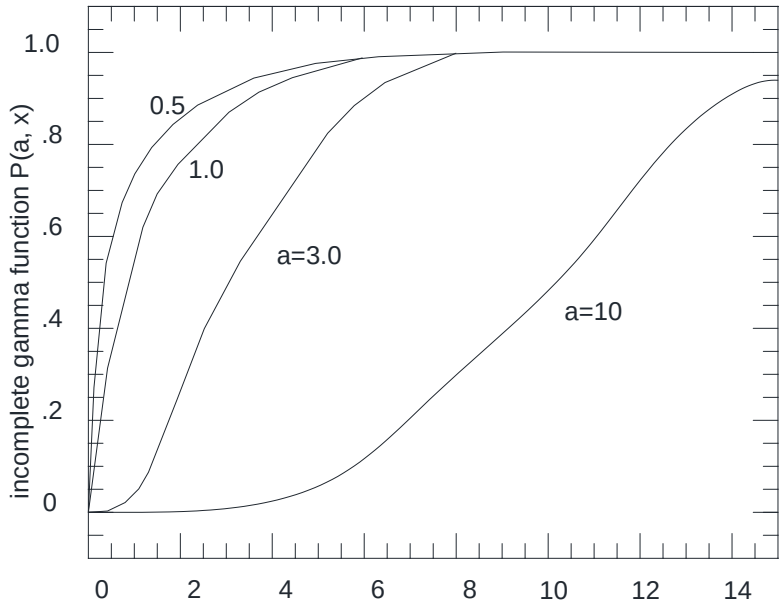


Figura 8. La función gamma incompleta P(a, x) para cuatro valores de a. [20]

El complemento de P (a, x) es también confusamente llamado una función gamma incompleta,

$$Q(a,x) \equiv 1 - P(a,x) \equiv \frac{\Gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_x^\infty e^{-t} * t^{a-1} d \qquad (a > 0) \qquad (2.2.3.3)$$

Tiene los valores límites

Q(a, 0) = 1 y Q(a, ∞) = 0 (2.2.3.4)

Hay un desarrollo en serie de γ (a, x) como sigue:

$$\gamma(a,x) = e^{-x} * x^a \sum_{n=0}^{\infty} \frac{\Gamma(a)}{\Gamma(a+1+n)} x^n \qquad (2.2.3.5)$$

Uno realmente no se necesita para calcular una nueva Γ (a + 1 + n) para cada n; más bien se utiliza la Ecuación (2.2.2.3) y el coeficiente anterior.

Un desarrollo en fracciones continuas de Γ (a, x) es

$$\Gamma(a,x) = e^{-x} * x^a * \left(\frac{1}{x+} * \frac{1-a}{1+} * \frac{1}{x+} * \frac{2-a}{1+} * \frac{2}{x+} \ldots \right) \qquad (x > 0) \qquad (2.2.3.6)$$

Es computacionalmente mejor utilizar la parte incluso de (2.2.3.6), que converge dos veces más rápido:

$$\Gamma(a,x) = e^{-x} * x^a * \left(\frac{1}{x+1-a-} * \frac{1*(1-a)}{x+3-a-} * \frac{2*(2-a)}{x+5-a-} \ldots \right) \qquad (x > 0) \qquad (2.2.3.7)$$

Resulta que (2.2.3.5) converge rápidamente para x menor que aproximadamente a + 1, mientras que (2.2.3.6) o (2.2.3.7) converge rápidamente para x mayor que aproximadamente a + 1. En estos respectivos regímenes cada uno requiere un poco más de $\sqrt{a}$ veces los términos que convergen, y esto muchos sólo cerca de x = a, donde las funciones gamma incompletas están variando más rápidamente. Para los valores moderados de a, menos de 100, por ejemplo, (2.2.3.5) y (2.2.3.7) juntos permiten la evaluación de la función para todo x. Un dividendo adicional es que nunca necesitamos para calcular un valor de la función cero restando dos números casi iguales. Algunas aplicaciones requieren P (a, x) y Q (a, x) para valores mucho más grandes de a, donde tanto la serie y la fracción continua son ineficientes. En este régimen, sin embargo, el integrando en la ecuación (2.2.3.1) cae bruscamente en ambas direcciones desde su pico, dentro de pocas veces $\sqrt{a}$. Un procedimiento eficaz es evaluar la integral directamente, con un solo paso de alto orden con la cuadratura de Gauss-Legendre que se extiende desde x lo suficientemente lejos en la cola más cercana para conseguir valores insignificantes del integrando. En realidad se trata de "medio paso," porque necesitamos las abscisas densas sólo cerca de x, no lejos de la cola donde el integrando es efectivamente cero.

Empaquetamos las diversas partes gamma incompletas en un objeto Gamma. El único estado persistente es el valor *gln*, que se fija para Γ (a) para la más reciente llamada a P (a, x) o Q (a, x). Esto es útil cuando se necesita una convención normalización diferente, por ejemplo γ (a, x) o Γ (a, x) en la ecuación (2.2.3.1) o (2.2.3.3)

2.2.4 Inversa de la Función Gamma Incompleta

En muchas aplicaciones estadísticas se necesita la inversa de la función gamma incompleta, es decir, el valor x tal que P (a, x) = p, para un valor dado 0 <= p <= 1. El método de Newton funciona bien si podemos idear una estimación inicial suficientemente buena. De hecho, este es un buen lugar para utilizar el método de Halley, ya que la segunda derivada (es decir, la primera derivada del integrando) es fácil de cálculo.

Para a <= 1, primero más o menos aproximamos $P_a \equiv P(a, 1)$:

$$P_a = P(a, 1) \approx 1 - 0.253a - 0.12a^2 \qquad 0 \le a \le 1 \qquad (2.2.3.1)$$

y luego resolvemos para x en una u otra de las aproximaciones (en bruto):

$$P(a,x) = \begin{cases} P_a * x^a, & x < 1 \\ P_a + (1 - P_a)(1 - e^{1-x}), & x \geq 1 \end{cases} \qquad (2.2.3.2)$$

2.2.5. Función de Error

La función de error y la función de error complementaria son casos especiales de la función gamma incompleta y se obtienen moderadamente eficientemente por los procedimientos anteriores. Sus definiciones son

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, d \qquad (2.2.5.1)$$

Y

$$e \quad (x) \equiv 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2}\, d \qquad (2.2.5.2)$$

Las funciones tienen los siguientes valores límites y simetrías

erf(0) = 0 erf(∞) = 1 erf(-x) = -erf(x) (2.2.5.3)

erfc(0) = 1 erfc(∞) = 0 erfc(-x) = 2 - erfc(x) (2.2.5.4)

Están relacionados con las funciones gamma incompletas por

$$\text{erf}(x) = P\left(\frac{1}{2}, x^2\right) \qquad (x \geq 0) \qquad (2.2.5.5)$$

Y

$$\text{erfc}(x) = Q\left(\frac{1}{2}, x^2\right) \qquad (x \geq 0) \qquad (2.2.5.6)$$

Un cálculo más rápido se aprovecha de una aproximación de la forma

$$\text{erfc}(z) = t * \exp[-z^2 + P(t)], \qquad z > 0 \qquad (2.2.5.7)$$

Donde $t \equiv \frac{2}{2+z}$ (2.2.5.8)

y P (t) es un polinomio de 0 <= t <= 1 que pueden encontrar por los métodos de Chebyshev. Al igual que con Gamma, la implementación es por un objeto que también incluye la función inversa, aquí una inversa para ambos erf y erfc. El método de Halley se utiliza de nuevo para las inversas (como sugiere P.J. Acklam).

}

2.2.6. Función Beta Incompleta

La función beta incompleta es definida por

$$I_x(a,b) \equiv \frac{B_x(a,b)}{B(a,b)} \equiv \frac{1}{B(a,b)} \int_0^x t^{a-1} * (1-t)^{b-1}\, d \qquad (a,b>0) \qquad (2.2.6.1)$$

Tiene los valores límite

$$I_0(a,b) = 0 \qquad I_1(a,b) = 1 \qquad (2.2.6.2)$$

y la relación de simetría

$$I_x(a,b) = 1 - I_{1-x}(b,a) \qquad (2.2.6.3)$$

Si a y b son ambos bastante mayor que uno, entonces $I_x(a,b)$ se eleva desde "casi cero" a "casi uno" de forma considerable en alrededor de x = a / (a + b). La Figura 10. Plotea la función para varios pares (a,b)

La función beta incompleta tiene un desarrollo en serie

$$I_x(a,b) = \frac{x^a * (1-x)^b}{a * B(a,b)} \left[1 + \sum_{n=0}^{\infty} \frac{B(a+1,n+1)}{B(a+b,n+1)} x^{n+1}\right] \qquad (2.2.6.4)$$

pero esto no demuestra ser muy útil en su evaluación numérica. (Observe, sin embargo, que las funciones beta en los coeficientes pueden ser evaluados para cada valor de n con sólo el valor anterior y algunos multiplicaciones, usando las ecuaciones (2.2.2.9) y (2.2.2.3))

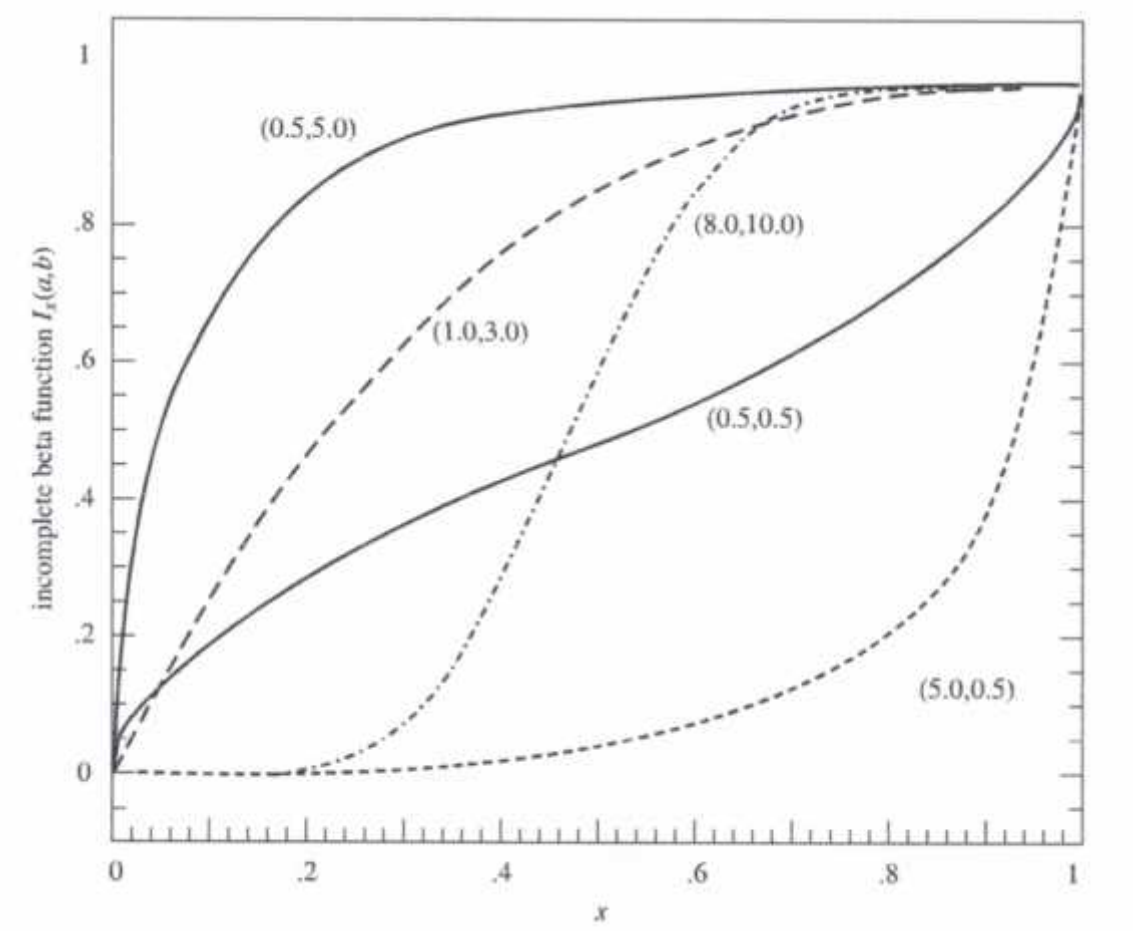


Figura 9. La función beta incompleta $I_x(a,b)$ para cinco diferentes pares de (a, b). [20]

La representación fracción continua demuestra ser mucho más útil:

$$I_x(a,b) = \frac{x^a*(1-x)^b}{a*B(a,b)}\left[\frac{1}{1+}*\frac{d_1}{1+}*\frac{d_2}{1+}\ldots\right] \qquad (2.2.6.5)$$

Donde

$$d_{2m+1} = -\frac{(a+m)*(a+b+m)*x}{(a+2m)*(a+2m+1)}$$

$$d_{2m} = \frac{m*(b-m)*x}{(a+2m-1)*(a+2m)} \qquad (2.2.6.7)$$

Esta fracción continua converge rápidamente para x < (a + 1) / (a + b + 2), excepto cuando a y b son tanto grandes, cuando se puede tomar $O(\sqrt{\min(a,b)})$ iteraciones. Para x > (a+1)/(a+b+2) sólo podemos usar la relación de simetría (2.2.6.3) para obtener, teóricamente, una computación equivalente en el que la convergencia es más rápida. Nuestra estrategia de cálculo es, pues, muy similar a la utilizada en Gamma: Usamos la fracción continuada excepto cuando a y b son tanto grandes, en cuyo caso hacemos un

solo paso de alto orden con la cuadratura de Gauss-Legendre.

También como en Gamma, codificamos una función inversa utilizando el método de Halley. Cuando a y b son ambos > = 1, la estimación inicial viene del §26.5.22 del libro "Handbook of mathematical Functions" de Abramowits, M. y Stegun, I.A 1964. Cuando cualquiera es menor que 1, la suposición proviene de primera aproximación crudamente

$$\int_0^1 t^{a-1} * (1-t)^{b-1}\, d \;\approx \frac{1}{a}\left(\frac{a}{a+b}\right)^a + \frac{1}{b}\left(\frac{b}{a+b}\right)^b \equiv S \qquad (2.2.6.7)$$

que proviene de romper la integral en t = a / (a + b) y haciendo caso omiso de un factor en el integrando en cada lado de la ruptura. Entonces Escribimos

$$I_x(a,b) \approx \begin{cases} \frac{x^a}{(S\ )} & x \le a/(a+b) \\ \frac{(1-x)^b}{(S\ )} & x > a/(a+b) \end{cases} \qquad (2.2.6.8)$$

y resolver para x en los regímenes respectivos. Mientras que la aproximación cruda, es suficiente para conseguir una buena convergencia en todos los casos.

2.2.7.1. Funciones estadísticas

Ciertas funciones especiales reciben el uso frecuente debido a su relación con las distribuciones estadísticas univariadas comunes, es decir, las densidades de probabilidad en una sola variable. En esta sección examinamos algunas de esas distribuciones comunes de manera unificada, dar, en cada caso, las rutinas para el cálculo de la función de densidad de probabilidad p (x); la función de densidad acumulada o cdf, escrito P (<x); y la inversa de la función de densidad acumulativa x (P). Se necesita la última función para encontrar los valores de x asociados con puntos porcentuales o cuantiles especificados en las pruebas de significación, por ejemplo, el 0,5%, 5%, 95% o 99,5% puntos.

2.2.7.1. Distribución Normal (o de Gauss)

Si x proviene de una distribución normal con media μ y desviación estándar σ, entonces escribimos

$$x \sim N(\mu, \sigma), \qquad \sigma > 0 \qquad (2.2.7.1.1)$$

$$p(x) = \frac{1}{\sqrt{2*\pi}*\sigma} e^{\left(-\frac{1}{2} * \left[\frac{x-\mu}{\sigma}\right]^2\right)}$$

con p (x) la función de densidad de probabilidad. Observe el uso especial de la notación "~" en esta sección, que se puede leer como "se extrae de una distribución". La varianza de la distribución es, por supuesto, σ².

La función de distribución acumulativa es la probabilidad de un valor <= x. Para la distribución normal, esto se da en términos de la función de error complementaria

$$c \equiv P(< x) \equiv \int_{-\infty}^{x} p(x')dx' = \frac{1}{2}e \left(-\frac{1}{\sqrt{2}}\left[\frac{x-\mu}{\sigma}\right]\right) \quad (2.2.7.1.2)$$

La cdf inversa puede calcularse en términos de la inversa de erfc,

$$x(P) = \mu - \sqrt{2} * \sigma * e^{-1}(2P) \quad (2.2.7.1.3)$$

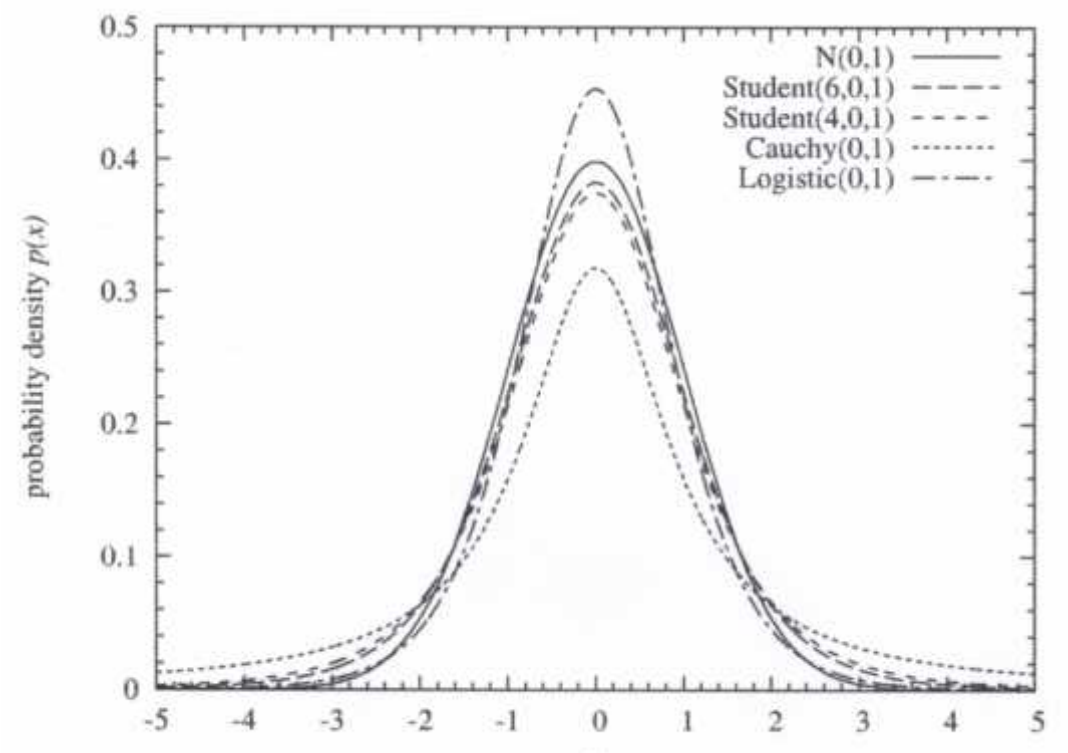


Figura 10. Ejemplos de distribuciones centralmente puntiagudos que son simétricos en la recta real. [20]

Vamos a utilizar las convenciones del código de seguridad para todas las distribuciones en esta sección. Los parámetros de las distribuciones (en este caso, μ y σ) son fijados por el constructor y luego referenciados según sea necesario por las funciones miembro. La función de densidad siempre será p(), la función de distribución acumulada es cdf(), y la inversa de cdf es invcdf (). Por lo general, compruebe los argumentos de funciones de probabilidad para la validez, ya que muchos errores en el programa pueden aparecer como por ejemplo, una probabilidad fuera del rango [0, 1].

2.2.7.2. Distribución de Cauchy

Al igual que la distribución normal, la distribución de Cauchy es una distribución simétrica con un pico en el centro, con un parámetro μ que especifica su centro y un parámetro σ que especifica su anchura. A diferencia de la distribución normal, las distribuciones de Cauchy tiene colas que se descomponen muy lentamente en el infinito, como $|x|^{-2}$, tan lentamente que ni siquiera existen momentos de orden superior que el momento cero (el área bajo la curva). Por ello, el parámetro μ es, no es, estrictamente hablando, la media, y el parámetro σ no es, técnicamente, la desviación estándar. Pero estos dos parámetros sustituyen por esos momentos como medidas de posición central y dispersión.

La definición de densidad de probabilidad es

$$x \sim C \quad hy(\mu, \sigma), \qquad \sigma > 0$$

$$p(x) = \frac{1}{\pi * \sigma}\left(1 + \left[\frac{x-\mu}{\sigma}\right]^2\right)^{-1} \qquad (2.2.7.2.1)$$

Si x ~ Cauchy(0, 1), entonces también 1/x ~ Cauchy(0, 1) y también $(a * x + b)^{-1} \sim C \quad hy(-\frac{b}{a}, \frac{1}{a})$

La función de distribución acumulativa cdf está dada por

$$c \equiv P(< x) \equiv \int_{-\infty}^{x} p(x')dx' = \frac{1}{2} + \frac{1}{\pi}\tan^{-1}\left(\frac{x-\mu}{\sigma}\right) \qquad (2.2.7.2.2)$$

La inversa de la función de distribución acumulativa está dada por

$$x(P) = \mu + \sigma * \tan\left(\pi * \left[P - \frac{1}{2}\right]\right) \qquad (2.2.7.2.3)$$

La Figura 11 muestra Cauchy (0,1) en comparación con la distribución normal N (0, 1), así como varias otras distribuciones de forma similar, que se discuten a continuación.

La distribución de Cauchy a veces se llama la distribución de Lorentz.

2.2.7.3. Distribución t de Student

Una generalización de la distribución de Cauchy es la distribución t de Student, llamado así por el estadístico William Gosset de inicios del siglo 20, que publicó bajo el nombre de "Estudiante", porque su empleador, Guinnes Breweries, le obligaba a usar un seudónimo. Al igual que la distribución de Cauchy, la distribución t de Student tiene colas que se descomponen, en el infinito, pero tiene un parámetro ν adicional que especifica cómo se desintegran rápidamente, es decir, como $|t|^{-(\nu+1)}$. Cuando ν es un número entero, el número de momentos convergentes, incluyendo el cero, es por lo tanto ν.

La definición de densidad de probabilidad (convencionalmente escrito en una variable t en lugar de x) es

$$t \sim S \qquad (\nu, \mu, \sigma), \qquad \nu > 0, \sigma > 0 \tag{2.2.7.3.1}$$

$$p(t) = \frac{\Gamma\left(\frac{1}{2}[\nu+1]\right)}{\Gamma\left(\frac{1}{2}\nu\right)\sqrt{\nu}\,\sigma}\left(1 + \frac{1}{\nu}\left[\frac{t-\mu}{\sigma}\right]^2\right)^{-\frac{1}{2}(\nu+1)}$$

La distribución de Cauchy se obtiene en el caso ν = 1. En el límite opuesto, ν → ∞, se obtiene la distribución normal. En los días anteriores a la computadora, esta fue la base de varios esquemas de aproximación a la distribución normal, ahora todos en general son irrelevantes.

La figura 11 muestra ejemplos de la distribución t de Student para ν = 1 (Cauchy), ν = 4, y ν = 6. La aproximación a la distribución normal es evidente.

La media de Student (ν, μ, σ) es (por simetría) μ. La varianza no es σ², sino

$$V\left(S \qquad (\nu, \mu, \sigma)\right) = \frac{\nu}{\nu-2}\sigma^2 \tag{2.2.7.3.2}$$

La cdf está dada por una función beta incompleta. Si dejamos que

$$x \equiv \frac{\nu}{\nu + \left(\frac{t-\mu}{\sigma}\right)^2} \tag{2.2.7.3}$$

Entonces

$$c \equiv P(<x) \equiv \int_{-\infty}^{t} p(t')\,dt' = \begin{cases} \frac{1}{2} I_x\left(\frac{1}{2}\nu, \frac{1}{2}\right), & t \le \mu \\ 1 - \frac{1}{2} I_x\left(\frac{1}{2}\nu, \frac{1}{2}\right), & t \ge \mu \end{cases} \qquad (2.2.7.4)$$

En la práctica, la cdf t de Student en la forma de arriba se utiliza muy poco, ya que la mayoría de las pruebas estadísticas utilizando t de Student son bilaterales. Convencionalmente, la función de dos colas A (t | ν) se define (únicamente) para el caso μ= 0 y σ = 1 por

$$A(t|\nu) \equiv \int_{-t}^{t} p(t')dt' = 1 - I_x\left(\frac{1}{2}\nu, \frac{1}{2}\right) \qquad (2.2.7.5)$$

con x como se indica anteriormente. La estadística A (t | ν) se utiliza en particular en las pruebas de si dos distribuciones observadas tienen la misma media. El código siguiente implementa ambas ecuaciones (2.2.7.4) y (2.2.7.5), así como sus inversas.

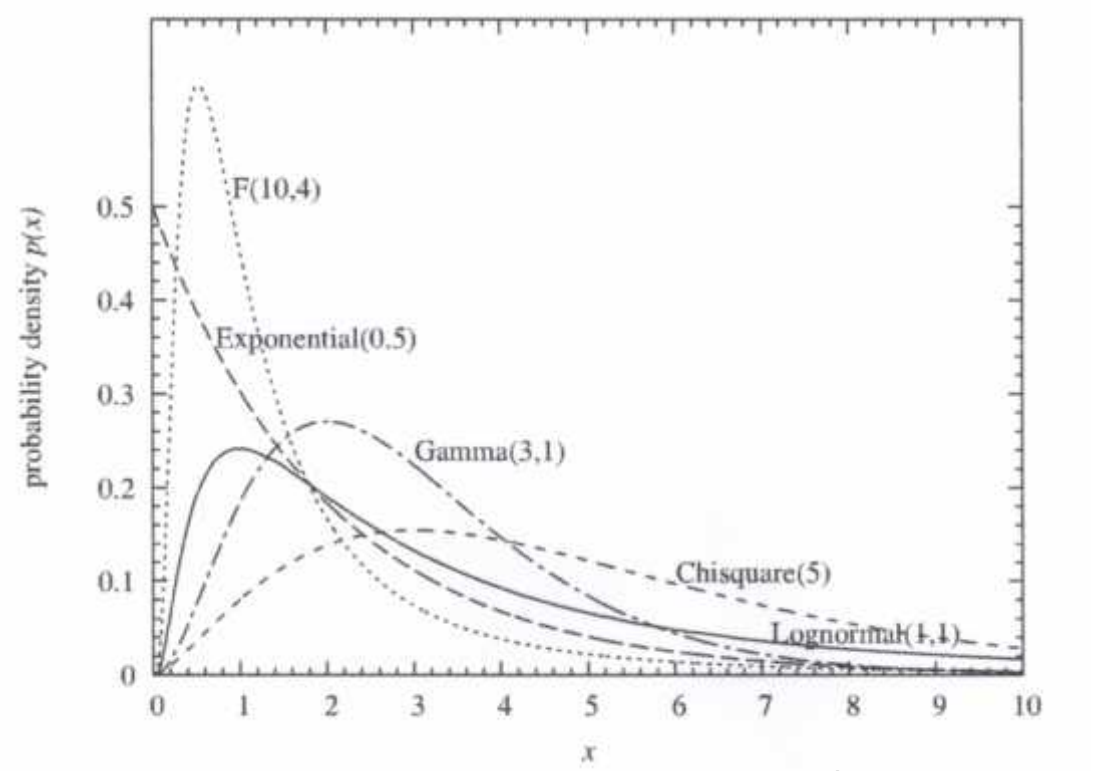


Figura 11. Ejemplos de distribuciones comunes sobre la media línea x > 0. [20]

2.2.7.4. La Distribución Chi – Cuadrado

La distribución chi-cuadrado (o χ^2) tiene un solo parámetro ν > 0 que controla tanto la ubicación y ancho de su pico. En la mayoría de aplicaciones ν es un número entero y se refiere como el número de grados de libertad.

La definición de densidad de probabilidad es

$$\chi^2 \sim Chis \quad (\nu), \quad \nu > 0$$

$$p(\chi^2)d\chi^2 = \frac{1}{2^{\frac{1}{2}\nu}\Gamma\left(\frac{1}{2}\nu\right)}(\chi^2)^{\frac{1}{2}\nu-1}e^{\left(-\frac{1}{2}\chi^2\right)}d\chi^2, \quad \chi^2 > 0 \quad (2.2.7.4.1)$$

donde hemos escrito los diferenciales $d\chi^2$ simplemente para destacar que χ^2, no χ, debe ser visto como la variable independiente.

La media y la varianza son dados por
Media{Chisquare(ν)} = ν

(2.2.7.4.2)

Varianza{Chisquare(ν)} = 2*ν

Cuando ν ≥ 2 hay una sola moda en χ^2 = ν – 2.

La distribución de chi-cuadrado en realidad es sólo un caso especial de la distribución gamma, a continuación, por lo que su función de distribución está dada por una función gamma incompleta P (a, x),

$$c \equiv P(<\chi^2) \equiv P(\chi^2|\nu) \equiv \int_0^{\chi^2} p(\chi^{2\prime})d\chi^{2\prime} = P\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right) \quad (2.2.7.4.3)$$

Una frecuencia también ve el complemento de la función de distribución, que se puede calcular ya sea desde la función gamma incompleta P (a, x), o de su complemento Q (a, x) (a menudo más precisa si P es muy cercano a 1):

$$Q(\chi^2|\nu) \equiv 1 - P(\chi^2|\nu) = 1 - P\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right) \equiv Q\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right) \quad (2.2.7.4.4)$$

La cdf inversa se da en términos de la función que es la inversa de P (a, x) en su segundo argumento, que aquí designamos

$$x(P) = 2 * P^{-1}\left(\frac{\nu}{2}, P\right) \quad (2.2.7.4.5)$$

2.2.7.5. Distribución Gamma

La distribución gamma es definido por

X ~ Gamma(α, β), α > 0, β > 0

(2.2.7.5.1)

$$p(x) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta * x}, \qquad x > 0$$

La distribución exponencial es el caso especial con α = 1. La distribución chi-cuadrado es el caso especial con α = ν / 2 y β = 1/2.

La media y la varianza son dadas por,

Media{Gamma(α, β)} = α / β

(2.2.7.5.2)

Varianza{Gamma(α, β)} = α / β²

Cuando α ≥ 1 hay una única modo en x = (α – 1) / β.

Evidentemente, la cdf es la función gamma incompleta

$$c \equiv P(< x) \equiv \int_0^x p(x')dx' = P(\alpha, \beta * x) \qquad (2.2.7.5.3)$$

mientras que la inversa de la función de distribución acumulativa se da en términos de la inversa de P (a, x) en su segundo argumento por

$$x(P) = \frac{1}{\beta} P^{-1}(\alpha, P) \qquad (2.2.7.5.4)$$

2.2.7.6 Distribución – F

La distribución F está parametrizado por dos valores positivos ν_1 y ν_2, usualmente (pero no siempre) enteros.

La definición de densidad de probabilidad es

$$F \sim F(\nu_1, \nu_2), \qquad \nu_1 > 0, \quad \nu_2 > 0$$

$$p(F) = \frac{\nu_1^{\frac{1}{2}*\nu_1} * \nu_2^{\frac{1}{2}*\nu_2}}{B\left(\frac{1}{2}*\nu_1, \frac{1}{2}*\nu_2\right)} * \frac{F^{\frac{1}{2}*\nu_1 - 1}}{(\nu_2 + \nu_1 * F)^{(\nu_1+\nu_2)/2}} \qquad (2.2.7.6.1)$$

Donde B(a, b) denota la función beta. La media y la varianza están dados por

$$M \quad \{F(\nu_1, \nu_2)\} = \frac{\nu_2}{\nu_2 - 2}, \qquad \nu_2 > 2$$

$$V \quad \{F(\nu_1, \nu_2)\} = \frac{2 * \nu_2^2 * (\nu_1 + \nu_2 - 2)}{\nu_1 * (\nu_2 - 2)^2 (\nu_2 - 4)}, \qquad \nu_2 > 4$$

(2.2.7.6.2)

Cuando $\nu_1 > 2$ hay una única moda

$$F = \frac{\nu_2 * (\nu_1 - 2)}{\nu_1 * (\nu_2 + 2)} \qquad (2.2.7.6.3)$$

Para ν_1 fijo, si $\nu_2 \to \infty$, la distribución F se convierte en una distribución chi-cuadrado, a saber

$$\lim_{\nu_2 \to \infty} F(\nu_1, \nu_2) \cong \frac{1}{\nu_1} Chis \qquad (\nu_1) \qquad (2.2.7.6.4)$$

donde $\cong$ significa "son distribuciones idénticas"

La cdf de la distribución F está dada en términos de la función beta incompleta $I_x(a, b)$ por

$$c \quad \equiv P(< x) \equiv \int_0^x p(x')dx' = I_{\nu_1 * F/(\nu_2 + \nu_1 * F)} \left(\frac{1}{2} * \nu_1, \frac{1}{2} * \nu_2 \right) \qquad (2.2.7.6.5)$$

Mientras que la inversa de cdf está dado en términos de la inversa de $I_x(a, b)$ en su argumento por

$$u \equiv I_P^{-1} \left(\frac{1}{2} * \nu_1, \frac{1}{2} * \nu_2 \right)$$

$$x(P) = \frac{\nu_2 * u}{\nu_1 * (1 - u)} \qquad (2.2.7.6.6)$$

Un uso frecuente de la distribución F es para probar si dos muestras observadas tienen la misma varianza

2.2.7.7. Distribución Beta

La distribución beta es definido sobre un intervalo unitario 0 < x < 1 por

$$x \sim B \quad (\alpha, \beta), \qquad \alpha > 0, \quad \beta > 0 \tag{2.2.7.7.1}$$

$$p(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1} * (1-x)^{\beta-1}, \qquad 0 < x < 1$$

Cuando α > 1 y β > 1, hay una única moda dada por (α -1) /(α + β – 2). Cuando α < 1 y β < 1, la función de distribución es “en forma de U” con un mínimo en este mismo valor. En otros casos no hay ni un máximo ni un mínimo.

En los datos límite β se hace grande a medida que α se mantiene fijo, toda la acción en los cambios de distribución beta hacia x = 0, y la función de densidad toma la forma de una distribución gamma. Más precisamente,

$$\lim_{\beta\to\infty} \beta * B \quad (\alpha, \beta) \cong G \quad (\alpha, 1) \tag{2.2.7.7.2}$$

La cdf es la función beta incompleta

$$c \quad \equiv P(< x) \equiv \int_0^x p(x')dx' = I_x(\alpha, \beta) \tag{2.2.7.7.3}$$

Mientras la inversa de cdf está dado en términos de la inversa de $I_x(\alpha, \beta)$ en su argumento de subíndice

$$x(P) = I_P^{-1}(\alpha, \beta) \tag{2.2.7.7.4}$$

2.2.7.8. La Distribución de Kolmogorov – Smirnov

La distribución de Kolmogorov – Smirnov o KS, definido para z positiva, es clave para una prueba estadística importante que se discute más adelante. Su función de densidad de probabilidad no entra directamente en la prueba y está prácticamente nunca escrito. Lo que se necesita normalmente para calcular es la función de distribución acumulativa cdf, que se denota $P_K\ (z)$, o su complemento, $Q_K\ (z) \equiv 1 - P_K\ (z)$

La cdf $P_K\ (z)$ es definida por la serie

$$P_K\ (z) = 1 - 2 * \sum_{j=1}^{\infty}(-1)^{j-1} * e\quad(-2 * j^2 * z^2) \qquad (2.2.7.8.1)$$

O por la serie equivalente

$$P_K\ (z) = \frac{\sqrt{2*\pi}}{z} * \sum_{j=1}^{\infty} e\quad\left(-\frac{(2*j-1)^2*\pi^2}{8*z^2}\right) \qquad (2.2.7.8.2)$$

Los valores límite son lo que se espera para los cdf llamados "P" "Q":

$$P_K\ (0) = 0 \qquad P_K\ (\infty) = 1$$
$$Q_K\ (0) = 1 \qquad Q_K\ (\infty) = 0 \qquad (2.2.7.8.3)$$

Tanto la serie (2.2.7.8.1) y (2.2.7.8.2) son convergentes para todo z> 0. Por otra parte, para cualquier z, una u otra serie converge muy rápidamente, lo que requiere no más de tres términos para obtener IEEE doble precisión, precisión fraccionaria. Un buen lugar para cambiar de una serie a la otra es en z ≈ 1.18. Esto hace que las funciones KS computables por un solo exponencial y un pequeño número de operaciones aritméticas (ver el código de abajo).

Obtener las funciones inversas $P_K^{-1}(P)$ y $Q_K^{-1}(Q)$, que devuelve un valor de z de P o Q, es un poco más complicado. Para Q <= 0,3 (es decir, P ≥ 0.7), una iteración basada en (2.2.7.8.1) funciona muy bien:

$$x_0 \equiv 0$$
$$x_{i+1} = \frac{1}{2}Q + x_i^4 - x_i^9 + x_i^1 \ - x_i^2 \ + \cdots \qquad (2.2.7.8.4)$$
$$z(Q) = \sqrt{-\frac{1}{2}l\alpha\ (x_\infty)}$$

Para x ≤ 0.06 usted necesita las primeras dos potencias de x_i.

Para mayores valores de Q, es decir, P <= 0.7, el número de potencias de x requerida rápidamente se vuelve excesiva. Un enfoque útil es escribir (2.2.7.8.4.2) como

$$y\quad(y) = -\frac{\pi P^2}{8} * \left(1 + y^4 + y^1 \ + \cdots + y^{2j(j-1)} + \cdots\right)^{-1}$$
$$z(P) = \frac{\pi/2}{\sqrt{-l\alpha\ (y)}} \qquad (2.2.7.8.5)$$

Si podemos obtener una buena aproximación inicial suficiente para y, podemos resolver la primera ecuación en (2.2.7.8.5) por una variante del método de Halley: Use valores de y de la iteración anterior en el lado derecho de (2.2.7.8.5), y el uso Halley sólo para la pieza ylog(y), de manera que la primera y la segunda derivadas son funciones analíticamente simples.

Una buena aproximación inicial se obtiene mediante el uso de la función inversa a ylog (y) (la función invxlogx) con el argumento -π* P² / 8. El número de iteraciones dentro de la función invxlogx y el bucle Halley es nunca más de media docena en cada uno, a menudo menos. El código para las funciones KS y sus inversas siguen.

2.2.7.9. Distribución de Poisson

La epónima distribución de Poisson fue derivado por Poisson en 1837. Se aplica a un proceso en el que se producen, eventos correlacionados discretos a una tasa por unidad de tiempo. Si, por un período determinado, λ es la media esperada del número de eventos, entonces la distribución de probabilidad de ver exactamente k eventos, k ≥ 0, puede escribirse como

$$k \sim P \qquad (\lambda), \qquad \lambda > 0$$

(2.2.7.9.1)

$$p(k) = \frac{1}{k!}\lambda^k e^{-\lambda}, \qquad k = 0, 1, \ldots$$

Evidentemente $\sum_k p(k) = 1$, ya que los factores k-dependiente en (2.2.7.9.1) es el desarrollo en serie de e^{λ}.

La media y la varianza de Poisson (λ) son ambos λ. Hay una solo moda en k =⌊λ⌋, es decir, en λ redondeará a un entero.

La cdf de la distribución de Poisson es una función gamma incompleta Q (a, x),

$$P_{\lambda}(< k) = Q(k, \lambda) \qquad (2.2.7.9.2)$$

Desde que k es discreta, $P_{\lambda}(< k)$es por supuesto diferente de $P_{\lambda}(\leq k)$, siendo este último determinado por

$$P_{\lambda}(\leq k) = Q(k + 1, \lambda) \qquad (2.2.7.9.3)$$

(Cuidado: algunas referencias definen $P_\lambda(\leq k)$, como el cdf, en lugar de $P_\lambda(< k)$) Algunos valores particulares son

$P_\lambda(< 0) = 0 \qquad P_\lambda(< 1) = e^{-\lambda} \qquad P_\lambda(< \infty) = 1$ (2.2.7.9.4)

Algunas otras relaciones que implican las funciones gamma incompleta Q (a, x) y P (a, x) son

$P_\lambda(\geq k) = P(k, \lambda) = 1 - Q(k, \lambda)$

(2.2.7.9.5)

P_λ(>k)= $P(k + 1, \lambda) = 1 - Q(k + 1, \lambda)$

Debido a la discontinuidad en k, el inverso de la cdf se debe definir con cierto cuidado: Dado un valor P, definimos $k_\lambda(P)$ como el número entero tal que

$P_\lambda(< k) \leq P < P_\lambda(\leq k)$ (2.2.7.9.6)

En aras de la concisión, el código de abajo engaña un poco y permite la derecha < sea <=. Si usted puede ser el suministro de P de que son exactos de $P_\lambda(< k)$, entonces usted necesita para comprobar $k_\lambda(P)$ como regresó, y $k_\lambda(P)$+ 1. (Esto esencialmente nunca sucederá para P "redondo" como 0,95, 0,99, etc.)

2.2.7.10. Distribución Binomial

Al igual que la distribución de Poisson, la distribución binomial es una distribución discreta sobre k ≥ 0. Tiene dos parámetros, n ≥ 1, el "tamaño de la muestra" o el valor máximo para el que k puede ser distinto de cero; y p, la "probabilidad de eventos" (que no debe confundirse con p (k), la probabilidad de un k particular). Escribimos

$k \sim B\ n\alpha \qquad (n, p), \qquad n \geq,\ 0 < p < 1$

(2.2.7.10.1)

$p(k) = \binom{n}{k} p^k (1 - p)^{n-k}, \qquad k = 0, 1, \ldots, n$

Donde $\binom{n}{k}$ es, por supuesto, el coeficiente binomial

La media y la varianza están dados por

Media {Binomial(n, p) } = n * p

(2.2.7.10.2)

Varianza {Binomial(n, p)} = n * p * (1 - p)

Hay una única moda en el valor de k que satisface

(n + 1) * p – 1 < k ≤ (n + 1) * p (2.2.7.10.3)

La distribución es simétrica si y sólo si p = 1/2. De lo contrario, tiene asimetría positiva para p <1/2 y negativo para p > 1/2.

La distribución de Poisson se obtiene de la distribución binomial en el límite de n → ∞, p→ 0 con el n * p restante finito. Más precisamente,

$$\lim_{n\to\infty} Bi \qquad (n, \frac{\lambda}{n}) \cong P \quad so \quad (\lambda) \qquad (2.2.7.10.4)$$

La cdf de la distribución binomial se puede calcular a partir de la función beta incompleta $I_x(a, b)$,

$$P(< k) = 1 - I_p(k, n - k + 1) \qquad (2.2.7.10.5)$$

así que también tenemos (de forma análoga a la distribución de Poisson)

$$P(\leq k) = 1 - I_p(k + 1, n - k)$$
$$P(> k) = I_p(k + 1, n - k) \qquad (2.2.7.10.6)$$
$$P(\geq k) = I_p(k, n - k + 1)$$

(Cuidado: algunas referencias definen $P(\leq k)$ como la cdf, en lugar de P(< k) Algunos valores particulares son

P(< 0) = 0 P (< [n+1]) = 1 (2.2.7.10.7)

La inversa de cdf se define exactamente como para la distribución de Poisson, por encima, y con la misma pequeña advertencia sobre el código.

2.3. DESCRIPCIÓN ESTADÍSTICA DE LOS DATOS

En este capítulo y en el siguiente, el concepto de datos entra en el debate de manera más prominente que antes.

Los datos consisten en números, por supuesto. Sin embargo, estos números se asignan a la computadora, no es producido por ella. Estos son los números a ser tratados con respeto considerable, ni para ser manipulado, ni sometidos a un proceso computacional cuyo carácter no entiende completamente. Usted está bien asesorado para adquirir una reverencia por los datos, en lugar diferente de la actitud "deportiva" que a veces es admisible, o incluso loable, en otra tarea numérica.

El análisis de los datos implica inevitablemente cierto tráfico con el campo de la estadística, esa maravillosa zona gris que no es del todo una rama de las matemáticas - y tan ciertamente no es una rama de la ciencia. En las siguientes secciones, se encontrará repetidamente el siguiente paradigma, generalmente se llama una prueba de una cola o prueba de p-valor:
Aplicar alguna fórmula a los datos para calcular "una estadística"

Calcular donde el valor de esa estadística cae en una distribución de probabilidad que se calcula sobre la base de algunas "hipótesis nula"

Si se cae en un lugar muy poco probable, salida en un extremo de la distribución, la conclusión de que la hipótesis nula es falsa para el conjunto de datos.

Si una estadística cae en una parte razonable de la distribución, no se debe cometer el error de concluir que la hipótesis nula se ha "verificado" o se "probó". ¡Esa es la maldición de la estadística, que nunca puede probar las cosas, sólo refutarlas! A lo sumo, se puede fundamentar una hipótesis descartando, estadísticamente, toda una larga lista de hipótesis competidoras, todas las que alguna vez se ha propuesto. Después de un tiempo sus adversarios y competidores abandonarán tratando de pensar en hipótesis alternativas, o de lo contrario envejecer y morir, y entonces su hipótesis se aceptará. ¡Parece una locura, lo sabemos, pero así es como funciona la ciencia!

En esta tesis se hace una distinción un tanto arbitraria entre los procedimientos de análisis de datos que son independientes del modelo y los que son dependientes del modelo. En la primera categoría, se incluyen los llamados estadísticos descriptivos que caracterizan a un dato en términos generales: la media, la varianza, y así sucesivamente. También se incluyen las pruebas estadísticas que tratan de establecer y medir un grado

de correlación entre dos conjuntos de datos.

En la otra categoría, las estadísticas que dependen del modelo, que agrupan a todo el tema de ajuste de datos a una teoría, la estimación de parámetros, por mínimos cuadrados encajados, y así sucesivamente.

2.3.1. Momentos de una Distribución: Media, Varianza, Asimetría y así sucesivamente

Cuando un conjunto de valores tiene una suficientemente fuerte tendencia central, es decir, una tendencia a agruparse alrededor de algún valor en particular, entonces puede ser útil para caracterizar el conjunto por unos pocos números que están relacionados con sus momentos, las sumas de potencias enteras de sus valores.

El más conocido es la media de los valores $x_0, \ldots, x_{N-1}$

$$\bar{x} = \frac{1}{N}\sum_{j=0}^{N-1} x_j \qquad (2.3.1.1)$$

que estima el valor alrededor del cual se produce la agrupación central. Observe el uso de una barra superior para denotar la media. Usted debe ser consciente de que la media no es el único estimador disponible de esta cantidad, ni es necesariamente la mejor. Para los valores extraídos de una distribución de probabilidad con muy amplias "colas", la media puede converger poco, o nada en absoluto, a medida que aumenta el número de puntos muestreados. Estimadores alternativos, la mediana y la moda, se mencionan al final de esta sección.

Habiendo caracterizado el valor central de una distribución, una nueva caracterización convencionalmente es, su "ancho" o "variabilidad" en torno a ese valor. Aquí, de nuevo, más de una medida está disponible. El más común es la varianza,

$$V \quad (x_0, \ldots, x_{N-1}) = \frac{1}{N-1}\sum_{j=0}^{N-1}\left(x_j - \bar{x}\right)^2 \qquad (2.3.1.2)$$

o su raíz cuadrada, la desviación estándar,

$$\sigma(x_0, \ldots, x_{N-1}) = \sqrt{V \quad (x_0, \ldots, x_{N-1})} \qquad (2.3.1.3)$$

La ecuación (2.3.1.2) calcula la desviación media al cuadrado de x desde su valor medio. Hay una larga historia acerca de por qué el denominador de (2.3.1.2) es N - 1 en lugar

de N. Si usted nunca ha oído esa historia, usted debe consultar cualquier buen texto de estadística. Aquí vamos a estar contentos en señalar que la N - 1 se debe cambiar a N si alguna vez en la situación de la medición de la varianza de una distribución cuya media $\bar{x}$ se conoce a priori en lugar de ser estimada a partir de los datos. (También podríamos comentar que si la diferencia entre N y N - 1 siempre es importante para usted, entonces usted está probablemente para nada bueno de todos modos -. Por ejemplo, tratando de corroborar una hipótesis cuestionable con datos marginales)

Si calculamos la ecuación (2.3.1.1) muchas veces con diferentes conjuntos de datos muestreados (teniendo cada conjunto N valores), los valores $\bar{x}$, ellos mismos tienen una desviación estándar. Esto se llama el error estándar de las medias estimadas $\bar{x}$. Cuando la distribución subyacente es gaussiana, se da aproximadamente en $\sigma/\sqrt{N}$. Correspondientemente, hay un error estándar de la varianza estimada, la ecuación (2.3.1.2), que es aproximadamente $\sigma^2 * \sqrt{2/N}$, y un error estándar estimado para la σ, la ecuación (2.3.1.3), que es aproximadamente $\sigma/\sqrt{2N}$.

Como la media depende del primer momento de los datos, así que la varianza y la desviación estándar dependen del segundo momento. No es raro que, en la vida real, que se ocupa de una distribución cuyo segundo momento no existe (es decir, es infinito). En este caso, la varianza o la desviación estándar es inútil como una medida de la anchura de los datos alrededor de su valor central: Los valores obtenidos a partir de las ecuaciones (2.3.1.2) o (2.3.1.3) no convergen con un mayor número de puntos, ni muestran cualquier consistencia del conjunto de los datos que establecen de que los datos extraídos son de la misma distribución. Esto puede ocurrir incluso cuando la anchura del pico parece, a simple vista, perfectamente finito. Un estimador más robusto de la anchura es la desviación media o desviación absoluta media, definida por

$$A \quad (x_0, \ldots, x_{N-1}) = \frac{1}{N}\sum_{j=0}^{N-1}\left|x_j - \bar{x}\right| \qquad (2.3.1.4)$$

A menudo se sustituye la mediana de la muestra para $\bar{x}$ en la ecuación (2.3.1.4). Para cualquier muestra fija, la mediana de hecho minimiza la desviación media absoluta.

Los estadísticos han olido históricamente en el uso de (2.3.1.4) en lugar de (2.3.1.2), ya que los soportes de valor absoluto en (2.3.1.4) son "no analítica" y hacen más difícil teorema demostrando. En los últimos años, sin embargo, la moda ha cambiado, y el tema de la estimación robusta (es decir, la estimación de distribuciones amplias con un número importante de puntos "atípicos") se ha convertido en muy popular e importante. Momentos más altos, o estadísticas que implican mayores potencias de los datos de entrada, casi siempre son menos robustos que los momentos más bajos o estadísticas

que involucran sólo sumas lineales o (el momento más bajo de todos) contando.

Siendo ese el caso, la asimetría o tercer momento, y la curtosis o cuarto momento se deben utilizar con precaución o, mejor aún, no usarlos.

La asimetría caracteriza el grado de asimetría de una distribución alrededor de su media. Mientras que la media, la desviación estándar y la desviación media son cantidades dimensionales, es decir, tienen las mismas unidades que la cantidades x_j medida, la asimetría se define convencionalmente de tal manera como para que sea adimensional. Es un número puro que caracteriza sólo la forma de la distribución. La definición usual es

$$S \quad (x_0, \ldots, x_{N-1}) = \frac{1}{N}\sum_{j=0}^{N-1}\left[\frac{x_j - \bar{x}}{\sigma}\right]^3 \qquad (2.3.1.5)$$

Un valor positivo de la asimetría significa una distribución con una cola asimétrica que se extiende hacia fuera hacia x más positivos; un valor negativo significa una distribución cuya cola se extiende hacia fuera x más negativa. (Véase la Figura 12)

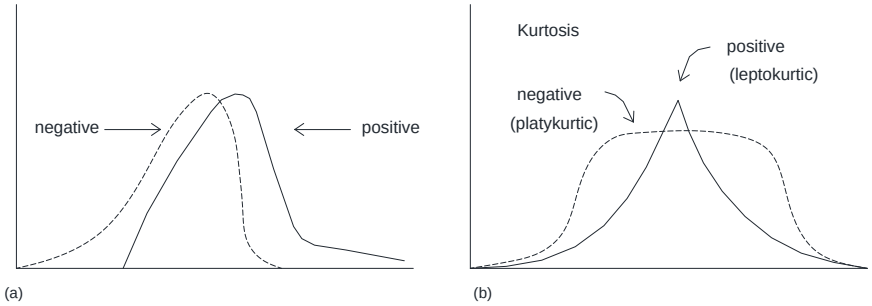


Figura 12. a) Asimetría o tercer momento b) Curtosis o cuarto momento [20]

La curtosis es también una cantidad adimensional. Mide la piquitud o planitud relativa de una distribución. ¿En relación a qué? ¡A una distribución normal!

La definición convencional de la curtosis es:

$$K \quad (x_0, \ldots, x_{N-1}) = \left\{\frac{1}{N}\sum_{j=0}^{N-1}\left[\frac{x_j - \bar{x}}{\sigma}\right]^4\right\} - 3 \qquad (2.3.1.6)$$

Donde el término -3 hace el valor cero para una distribución normal.

2.3.2. ¿Tienen Dos Distribuciones las Mismas Medias o Varianzas?

No es raro que queremos saber si dos distribuciones tienen la misma media. Por ejemplo, un primer conjunto de valores medidos puede haber sido realizado rigurosamente antes de algún evento, un segundo conjunto después de ella. Queremos saber si el evento, un "tratamiento" o un "cambio de un parámetro de control", marcaron la diferencia.

Nuestro primer pensamiento es preguntarse "cuántas desviaciones estándar" una media muestral es de otra. Ese número puede ser de hecho una cosa útil saber. No se refieren a la fuerza o la "importancia" de una diferencia de medias si esa diferencia es genuina. Sin embargo, por sí mismo, no dice nada acerca de si la diferencia es genuina, es decir, estadísticamente significativa. Una diferencia de las medias puede ser muy pequeña en comparación con la desviación estándar, y sin embargo muy significativa, si el número de puntos de datos es grande. Por el contrario, la diferencia puede ser moderadamente grande, pero no significativa, si los datos son escasos. Nos reuniremos estos conceptos distintos de la fuerza y significación varias veces en las próximas secciones.

Una cantidad que mide la importancia de una diferencia de medias no es el número de desviaciones estándar que están separados, pero si el número de los llamados errores estándar que están separados. El error estándar de un conjunto de valores mide la precisión con la que la media de la muestra estima de la población (o "true") o de la verdadera media. Típicamente, el error estándar es igual a la desviación estándar de la muestra dividida por la raíz cuadrada del número de puntos en la muestra.

2.3.2.1. Prueba t de Student para Diferencias de Medias Significativas

La aplicación de los conceptos de error estándar, la estadística convencional para medir la importancia de la diferencia de las medias se denomina t de Student. Cuando se piensa que las dos distribuciones tienen la misma varianza, pero posiblemente diferentes medias, entonces la t de Student se calcula como sigue: En primer lugar, estimar el error estándar de la diferencia de las medias, SD, de la "varianza conjunta" por la fórmula

$$S_D = \sqrt{\frac{\sum_{i \in A}(x_i - \overline{x_A})^2 + \sum_{j \in B}(x_j - \overline{x_B})^2}{N_A + N_B - 2} \cdot \left(\frac{1}{N_A} + \frac{1}{N_B}\right)} \qquad (2.3.2.1)$$

donde cada suma es sobre los puntos en una muestra, la primera o segunda; cada media asimismo se refiere a una muestra o la otra; N_A y N_B son el número de puntos en la primera y segunda muestras, respectivamente. En segundo lugar, calcular t por la fórmula

$$t = \frac{\overline{x_A} - \overline{x_B}}{s_D} \qquad (2.3.2.2)$$

En tercer lugar, evaluar el p-valor o la importancia de este valor de t para la distribución de Student con $N_A + N_B - 2$ grados de libertad, por la ecuación (2.2.7.5).

El valor p es un número entre cero y uno. Es la probabilidad de que | t | podría ser tan grande o grande sólo por casualidad, para distribuciones con medias iguales. Por lo tanto, un pequeño valor numérico del p-valor (0.01 o 0.001) significa que la diferencia observada es "muy significativa". La función A (t | v) en la ecuación (2.2.7.5) es uno menos el valor de p.

El siguiente caso a considerar es que las dos distribuciones tienen significativamente diferentes varianzas, pero sin embargo se quiere saber si sus medias son las mismas o diferentes. (¡Un tratamiento para la calvicie ha causado a algunos pacientes a perder todo el pelo y volvió a otros en hombres lobo, pero queremos saber si ayuda a curar la calvicie en promedio!) Sospeche de la prueba t de varianzas desiguales: Si dos distribuciones tienen muy diferentes varianzas, entonces ellos también pueden ser sustancialmente diferentes en forma; en ese caso, la diferencia de la medias pueden no ser algo particularmente útil saber.

Para averiguar si los dos conjuntos de datos tienen varianzas que son significativamente diferentes, se utiliza la prueba F, descrita más adelante en esta sección.

La estadística relevante para la prueba t de desigualdad de varianzas es

$$t = \frac{\overline{x_A} - \overline{x_B}}{\left[\frac{V(x_A)}{N_A} + \frac{V(x_B)}{N_B}\right]^{1/2}} \qquad (2.3.2.3)$$

Esta estadística se distribuye aproximadamente como la t de Student con un número de grados de libertad igual a

$$\frac{\left[\frac{V(x_A)}{N_A} + \frac{V(x_B)}{N_B}\right]^2}{\frac{\left[V(x_A)/N_A\right]^2}{N_A - 1} + \frac{\left[V(x_B)/N_B\right]^2}{N_B - 1}} \qquad (2.3.2.4)$$

La expresión (2.2.7.11.4) en general no es un entero, pero a la ecuación (2.2.7.5) no le importa.

La rutina es tutest.

Nuestro ejemplo final de la prueba t de Student es el caso de muestras pareadas. Aquí nos imaginamos que gran parte de la varianza en ambas muestras es debido a los efectos que son punto por punto idéntico en las dos muestras. Por ejemplo, podríamos tener dos candidatos que han sido valorados por cada uno de los mismos diez miembros de un comité de contratación. Queremos saber si la media de los diez puntajes difiere significativamente. En primer lugar, tratamos la prueba t de varianzas (tutest) desiguales anterior, y obtenemos un valor de prob que no es especialmente significativa (por ejemplo,> 0.05). Pero tal vez el significado está siendo lavado por la tendencia de algunos miembros del comité siempre para dar altas puntuaciones y los demás siempre para dar puntuaciones bajas, lo que aumenta la variación aparente y por lo tanto disminuye la importancia de cualquier diferencia en las medias. Así Tratamos las fórmulas de muestras pareadas,

$$C \quad (x_A, x_B) \equiv \frac{1}{N-1} \sum_{i=0}^{N-1} (x_A - \overline{x_A}) \square (x_B - \overline{x_B}) \qquad (2.3.2.5)$$

$$S_D = \left[\frac{V \quad (x_A) + V \quad (x_B) - 2 \square C \quad (x_A, x_B)}{N} \right]^{1/2} \qquad (2.3.2.6)$$

$$t = \frac{\overline{x_A} - \overline{x_B}}{S_D} \qquad (2.3.2.7)$$

donde N es el número en cada muestra (número de pares). Observe que es importante que para un determinado valor de i etiquetar los puntos correspondientes en cada muestra, es decir, los que están emparejados. Se evalúa el valor de p para el estadístico t en (2.3.2.7) para N - 1 grados de libertad.

La rutina es tptest

2.3.2.2. La Prueba F para Varianzas Significativamente Diferentes

La F-prueba, prueba la hipótesis de que dos muestras tienen diferentes varianzas por tratar de rechazar la hipótesis nula de que sus varianzas son realmente consistentes.

El estadístico F es la radio de varianza uno a la otra, por lo que los valores ya sea >> 1 o 1 << indican diferencias muy significativas. La distribución de F en el caso nulo está dada en la ecuación (2.2.7.6.5), que se evaluó usando la rutina betai. En el caso más común, estamos dispuestos a refutar la hipótesis nula (de varianzas iguales) ya sea por valores muy grandes o muy pequeños de F, por lo que el p-valor correcto es de dos colas, la suma de dos funciones beta incompletos. Resulta, por la ecuación (2.2.6.3), que las dos colas son siempre iguales; necesitamos calcular una sola, yduplicarlo.
Ocasionalmente, cuando la hipótesis nula es fuertemente viable, la identidad de las dos colas se puede confundir, dando una probabilidad indicada mayor que uno. Cambiando la probabilidad de dos menos por sí mismo, se intercambia correctamente las colas. Estas consideraciones y la ecuación (2.2.6.3) dan la rutina ftest.

2.3.3. ¿Son Dos Distribuciones Diferentes?

Dados dos conjuntos de datos, podemos generalizar las preguntas formuladas en el apartado anterior y pedimos la misma pregunta: ¿Son los dos conjuntos procedentes de la misma función de distribución, o de función de distribución diferente? De manera equivalente, en lenguaje estadístico adecuado, "¿Podemos refutar, a un cierto nivel requerido de importancia, la hipótesis nula de que dos conjuntos de datos se han extraído de la misma función de distribución de la población? Refutar la hipótesis nula de hecho demuestra que los conjuntos de datos son de diferentes distribuciones. El no poder refutar la hipótesis nula, por otro lado, sólo muestra que los conjuntos de datos pueden ser coherentes con una sola función de distribución, ya que, por ejemplo, ninguna cantidad práctica de los datos puede distinguir entre dos distribuciones que se diferencian sólo por una parte en 10 ^ 10.

Demostrando que dos distribuciones son diferentes, o que demuestren que son coherentes, es una tarea que viene todo el tiempo en muchas áreas de investigación: ¿Están las estrellas visibles distribuidas uniformemente en el cielo? (Es decir, ¿es la distribución de las estrellas función de la declinación?) ¿Son los patrones educativos en Brooklyn como en el Bronx los mismos? (Es decir, ¿son la distribución de las personas en función del último grado-asistido a la misma?) Haga dos marcas de luces fluorescentes ¿tienen la misma distribución de tiempos de combustión de salida? ¿Es la incidencia de la varicela el mismo para los primogénitos, segundos hijos, terceros hijos, etc.?

Estos cuatro ejemplos ilustran las cuatro combinaciones derivadas de dos dicotomías diferentes: 1) Los datos son ya sea continua o agrupada. 2) O bien queremos comparar

un conjunto de datos a una distribución conocida, o queremos comparar dos conjuntos de datos igualmente desconocidos. Los conjuntos de datos sobre las luces fluorescentes y en las estrellas son continuas, ya que podemos dar listas de tiempos de agotamiento individuales o de posiciones estelares. Los conjuntos de datos sobre la varicela y el nivel de estudios se han agrupado, ya que se nos da tablas de números de eventos en categorías discretas: primogénito, nacido de segunda, etc.; o sexto grado, séptimo grado, etc. Las estrellas y la varicela, por otro lado, comparten la propiedad de que la hipótesis nula es una distribución conocida (distribución del área en el cielo, o la incidencia de la varicela en la población general). Las luces fluorescentes y nivel de estudios implican la comparación de los dos conjuntos igualmente desconocidos de datos (las dos marcas, o Brooklyn y el Bronx).

Uno siempre puede convertir los datos continuos en datos continuos agrupadas, mediante la agrupación de los eventos en rangos especificados de la variable continua (s): declinación entre 0 y 10 grados, 10 y 20, 20 y 30, etc. La categorización implica una pérdida de información, sin embargo. Además, hay a menudo una considerable arbitrariedad en cuanto a cómo deben ser elegidos los contenedores. Junto con muchos otros investigadores, preferimos evitar la categorización innecesaria de datos.

La prueba aceptada por las diferencias entre las distribuciones agrupadas es la prueba de chi cuadrado. Para los datos continuos como una función de una sola variable, la prueba más generalmente aceptada es la prueba de Kolmogorov-Smirnov. Consideramos que cada uno de ellos.

2.3.4.1. La Prueba Chi – Cuadrado

Supongamos que N_i es el número de eventos observados en el i-ésimo compartimiento, y que n_i es el número esperado de acuerdo con alguna distribución conocida. Tenga en cuenta que los N_i son números enteros, mientras que los de n_i pueden no ser. Entonces el estadístico chi-cuadrado es

$$\chi^2 = \sum_i \frac{(N_i - n_i)^2}{n_i} \qquad (2.3.4.1.1)$$

donde la suma es sobre todos los contenedores. Un gran valor de chi-cuadrado indica que es poco probable la hipótesis nula (que el N_i de se han extraído de la población representada por los n_i).

Cualquier término j en (2.3.4.1.1) con 0 = n_j = N_j debe omitirse de la suma. Un término

con n_j = 0, $N_j \neq 0$ da una chi-cuadrado infinito, como debe ser, ya que en este caso el de N_i no puede extraerse de los n_i!

La función de probabilidad chi-cuadrado, $Q\ (\chi^2|\nu)$ es una función gamma incompleta, y ya se discutió en §2.2.7.4 (véase la ecuación (2.2.7.4.3)). En sentido estricto, $Q\ (\chi^2|\nu)$ es la probabilidad de que la suma de los cuadrados de ν variables normales aleatorias de varianza la unidad (y media cero) será mayor que la de chi-cuadrado. Los términos de la suma (2.3.4.1.1) no son exactamente los cuadrados de una variable normal. Sin embargo, si el número de eventos en cada compartimiento es grande (>> 1), entonces la distribución normal se alcanza, aproximadamente, y la función de probabilidad de chi-cuadrado es una buena aproximación a la distribución de (2.3.4.1.1) en el caso de la hipótesis nula. Su uso para estimar la significación de p-valor de la prueba de chi-cuadrado es estándar (pero véase (2.3.4.1.2))

El valor apropiado de ν, el número de grados de libertad, tiene cierta discusión adicional. Si los datos se recogen con el modelo de n_i fijo - es decir, no más tarde renormalizado para encajar el número total observado de eventos $\sum N_i$ entonces ν es igual al número de contenedores es N_B (Tenga en cuenta que este no es el número total de eventos!) Mucho más comúnmente, los de n_i se normalizan después del hecho de modo que su suma es igual a la suma de la de N_i. En este caso, el valor correcto para ν es $N_B - 1$, y el modelo se dice que tiene en la restricción (knstrn = 1 en el programa a continuación). Si el modelo que da los n_i tiene parámetros libres adicionales que se ajustaron después de que el hecho de llegar a un acuerdo con los datos, a continuación, cada uno de estos parámetros adicionales "ajustados" disminuye ν (y aumenta knstrn) por una unidad adicional.

Seguidamente consideramos el caso de la comparación de dos conjuntos de datos agrupados. Deje R_i sea el número de eventos en el intervalo i para el conjunto de datos y S_i el número de eventos en el mismo intervalo i para el segundo conjunto de datos. Entonces el estadístico chi-cuadrado es

$$\chi^2 = \sum_i \frac{(R_i - S_i)^2}{R_i + S_i} \qquad (2.3.4.1.2)$$

Comparando (2.3.4.1.1) a (2.3.4.1.2), debe tener en cuenta que el denominador de (2.3.4.1.2) no es más que el promedio de R_i y S_i (lo que sería un estimador de n_i en (2.3.4.1.1)). Más bien, es el doble de la media, la suma. La razón es que cada término de una suma de chi-cuadrado se supone aproximar el cuadrado de una cantidad

distribuida normalmente con la unidad de la varianza. La varianza de la diferencia de dos cantidades normales es la suma de sus varianzas individuales, no la media.

Si se recogieron los datos de una manera tal que la suma de la R_i de es necesariamente igual a la suma de S_i es así, entonces el número de grados de libertad es igual a uno menos que el número de contenedores, NB - 1 (es decir, knstrn = 1), el caso habitual. Si este requerimiento estaba ausentes, entonces el número de grados de libertad sería NB. Ejemplo: Un observador de aves quiere saber si la distribución de las aves con visión en función de las especies es lo mismo este año como el pasado. Cada compartimiento corresponde a una especie. Si el observador de pájaros toma sus datos para ser los primeros 1.000 aves que vio en cada año, entonces el número de grados de libertad es NB - 1. Si él toma sus datos sean todas las aves que vio en una muestra aleatoria de día, los mismos días de cada año en cada año, entonces el número de grados de libertad es NB (knstrn = 0). En este último caso, tenga en cuenta que él también está probando si las aves eran más numerosas en general en un año u otro: Ese es el grado de libertad adicional (es decir, aumentar la knstrn a valores más positivos) de acuerdo con su número.

El programa es chstwo.

2.3.3.2. La Prueba de Kolmogorov - Smirnov

La prueba de Kolmogorov - Smirnov (o K-S) es aplicable a distribuciones sin agrupamiento, que son funciones de una sola variable independiente, es decir, a los conjuntos de datos en la que cada punto de datos se asocia con un número único (vida útil de cada bombilla cuando se apague, o declinación de cada estrella). En tales casos, la lista de puntos de datos se puede convertir fácilmente a un estimador insesgado $S_N(x)$ de la función de distribución acumulada de la distribución de probabilidad de que se haya extraído: Si los N eventos se encuentran en los valores x_i, i = 0,..., N-1, entonces, $S_N(x)$ es la función que da la fracción de puntos de datos a la izquierda de un valor dado x. Esta función es obviamente constante entre consecutiva (es decir, ordenados en orden ascendente) de x_i y salta por la misma constante de 1 / N en cada x_i. (Vea la figura 13).

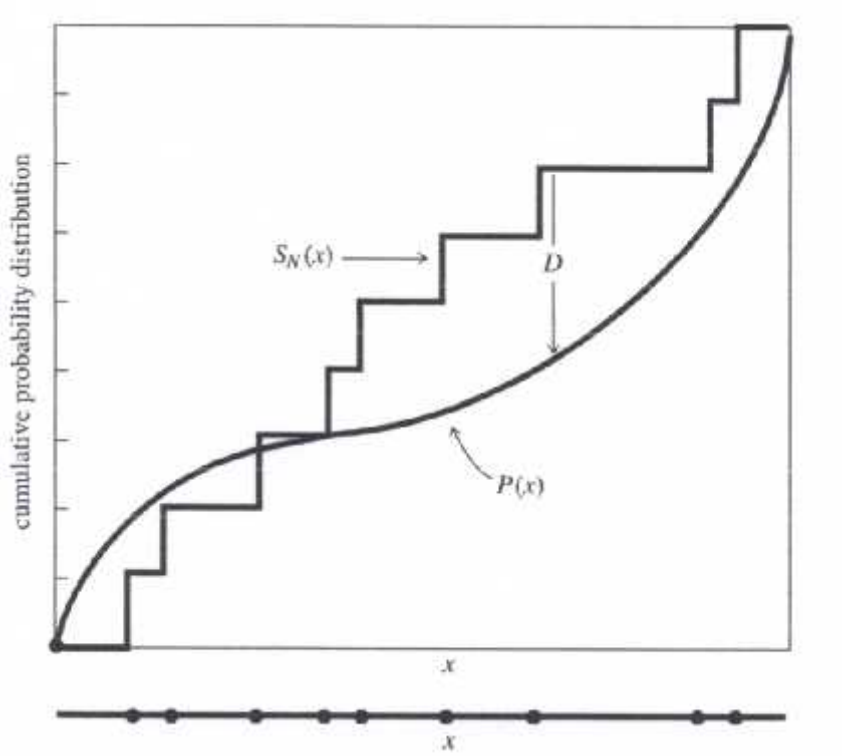


Figura 13. Kolmogorov - Smirnov estadística D. Una distribución de los valores medidos en x (que se muestra como N puntos en la abscisa inferior) es para ser comparado con una distribución teórica cuya probabilidad de distribución acumulativa se representa como P (x). Un Función - paso SN(x) de probabilidad acumulada se construye, que se eleva una cantidad igual en cada punto medido. D es la distancia más grande entre las dos distribuciones acumulativas. [15]

Diferentes funciones de distribución, o conjuntos de datos, dan diferentes estimaciones de la función de distribución acumulada por el procedimiento anterior. Sin embargo, todas las funciones de distribución acumulativa de acuerdo en el valor permisible más pequeño de x (donde son cero) y a los mayores valores permisibles de x (donde se encuentran la unidad). (Los valores máximos y mínimos pueden por supuesto ser $\pm\infty$) por lo que es el comportamiento entre los valores mayores y menores que distinguen a distribuciones.

Uno puede pensar en cualquier número de estadística para medir la diferencia global entre dos funciones distribuciones acumulativas: el valor absoluto de la zona entre ellos, por ejemplo, o su diferencia media cuadrática integrada. La prueba de Kolmogorov - Smirnov D es una medida particularmente simple: Se define como el valor máximo de la diferencia absoluta entre dos funciones de distribución acumulada. Por lo tanto, para comparar de un conjunto de datos SN(x) a un P(x) de distribución acumulativa conocida, la estadística K-S es

$$D = \max_{-\infty < x < \infty} |S_N(x) - P(x)| \qquad (2.3.3.2.1)$$

Mientras que para la comparación de dos diferentes funciones de distribución acumulativas SN1(X) y SN2(x), la estadística K-S es

$$D = \max_{-\infty < x < \infty} \left| S_{N_1}(x) - S_{N_2}(x) \right| \qquad (2.3.3.2.2)$$

Lo que hace que la estadística K-S útil es que su distribución en el caso de la hipótesis nula (conjuntos de datos extraídos de la misma distribución) se puede calcular, por lo menos a una aproximación útil, dando así la significación p-valor de cualquier valor distinto de cero observó de D. Una característica central de la prueba de K-S es que es invariante bajo reparametrización de x; en otras palabras, se puede deslizar de forma local o estirar el x - eje en la figura 14.3.1, y la distancia D máximo se mantiene sin cambios. Por ejemplo, recibirá el mismo significado utilizando x como el uso de log x.

La función que entra en el cálculo del valor de p fue discutido previamente en el § 2.2, fue definido en las ecuaciones (2.2.7.8.1) y (2.2.7.8.2), y se implementó en el objeto KSdist. En términos de los QKS función, el valor p de un valor observado de D (como una refutación de la hipótesis nula de que las distribuciones son iguales) es dado aproximadamente en §2.2.7.8 por la fórmula

$$P \quad (D > o \quad) = Q_K \left(\left[\sqrt{N_e} + 0.12 + 0.11/\sqrt{N_e} \right] \square D \right) \qquad (2.3.3.2.3)$$

Donde N_e es el número efectivo de puntos de datos, Ne = N para el caso (2.3.3.2.1) de una distribución, y

$$N_e = \frac{N_1 \square N_2}{N_1 + N_2} \qquad (2.3.3.2.3)$$

Para el caso (2.3.4.2.2) de dos distribuciones, donde N1 es el número de puntos de datos en la primera distribución y N2 el número en el segundo.

La naturaleza de la aproximación que participan en (2.3.3.2.3) es que se vuelve asintótica precisamente cuando el N_e llega a ser grande, pero ya es bastante bueno para $N_e \geq 4$, el menor número que podría utilizarse nunca en realidad.

Aquí está la rutina para el caso de una distribución: ksone

Mientras que la estadística K-S está destinada para su uso con una distribución continua, también puede ser utilizada para una distribución discreta. En este caso, se puede demostrar que la prueba es conservadora, es decir, la estadística devuelta no es más grande que en el caso continuo. Si usted permite que las variables discretas en el caso de las dos distribuciones, usted tiene que considerar cómo tratar los vínculos. La manera estándar para manejar los vínculos es la combinación de todos los puntos de datos vinculados y añadirlos a la cdf a la vez. Este refinamiento se incluye en la rutina kstwo.

2.3.4. Análisis por Tabla de Contingencia de Dos Distribuciones

En esta sección y las siguientes tres secciones, nos ocupamos de las medidas de asociación para dos distribuciones. La situación es la siguiente: Cada punto de datos tiene dos o más cantidades diferentes asociados a ella, y queremos saber si el conocimiento de una cantidad nos da ninguna ventaja demostrable para predecir el valor de otra cantidad. En muchos casos, una variable muy bien ser una variable "independiente" o "control", y otro será una variable "dependiente" o "medida". Entonces, queremos saber si esta última variable está en dependencia o asociada a la variable anterior hecho. Si es así, queremos tener alguna medida cuantitativa de la fuerza de la asociación. A menudo se oye este vagamente declaró que la cuestión de si dos variables están correlacionadas o no correlacionadas, pero vamos a reservar dichos términos para un tipo particular de asociación (lineal, o por lo menos monótona), como se comenta en §2.3.6 y §2.3.7

Tenga en cuenta que, como en las secciones anteriores, los diferentes conceptos de significación y fuerza aparecen: La asociación entre dos distribuciones puede ser muy importante, incluso si eso que la asociación es débil - si la cantidad de datos es lo suficientemente grande.

Es útil distinguir entre algunos diferentes tipos de variables, con diferentes categorías formando una jerarquía suelta.

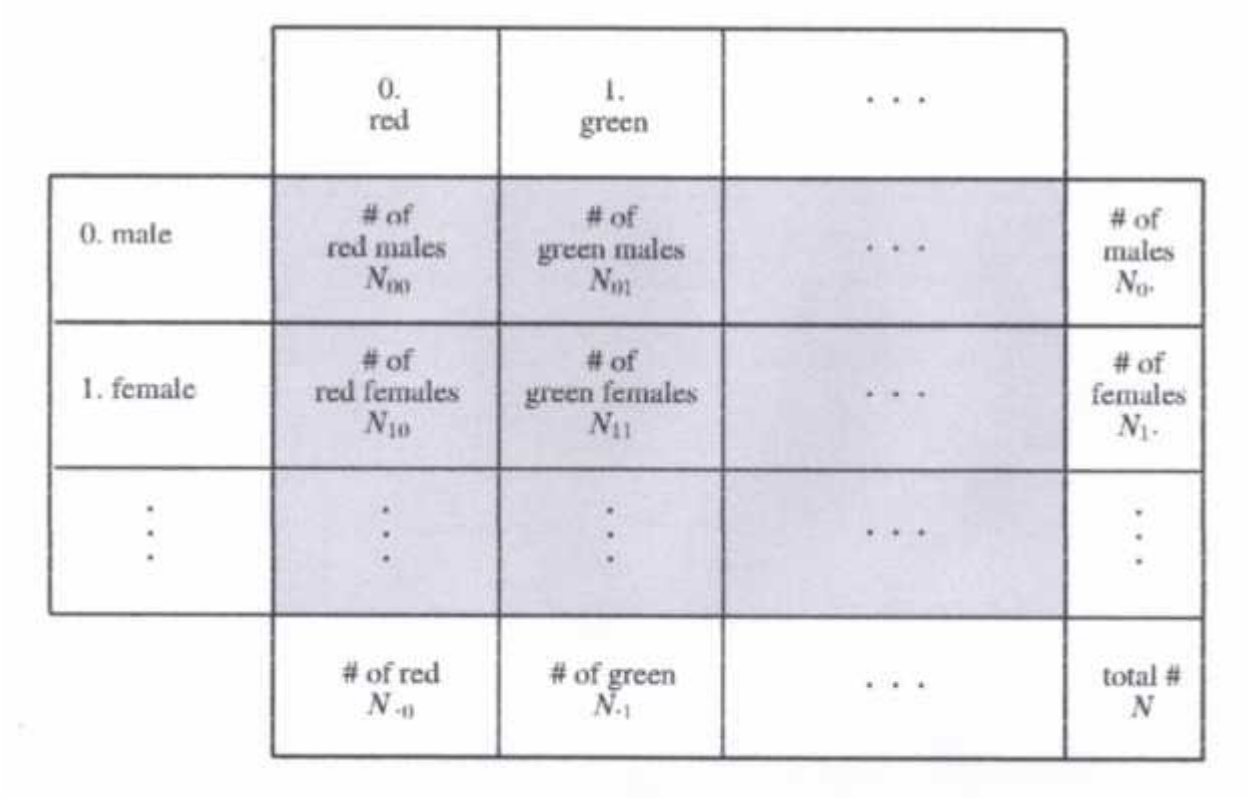
Una variable se llama nominal si sus valores son los miembros de un conjunto no ordenado. Por ejemplo, "estado de residencia" es una variable nominal que (en los EE.UU.) adquiere uno de los 50 valores; en astrofísica, "tipo de galaxia" es una variable nominal con los tres valores "espiral", "elíptica" e "irregular".
Una variable se denomina ordinal cuando sus valores son los miembros de un conjunto discreto pero ordenado. Ejemplos son, de grado en la escuela, orden planetario del Sol (Mercurio = 1, Venus = 2,...), y el orden de las crías. No tiene por qué haber ningún

concepto de "distancia métrica igual" entre los valores de una variable ordinal, sólo que ellos deben ser ordenados intrínsecamente.

Vamos a llamar a una variable continua si sus valores son números reales, como lo son los tiempos, distancias, temperaturas, etc. (Los científicos sociales a veces distinguen entre variables continuas de intervalo y de razón, pero no encontramos esa distinción muy convincente.)

Una variable continua siempre se puede hacer en un ordinal uno por se va a agrupar en rangos. Si optamos por ignorar la orden de los contenedores, entonces podemos convertirlo en una variable nominal. Las variables nominales constituyen el tipo más bajo de la jerarquía, y por lo tanto el más general. Por ejemplo, un conjunto de varias variables continuas u ordinales se puede convertir, si crudamente, en una sola variable nominal, por grueso se va a agrupar cada variable y después de tomar cada combinación distinta de agrupar como un único valor nominal. Cuando los datos multidimensionales son escasos, esto es a menudo la única forma sensata de proceder.

El resto de esta sección se ocupará de las medidas de asociación entre la variable nominal. Para cualquier par de variables nominales, los datos se pueden mostrar como una tabla de contingencia, una tabla cuyas filas están etiquetadas por los valores de una variable nominal, cuyas columnas son etiquetados por los valores de la otra variable nominal, y cuyas entradas son números enteros no negativos dando el número de eventos observados para cada combinación de fila y columna (véase la figura 14.) El análisis de la asociación entre las variables nominales se llama así el análisis de tablas de contingencia o de análisis de tabla de contingencia.



	0. red	1. green	. . .	
0. male	# of red males N_{00}	# of green males N_{01}	. . .	# of males $N_{0\cdot}$
1. female	# of red females N_{10}	# of green females N_{11}	. . .	# of females $N_{1\cdot}$
. . .	. . .	. . .	. . .	. . .
	# of red $N_{\cdot 0}$	# of green $N_{\cdot 1}$	. . .	total # N

Figura 14. Ejemplo de una tabla de contingencia de dos variables nominales, aquí el sexo y color. Se muestran la fila y la columna marginales (totales). Las variables son "nominales", es decir, el orden en que se enumeran sus valores es arbitraria y no afecta el resultado del análisis de tablas de contingencia. [15]

El resto de esta sección se ofrece un enfoque, basado en el estadístico chi-cuadrado, que hace un buen trabajo de caracterización de la importancia de la asociación, pero es sólo regular como una medida de la fuerza (principalmente porque sus valores numéricos no tienen interpretaciones muy directas)

2.3.4.1. Medidas de Asociación Basadas en Chi-Cuadrado

Algo de notación primero: Vamos N_{ij} denota el número de eventos que se producen con la primera variable x tomando su valor i-ésimo y la segunda variable y teniendo en su valor j-ésimo. Sea N el número total de eventos, la suma de todos los N_{ij} . Deje $N_{i\circ}$ denota el número de eventos para los cuales la primera variable x toma su valor i-ésimo independientemente del valor de y; $N_{\circ j}$ $N_{\circ j}$ es el número de eventos con el valor j-ésimo de y con independencia de x. Así que tenemos

$$N_{i\circ} = \sum_j N_{ij} \qquad N_{\circ j} = \sum_i N_{ij} \tag{2.3.4.1}$$

$N = \sum_i N_{i\circ} = \sum_j N_{\circ j}$

En otras palabras, "punto" es un marcador de posición que quiere decir, "suma sobre el índice que falta". $N_{\circ j}$ y $N_{i\circ}$ a veces se llaman los totales o marginales de fila y columna, pero vamos a utilizar estos términos con cautela ya que nunca podemos seguir recto que son las filas y que son las columnas!

La hipótesis nula es que las dos variables x e y no tienen asociación. En este caso, la probabilidad de que un determinado valor de x le da un valor particular de y debe ser la misma que la probabilidad de que el valor de x independientemente de y. Por lo tanto, en la hipótesis nula, el número esperado para cualquier N_i , que vamos a denotar n_i , puede calcularse a partir de sólo los totales de fila y columna,

$\frac{n_i}{N_{\circ j}} = \frac{N_{i\circ}}{N}$ que implica $n_i = \frac{N_{i\circ} N_{\circ j}}{N}$ (2.3.4.2)

Observe que si una columna o fila total es cero, entonces el número esperado para todas las entradas en esa columna o fila es también cero; en ese caso, el contenedor nunca ocurre de x o y simplemente se debe quitar del análisis.

El estadístico chi-cuadrado ahora está dada por la ecuación (2.3.6.3), que, en el presente caso, se suma sobre todas las entradas de la tabla:

$\chi^2 = \sum_{i,j} \frac{(N_i - n_i)^2}{n_i}$ (2.3.4.3)

El número de grados de libertad es igual al número de entradas en la tabla (producto del tamaño de la fila y del tamaño de la columna) menos el número de restricciones que han surgido de nuestro uso de los propios datos para determinar la n_i . Cada total de total de la columna y fila es una restricción, excepto que esta más de conteos por uno, ya que el total de los totales de las columnas y el total de los totales de las filas es igual N, el número total de puntos de datos. Por lo tanto, si la tabla es de tamaño I por J, el número de grados de libertad es I*J - I - J - 1.

La ecuación (2.3.4.3), junto con la función de probabilidad chi-cuadrado (§2.2.7.4), ahora la importancia de una asociación entre las variables x e y. Por cierto, las dos muestra – de la prueba de chi-cuadrado para la igualdad de las distribuciones, la ecuación (2.3.2.1.2), es un caso especial de la ecuación (2.3.4.3) con J = 2 y con la variable y

simplemente una etiqueta para distinguir las dos muestras.

Supongamos que hay una asociación significativa. ¿Cómo podemos cuantificar su fuerza, de modo que (por ejemplo) se puede comparar la fuerza de una asociación con el otro? La idea aquí es encontrar alguna reparametrización de chi-cuadrado que se asigna en un intervalo conveniente, como de 0 a 1, donde el resultado no depende de la cantidad de datos que nos ha tocado a la muestra, sino que depende únicamente de la población subyacente de la que se extrajeron los datos. Hay varias maneras diferentes de hacer esto. Dos de los más comunes son los llamados V de Cramer y el coeficiente de contingencia C.

La fórmula para V de Cramer es

$$V = \sqrt{\frac{\chi^2}{N \cdot m \quad (I-1, J-1)}} \qquad (2.3.4.4)$$

Donde I y J son de nuevo los números de filas y columnas, y N es el número total de eventos. V de Cramer tiene la propiedad agradable que se encuentra entre cero y uno inclusive, es igual a cero cuando no existe una asociación, y es igual a uno solo cuando la asociación es perfecta: Todos los eventos de cualquier posición fila de una columna única, y viceversa. (En el lenguaje de ajedrez, no hay dos torres, colocadas sobre una entrada de la tabla que no sea cero, se pueden capturar entre sí.)

En el caso de I = J = 2, V de Cramer también se conoce como la estadística phi.
El coeficiente de contingencia C se define como

$$C = \sqrt{\frac{\chi^2}{\chi^2 + N}} \qquad (2.3.4.5)$$

También se encuentra entre cero y uno, pero (como se desprende de la fórmula) nunca puede alcanzar el límite superior. Mientras que puede ser utilizado para comparar la fuerza de asociación de dos tablas con el mismo I y J, su límite superior depende de I y J. Por lo tanto, nunca puede ser utilizado para comparar las tablas de diferentes tamaños.

El problema con ambos V de Cramer y el coeficiente de contingencia C es que, cuando se toman valores entre sus extremos, no hay una interpretación muy directa de lo que significa ese valor.

2.3.4.2 Correlación Lineal

Pasamos ahora a las medidas de asociación entre las variables que son ordinales o continuas, en lugar de nominal. Más ampliamente usado es el coeficiente de correlación lineal. Para pares de cantidades (x_i, y_i), i = 0,…, N - 1, el coeficiente de correlación lineal r (también llamado el coeficiente de correlación producto-momento, o r de Pearson) viene dado por la fórmula

$$r = \frac{\sum_i (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2} \cdot \sqrt{\sum_i (y_i - \bar{y})^2}} \qquad (2.3.4.2.1)$$

Donde, como de costumbre, $\bar{x}$ es la media de la x_i y $\bar{y}$ es la media de la y_i.

El valor de r está entre -1 y 1, inclusive. Se toma un valor de 1, denominado "correlación positiva completa", cuando los puntos de datos se encuentran en una línea recta perfecta con pendiente positiva, con x e y aumentar juntos. El valor 1 mantiene independiente de la de la magnitud de la pendiente. Si los puntos de datos se encuentran en una línea recta perfecta con pendiente negativa, y disminuyendo a medida que x aumenta, entonces r tiene el valor -1; esto se llama "correlación negativa completa." Un valor de r cerca de cero indica que las variables x e y no están correlacionados.

Cuando una correlación es conocida por ser significativa, r es una manera convencional de resumir su fuerza. De hecho, el valor de r se puede traducir en una declaración sobre lo residuales (desviaciones de la raíz cuadrada media) son de esperar si los datos se ajustan a una recta por el método de mínimos cuadrados. Desafortunadamente, r es un bastante pobre estadística para decidir si una correlación observada es estadísticamente significativa y / o si uno observado correlación es significativamente más fuerte que otro. La razón es que r es ignorante de las distribuciones individuales de x e y, así que no hay manera universal a calcular su distribución en el caso de la hipótesis nula.

Acerca de la única declaración general que puede hacerse es la siguiente: Si la hipótesis nula es que x e y no están correlacionados, y si las distribuciones para x e y cada uno tienen suficientes momentos convergentes ("colas" morir con la suficiente rapidez), y si N es grande (típicamente > 500), entonces r se distribuye aproximadamente normal, con una media de cero y una desviación estándar de 1 / sqrt (N). En ese caso, el significado (a doble cara) de la correlación, es decir, la probabilidad de que | r | debe ser mayor que su valor observado en la hipótesis nula, es

$$e \quad \left(\frac{|r| \cdot \sqrt{N}}{\sqrt{2}}\right) \qquad (2.3.4.2.2)$$

Donde erfc(x) es la función de error complementaria, la ecuación (2.2.5.1), calculado por las rutinas de Erf. Erfc o erfc de §2.2.5. Un pequeño valor de (2.3.4.2.2) indica que las dos distribuciones son significativamente correlacionadas. (Ver expresión (2.3.4.2.9) abajo para una prueba más precisa.)

La mayoría de los libros de estadística tratan de ir más allá de (2.3.4.2.2) y dar pruebas estadísticas adicionales que se pueden hacer usando r. En casi todos los casos, sin embargo, estas pruebas son válidos sólo para una clase muy especial de hipótesis, a saber, que la distribución de x e y forman conjuntamente un binormal o distribución gaussiana bidimensional alrededor de sus valores medios, con densidad de probabilidad conjunta

$$p(x,y)d \quad = c \quad .\ e \quad \left(-\frac{1}{2}\ (a_0\ x^2 - 2a_0\ x \ + a_1\ y^2)\right)d \qquad (2.3.4.2.3)$$

Donde a_0 , a_0 y a_1 son constantes arbitrarias. Para esta distribución r tiene el valor:

$$r = -\frac{a_0}{\sqrt{a_0\ a_1}} \qquad (2.3.4.2.4)$$

Hay ocasiones en que (2.3.4.2.3) pueden ser conocidos por ser un buen modelo de los datos. Puede haber otras ocasiones en las que estamos dispuestos a tomar (2.3.4.2.3) como al menos una conjetura áspera y listo, ya que muchas distribuciones bidimensionales hacen parecerse a una distribución binormal, (es decir, una gaussiana bidimensional) en menos no demasiado lejos en sus colas. En cualquier situación, podemos utilizar (2.3.4.2.3) para ir más allá de (2.3.4.2.2) en cualquiera de varias direcciones:

En primer lugar, nos podemos permitir la posibilidad de que el número N de puntos de datos no es muy grande. En este caso, resulta que la estadística

$$t = r\ \sqrt{\frac{N-2}{1-r^2}} \qquad (2.3.4.2.5)$$

se distribuye en el caso nulo (de correlación) como la distribución t de Student con ν = N - 2 grados de libertad, cuyas nivel de significación bilateral está dada por 1 - A (t | v) (ecuación (2.2.7.5)). Como N llega a ser grande, este significado y (2.3.4.2.2) se convierten en asintóticamente la misma, de modo que uno nunca lo hace peor por el uso de (2.3.4.2.5), incluso si el supuesto binormal no está bien fundamentada.

En segundo lugar, cuando N es sólo moderadamente grande (> = 10), podemos comparar si la diferencia de dos de r significativamente distinto de cero, por ejemplo, de diferentes experimentos, es en sí mismo significativo. En otras palabras, podemos cuantificar si un cambio en alguna variable de control altera significativamente la correlación existente entre dos variables. Esto se hace mediante el uso de la z-transformación de Fisher para asociar cada r medido con una correspondiente z:

$$z = \frac{1}{2} ln \left(\frac{1+r}{1-r}\right) \qquad (2.3.4.2.6)$$

Luego, cada z es una distribución aproximadamente normal con un valor medio

$$\bar{z} = \frac{1}{2}\left[ln \left(\frac{1+r_{ti}}{1-r_{ti}}\right) + \frac{r_{ti}}{N-1}\right] \qquad (2.3.4.2.7)$$

Donde r_true es el valor real o de la población del coeficiente de correlación, y con una desviación estándar

$$\sigma(z) \approx \frac{1}{\sqrt{N-3}} \qquad (2.3.4.2.8)$$

Las ecuaciones (2.3.4.2.7) y (2.3.4.2.8), cuando son válidos, dan varias pruebas estadísticas útiles. Por ejemplo, el nivel de significación en la cual un valor medido de r difiere de algunos r_true valor hipotetizado está dada por

$$e \quad \left(\frac{|z-\bar{z}|\sqrt{N-3}}{\sqrt{2}}\right) \qquad (2.3.4.2.9)$$

Donde z y $\bar{z}$están dados por (2.3.7.1.6) y (2.3.7.1.7), con valores pequeños de (2.3.7.1.9) que indica una diferencia significativa. (Configuración $\bar{z}$ = 0 hace expresión (2.3.7.1.9) un reemplazo más precisa para la expresión (2.3.7.1.2) anteriormente). Del mismo modo, la importancia de una diferencia entre dos correlación medido r1 y r2 es

$$e \quad \left(\frac{|z_1-z_2|}{\sqrt{2}\sqrt{\frac{1}{N_1-3}+\frac{1}{N_2-3}}}\right) \qquad (2.3.4.2.10)$$

Donde z1 y z2 se obtienen de r1 y r2 usando (2.3.4.2.6), y donde N1 y N2 son, respectivamente, el número de puntos de datos en la medición de r1y r2.

Todos los significados anteriores son de dos caras. Si desea refutar la hipótesis nula en

favor de una hipótesis unilateral, tales r1> r2 (donde el sentido de la desigualdad se decidió a priori), entonces (i) si sus r1 medidos y r2 tienen el sentido equivocado, no han podido demostrar su hipótesis de una cola, pero (ii) si tienen el orden correcto, puede multiplicar los significados dados anteriormente por 0,5, lo que las hace más significativo.

Pero tenga en cuenta: Estas interpretaciones de la estadística r puede ser completamente de sentido si la distribución de probabilidad conjunta de sus variables x e y es muy diferente de una distribución binormal. La rutina es pearsn.

2.3.5 La correlación No Paramétrica o Rango

Es precisamente la incertidumbre en la interpretación de la significación del coeficiente de correlación lineal r que nos lleva a los conceptos importantes de correlación no paramétrica o rango. Al igual que antes, se nos da N pares de mediciones (x_i, y_i). Antes, las dificultades surgieron porque no sabíamos necesariamente la función de distribución de probabilidad de la que se extrajeron de la x_i o de y_i.

El concepto clave de correlación no paramétrica es la siguiente: Si reemplazamos el valor de cada x_i por el valor de su rango entre todas las otras de x_i en la muestra, es decir, 1, 2, 3,..., N, entonces la resultante lista de números se extrae de una función de distribución perfectamente conocido, a saber, de manera uniforme entre los enteros entre 1 y N, inclusive. Mejor que uniformemente, de hecho, ya que si los de x_i son todos distintos, a continuación, cada entero se producirá precisamente una vez. Si algunos de los de x_i tienen valores idénticos, es convencional para asignar a todos estos "vínculos" de la media de las filas que habrían tenido si sus valores habían sido ligeramente diferente. Este rango medio a veces será un entero, a veces un medio entero. En todos los casos, la suma de todos los rangos asignados será la misma que la suma de los enteros de 1 a N, es decir, 1/2 * N * (N + 1).

Por supuesto que sí exactamente el mismo procedimiento para el de y_i, reemplazando cada valor por su rango entre los otros yi de la muestra.

Ahora somos libres para inventar estadística para detectar la correlación entre las series uniformes de números enteros entre 1 y N, teniendo en cuenta la posibilidad de empates en las filas.
Hay, por supuesto, algo de pérdida de información en la sustitución de los números originales por filas. Podríamos construir algunos ejemplos en lugar artificiales donde una correlación podría ser detectado de forma paramétrica (por ejemplo, en el coeficiente de

correlación lineal r) pero no pudo ser detectado no paramétrica. Estos ejemplos son muy raros en la vida real, sin embargo, y la ligera pérdida de información en el ranking es un pequeño precio a pagar por una muy importante ventaja: ¡Cuando una correlación se demuestra a estar presente de forma no paramétrica, entonces es realmente allí! (Es decir, a un nivel de certeza de que depende de la significación elegido.) La correlación no paramétrica es más robusta que la correlación lineal, sin defectos no deseados, más resistentes en los datos, en el mismo tipo de sentido de que la mediana es más robusta que la media.

Como siempre en estadística, algunos elección particular de una estadística ya se han inventado para nosotros y consagrado, si no beatificado, por el uso popular. Vamos a discutir dos, el coeficiente de Spearman de correlación orden de rango (r_s) y tau de Kendall.

2.3.5.1 Coeficiente de Spearman de Correlación orden de rango

Deje R_i sea el rango de x_i entre los otros de x y S_i sea el rango de y_i entre los otros de y, con los lazos que se asigna el rango medio apropiado como se describe anteriormente. Entonces, el coeficiente de correlación de orden de rango se define como el coeficiente de correlación lineal de las filas, a saber

$$r_S = \frac{\sum_i (R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum_i (R_i - \bar{R})^2} \sqrt{\sum_i (S_i - \bar{S})^2}} \qquad (2.3.5.1.1)$$

La importancia de un valor distinto de cero de r_s es probado por la cómputo de

$$t = r_S \sqrt{\frac{N-2}{1-r_S^2}} \qquad (2.3.5.1.2)$$

Que se distribuye aproximadamente como la distribución de Student con N - 2 grados de libertad. Un punto clave es que esta aproximación no depende de la distribución original de las x y de y; es siempre la misma aproximación, y siempre es bastante bueno.

Resulta que r_s está estrechamente relacionada con otra medida convencional de la correlación no paramétrica, llamado, la suma del cuadrado de la diferencia de rangos, definida como

$$D = \sum_{i=0}^{N-1} (R_i - S_i)^2 \qquad (2.3.5.1.3)$$

(D es a veces D **, donde se utilizan los asteriscos para indicar que los lazos son tratados por midranking)

Cuando no hay empates en los datos, la relación exacta entre D y r_s es

$$r_S = 1 - \frac{6 \square D}{N^3 - N} \qquad (2.3.5.1.4)$$

Cuando existan vínculos, la relación exacta es un poco más complicada:
Que f_k sea el número de lazos en el grupo de orden k de los lazos entre el Ri y dejar g_m sea el número de lazos en el grupo m-ésimo de los lazos entre los Si. Entonces resulta que

$$r_S = \frac{1 - \frac{6}{N^3 - N} \square \left[D + \frac{1}{1} \sum_k (f_k^3 - f_k) + \frac{1}{1} \sum_m (g_m^3 - g_m) \right]}{\left[1 - \frac{\sum_k (f_k^3 - f_k)}{N^3 - N} \right]^{1/2} \square \left[1 - \frac{\sum_m (g_m^3 - g_m)}{N^3 - N} \right]^{1/2}} \qquad (2.3.5.1.5)$$

Sostiene exactamente. Tenga en cuenta que si de todos los f_k y todos los g_m son igual a uno, lo que significa que no existan vínculos, entonces la ecuación (2.3.7.1.5) se reduce a la ecuación (2.3.7.1.4)

En (2.3.7.1.2) dimos un estadístico t que pone a prueba la importancia de una r_s distintos de cero. También es posible probar el significado de D directamente. El valor esperado de D en la hipótesis nula de conjuntos de datos sin correlación es

$$\overline{D} = \frac{1}{6}(N^3 - N) - \frac{1}{1}\sum_k (f_k^3 - f_k) - \frac{1}{1}\sum_m (g_m^3 - g_m) \qquad (2.3.5.1.6)$$

Su varianza es

$$V \quad (D) = \frac{(N-1) \square N^2 \square (N+1)^2}{36} \left[\left[1 - \frac{\sum_k (f_k^3 - f_k)}{N^3 - N} \right] \right] \left[\left[1 - \frac{\sum_m (g_m^3 - g_m)}{N^3 - N} \right] \right] \qquad (2.3.5.1.7)$$

Y es una distribución aproximadamente normal, de modo que el nivel de significación es una función de error complementaria. Por supuesto, (2.3.5.1.2) y (2.3.5.1.7) no son pruebas independientes, sino simplemente variantes de la misma prueba. En el programa que sigue, se calcula tanto el nivel de significación obtenido mediante el uso de (2.3.8.1.2) y el nivel de significación obtenido mediante el uso de (2.3.5.1.7); su discrepancia le dará una idea de lo bien que las aproximaciones son. También se dará cuenta que partimos de la tarea de asignar rangos (incluyendo rangos medios atadas) en una función separada, crank.

2.3.5.2 La Correlación No Paramétrica Tau de Kendall

T de Kendall es aún más no paramétrico de rs de Spearman o D. En lugar de utilizar la diferencia numérica de filas, se utiliza sólo el orden relativo de rangos: un rango superior, inferior en rango, o en el mismo rango. ¡Pero en ese caso ni siquiera tenemos que clasificar los datos! Rangos serán más altos, más bajo, o el mismo si y sólo si los valores son más grandes, más pequeño, o igual, respectivamente. A fin de cuentas, preferimos rs como ser la prueba no paramétrica más sencillo, pero ambos somos estadística de uso general. De hecho, tau y rs están fuertemente correlacionados y, en la mayoría de las aplicaciones, son efectivamente la misma prueba.

Para definir τ, comenzamos con los puntos de datos N (xi, yi). Ahora considere todos 1/2 * N * (N-1) pares de puntos de datos, donde un punto de datos no se puede combinar con sí mismo, y donde los puntos en cualquier orden cuentan como un par. Llamamos a un par concordante si el orden relativo de las filas de los dos x (o para el caso de los dos x mismos) es el mismo que el orden relativo de las filas de los dos de y (o para el caso de los dos y es a sí mismos) . Llamamos a una pareja discordante si el orden relativo de las filas de los dos x es opuesta a la ordenación relativa de las filas de los dos y de. Si hay un empate en cualquiera de las filas de los dos x o las filas de los dos y de, entonces nosotros no llamamos a la par ya sea concordante o discordante. Si el empate es en las x, llamaremos a la par un "par y extra." Si el empate es en las y, llamaremos a la par con una "x par extra." Si el empate es en ambos las x y las y, no llamamos a la par nada en absoluto. ¿Sigues con nosotros?

Tau de Kendall es ahora la siguiente simple combinación de estos diversos aspectos:

$$\tau = \frac{c_{\cdot} \quad - d \quad co}{\sqrt{c_{\cdot} \quad + d \quad + e_{\cdot} \quad \_y} \; \sqrt{c_{\cdot} \quad + d \quad + e_{\cdot} \quad \_x}} \qquad (2.3.5.2.1)$$

Usted puede convencer fácilmente para que este debe estar entre 1 y -1, y que asume los valores extremos sólo por acuerdo de rango completo o completa reversión rango, respectivamente.

Más importante, Kendall ha funcionado, de la combinatoria, la distribución aproximada de tau en la hipótesis nula de no asociación entre x e y. En este caso, tau es una distribución aproximadamente normal, con valor cero de media y una varianza de

$$V \quad (\tau) = \frac{4N+1}{9N(N-1)} \qquad (2.3.5.2.2)$$

El siguiente programa procede de acuerdo con la descripción anterior, y por lo tanto los bucles sobre todos los pares de puntos de datos. Cuidado: Este es un algoritmo O (N ²), a diferencia del algoritmo para r_s, cuyas operaciones especie dominante son de orden NlogN. Si usted está calculando rutinariamente tau de Kendall para conjuntos de datos de más de unos pocos miles de puntos, usted puede estar adentro para algo de computación grave. Sin embargo, si usted está dispuesto a compartimentar sus datos en un número moderado de contenedores, sigue leyendo.

A veces sucede que hay sólo unos pocos valores posibles para cada x e y. En ese caso, los datos se pueden grabar como una tabla de contingencia que da el número de puntos de datos para cada contingencia de x e y.

Orden de rango coeficiente de correlación de Spearman no es estadística muy natural en estas circunstancias, ya que asigna a cada compartimiento x y y un valor midrank no muy significativa y luego asciende hasta un gran número de diferencias de rangos idénticos. Tau de Kendall, por otro lado, con su conteo simple, sigue siendo bastante natural. Además, su O (N^2) algoritmo ya no es un problema, ya que podemos organizar para que un bucle sobre pares de entradas de la tabla de contingencia (cada uno contiene muchos puntos de datos) en lugar de a través de pares de puntos de datos. Esto se implementa en el programa que sigue.

Tenga en cuenta que la tau de Kendall se puede aplicar sólo a tablas de contingencia donde ambas variables son ordinales, es decir, bien ordenado, y que busca específicamente correlaciones monótonas, no para asociaciones arbitrarias. Estas dos propiedades hacen que sea menos general que los métodos de las tablas de contingencia que se aplicaban a nominales, es decir, sin orden, las variables y las asociaciones arbitrarias.

Comparando el kendl1 anterior con kendl2 continuación, podrás ver que hemos cambiado una serie de variables de int para doubles. Esto es porque el número de eventos en una tabla de contingencia puede ser lo suficientemente grande como para provocar desbordamientos en algunos de la aritmética de enteros, mientras que el número de puntos de datos individuales en una lista no podría ser tan grande (para una rutina O(N^2) !)

Capítulo III. MARCO METODOLÓGICO O DESARROLLO DE CLASES.

En este capítulo que es el correlato del capítulo II, se desarrollarán las clases en C++ que modelan a algunos de los objetos de la Ingeniería de Transportes.

3.1 Características del Conductor, del Peatón, del Vehículo y del Camino en el Lenguaje de Programación C++.

4.1.1. Clase persona

Podemos ya ver el código en C++ de la clase persona, cuyo objetivo ha sido brindar una visión de muy alto nivel de la programación en C++ y básicamente de lo que es una clase en el contexto del lenguaje de programación C++.

En C++ todo lo que está entre /* y */ es un comentario multilínea, el compilador no lo reconoce.
Existe también lo que se denomina comentario de una línea, todo lo que está después de // en una sola línea es un comentario.

Este es el archivo de cabecera de la clase persona, es una declaración de la clase persona.

```
//---------------------------------------------------------------------------
#ifndef personaH
#define personaH
#include <string>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>


std::string fecha = __DATE__;
std::string anyoActual = fecha.substr(7, 4);
const char* s = anyoActual.c_str();
int anyoAct = atoi(s);


/** Persona es una clase que representa a un ser humano. */
class persona {
     private:
```

```
            // Campos de datos
            /** El nombre propio. */
            std::string nombre;

            /** El apellido. */
            std::string apellido;

            /** El número de identificación personal. */
            std::string DNI;

            /** El año de nacimiento. */
            int anyoNacimiento;

            // Constantes
            /** La edad en la que una persona puede votar. */
            static const int EDAD_VOTO = 18;
            static const int EDAD_SENIOR = 65;

      public:
            // Constructores
            /** Construir una persona con valores determinados.
                  @param Nombre El nombre propio
                  @param Apellido El apellido
                  @param dni El número de identificación personal
                  @param birth El año de nacimiento
            */
            persona(std::string Nombre, std::string Apellido, std::string dni,
int birth):
            nombre(Nombre), apellido(Apellido), DNI(dni), anyoNacimiento(birth) {
}

            /** Construir una persona con valores predeterminados. */
            persona(): nombre(""), apellido(""), DNI(""), anyoNacimiento(1900) {
}

            // Funciones modificadoras
            /** Establecer el campo nombre
                  @param Nombre El nombre propio o de pila.
            */
            void setNombre(std::string Nombre) {
                  nombre = Nombre;
            }

            /** Establecer el campo apellido
                  @param Apellido El apellido o nombre de la familia.
            */
            void setApellido(std::string Apellido) {
                  apellido = Apellido;
            }

            /** Establecer el campo anyoNacimiento
```

```
      @param birth El año de nacimiento
*/
void setNacimiento(int birth) {
      anyoNacimiento = birth;
}

// Funciones accesorias
/** Obtener el nombre propio de la persona
      @return El nombre propio como una cadena
*/
std::string getNombre() const {
      return nombre;
}

/** Obtener el apellido de la persona
      @return el apellido como cadena
*/
std::string getApellido() const {
      return apellido;
}

/** Obtener el número de identificación personal de la persona
      @return el número de identificación personal como cadena
*/
std::string getDNI() const {
      return DNI;
}

/** Obtener el año de nacimiento de la persona.
      @return el año de nacimiento como un valor int
*/
int getAnyoNacimiento() const {
      return anyoNacimiento;
}

// Otras funciones
/** Calcular la edad de una persona con su año de nacimiento.
      @param year El año actual
      @return el año menos el año de nacimiento
*/
int edad(int year = anyoAct) const;

/**Determinar si una persona puede votar o conducir.
      @param year El año actual
      @return true si la edad de la persona es mayor o igual a la
                  edad para votar o conducir
*/
bool puedeVotar(int year) const;

/** Determina si una persona es un adulto mayor.
      @param year El año actual
```

```
                @return true si la edad de la persona es mayor o
                            igual a la edad a la cual se consideró
                            un ciudadano adulto
        */
        bool esSenior(int year) const;

        virtual std::string to_string() const;

        /** Comparar dos objetos persona de igualdad.
                @param per El segundo objeto persona
                @return true si los objetos persona tienen el mismo número de
                        identificación personal; false de no ser así.
        */
        bool operator==(const persona& per) const;

        /** Comparar dos objetos persona para efectos de desigualdad
                @param per El segundo objeto persona
                @return la negación de los operadores iguales
        */
        bool operator!=(const persona& per) const;

        /** Declaración del operador de inserción en la cadena para persona.
                @param os El ostream objetivo
                @param per La emisión del objeto persona
                @return La cadena actualizada de salida
        */
        friend std::ostream& operator<<(std::ostream& os, const persona& per)
{
                os << "Persona: " << '\n'
                        << "Nombre de pila: " << per.nombre << '\n'
                        << "Apellidos: " << per.apellido << '\n'
                        << "Nº de DNI: " << per.DNI << '\n'
                        << "Año de nacimiento: " << per.anyoNacimiento << '\n';
        return os;
        }

};
//---------------------------------------------------------------------------
#endif
```

Y aquí se muestra el archivo de implementación de la clase persona.

```
//---------------------------------------------------------------------------

#include "persona.h"
#include <ostream>
#include <sstream>
//---------------------------------------------------------------------------
```

```
/** Calcula la edad de una persona en su cumpleaños de este año.
     @param year El año actual
     @return El año menos el año de nacimiento
*/
int persona::edad(int year) const {
     return year - anyoNacimiento;
}

/** Determina si una persona puede votar o conducir un vehículo motorizado
     @param year El año actual
     @return true si la edad de la persona es mayor o igual a la edad para votar
*/
bool persona::puedeVotar(int year) const {
    return edad(year) >= EDAD_VOTO;
}

/** Determina si una persona es un adulto mayor.
     @param year El año actual
     @return true si la edad de la persona es mayor o igual a la edad a la cual
     se consideró un ciudadano adulto mayor
*/
bool persona::esSenior(int year) const {
     return edad(year) >= EDAD_SENIOR;
}

std::string persona::to_string() const {
     std::ostringstream sb;
     sb << "Persona: " << '\n'
           << "Nombre de pila: " << nombre << '\n'
           << "Apellidos: " << apellido << '\n'
           << "Número de DNI: " << DNI << '\n'
           << "Año de nacimiento: " << anyoNacimiento << '\n';
     return sb.str();
}

/** Comparar dos objetos persona de igualdad.
     @param per El segundo objeto persona
     @return true si los objetos persona tienen el mismo número de
                 identificación personal; false de no ser así.
*/
bool persona::operator==(const persona& per) const {
     return DNI == per.DNI;
}

/** Comparar dos objetos persona para efectos de desigualdad
     @param per El segundo objeto persona
     @return la negación de los operadores iguales
*/
bool persona::operator!=(const persona& per) const {
    return !(*this == per);
```

```
}
```

3.1.2. Clase conductor

Entonces aquí está la declaración de la clase conductor, que se deriva de la clase persona.

```
//---------------------------------------------------------------------------

      #ifndef ConductorH
      #define ConductorH
      #include <cassert>
      #include "C:\Operaciones de Transito\Persona\persona.h"

//---------------------------------------------------------------------------
      /** Clase que representa a un conductor, se deriva de la clase persona
      */
      class conductor : public persona {
      private:
            /** Tiempo de Percepción - Reacción del conductor en segundos
            */
            float tiempoPercepcionReaccion;
      public:
            // Constructores
            /** Construye un conductor con campos determinados.
                  @param Nombre El nombre propio del conductor
                  @param Apellido El apellido del conductor
                  @param dni El número de identificación personal del conductor
                  @param birth El año de nacimiento del conductor
                  @param tiempoP_R El tiempo de Percepción - Reacción del
                                    conductor
             */
             conductor(std::string Nombre, std::string Apellido, std::string dni,
      int birth,
             float tiempoP_R): persona(Nombre, Apellido, dni, birth),
             tiempoPercepcionReaccion(tiempoP_R) {
                        assert(puedeVotar(anyoAct));
                  if(tiempoP_R < 1.26 )
                              tiempoPercepcionReaccion = 2.5;
                        else tiempoPercepcionReaccion = tiempoP_R;
             }

             /** Construye un conductor con campos predeterminados. */
             conductor(): persona(), tiempoPercepcionReaccion(2.5) {
                  assert(puedeVotar(anyoAct));
             }

             /**Establece el tiempo de Percepción - reacción del conductor
```

```
		@param tiempoP_R El tiempo de percepción - reacción en segundos
	 */
	void setTiempoPercReacc(float tiempoP_R);

	/** Devuelve el tiempo de percepción - reacción como un float.
		@return el tiempo de percepción - reacción en segundos
	*/
	float getTiempoP_R() {
		return tiempoPercepcionReaccion;
	}

	std::string to_string() const;
};

#endif
```

Y aquí está la implementación de la clase conductor

```
//---------------------------------------------------------------------------

	#include "Conductor.h"

//---------------------------------------------------------------------------


	/** Establece el tiempo de percepción - reacción del conductor
	@param tiempoP_R El tiempo de percepción - reacción en segundos
	*/
	void conductor::setTiempoPercReacc(float tiempoP_R) {
	if(tiempoP_R < 1.26 )
		tiempoPercepcionReaccion = 2.5;
	else tiempoPercepcionReaccion = tiempoP_R;
	}

	std::string conductor::to_string() const {
	std::ostringstream sb;
	sb << "Conductor: \n"
		<< persona::to_string()
		<< "Tiempo de Percepción - Reacción: " << tiempoPercepcionReaccion <<
	" segundos\n";
	return sb.str();
}
```

3.1.3. Clase peatón

Con estas consideraciones se tiene la clase peatón que se deriva también de la clase persona:

```
//---------------------------------------------------------------------------
```

```
#ifndef PeatonH
#define PeatonH
#include <iostream>
#include <string>
#include <iomanip>
#include "C:\Operaciones de Transito\Persona\persona.h"

//---------------------------------------------------------------------------
/** clase que implementa las características y funcionalidades de un peatón,
      se deriva de la clase persona.
 */

class peaton : public persona {
      private:
            float velocidad;                  //pies/segundo
            //constantes
            /** Velocidad mínima de desplazamiento de un discapacitado */
            static const float VelocMinimaPeaton;
            /** Velocidad máxima de desplazamiento de un peatón*/
            static const float VelocidadMaximaPeaton;


      public:
            // Constructores
            /** Construye un peatón con valores determinados
                  @param Nomb El nombre de pila del peatón
                  @param Ape El apellido del peatón
                  @param dni El número de identificación personal del peatón
                  @param anyoNac El año de nacimiento del peatón.
                  @param Veloc La velocidad del peatón en pies/segundo
            */
            peaton(std::string Nomb, std::string Ape, std::string dni, int anyoNac,
            float Veloc = 4.0): persona(Nomb, Ape, dni, anyoNac), velocidad(Veloc) {}

            peaton(): persona(), velocidad(4.0) {}

            float getVelocidad() const {
                  return velocidad;
        }
            void setVelocidad(float Veloc);

            std::string to_string() const;

            friend std::ostream& operator<<(std::ostream& os, const peaton& Peaton) {
                  os << "Peatón: " << Peaton.getNombre() << '\n';
                  os << "Nombre de pila: " << Peaton.getNombre() << '\n';
                  os << "Apellidos: " << Peaton.getApellido() << '\n';
                  os << "Nº de DNI: " << Peaton.getDNI() << '\n';
                  os << "Año de Nacimiento: " << Peaton.getAnyoNacimiento() << '\n';
                  os << std::setprecision(2);
```

```
                  os   <<   "Velocidad:   "   <<   std::fixed   <<   std::setw(5)   <<
Peaton.velocidad << '\n';
                  return os;
            }
};

#endif
```

Y aquí está su implementación:

```
//---------------------------------------------------------------------------

#include "Peaton.h"
//---------------------------------------------------------------------------

/** Archivo de implementación de la clase peatón*/

/** Establece la velocidad del peatón
     @param Veloc La velocidad del peatón en pies/segundo
*/
void peaton::setVelocidad(float Veloc){
     if(Veloc < VelocMinimaPeaton || Veloc > VelocidadMaximaPeaton)
           velocidad = 4.0;          //Se asume la velocidad de diseño de la
     else velocidad = Veloc;
}

std::string peaton::to_string() const {
     std::ostringstream sb;
     sb << "Peaton : \n"
           << persona::to_string()
           << "Velocidad del peatón: " << velocidad << " pies/seg\n";
     return sb.str();
}

const float VelocMinimaPeaton = 1.97;
const float VelocidadMaximaPeaton = 8.0;
```

3.1.4. Clase vehículo

Con todo el tratamiento teórico del capítulo II, mostramos enseguida la clase vehículo, hacemos notar que algunos conceptos como el radio de giro o la resistencia a la pendiente, necesitan de la clase camino, ya que la pendiente y la sobreelevación son características del camino.

```
//---------------------------------------------------------------------------

#ifndef VehículoH
#define VehículoH
```

```
#include <string>
#include <sstream>
#include <math.h>
#include "C:\Operaciones de Transito\...\Camino\Camino.h"
#include "C:\Operaciones de Transito\...\Conductor\Conductor.h"
//El peso específico del aire (0.0766 libras/pie³) al nivel del mar,
const float rho = 0.0766;

//C_D Coeficiente de arrastre aerodinámico 0.4 para automóviles
const float C_D = 0.4;

//C_rs constante para hallar la fuerza de resistencia al rodamiento, para
automóviles 0.012
const float C_rs = 0.012;

//C_rv constante para hallar la fuerza de resistencia al rodamiento, para
automóviles 0.65e-6 segundo²/pie²
const float C_rv = 0.65e-6;

/** Clase que implementa un vehículo
      @param Marca La marca del vehículo
      @param Modelo El modelo del vehículo
      @param year El año de fabricación del vehículo
      @param Capacidad El número de personas que pueden viajar, incluye al conductor
      @param Placa La identificación única del vehículo
      @param Peso El peso bruto del vehículo en libras
      @param Potencia La potencia máxima desarrollada por el vehículo en Horse
            Power
      @param Largo El largo del vehículo en pies
      @param Ancho El ancho del vehículo en pies
      @param Altura La altura del vehículo en pies
      @param Velocidad La velocidad del vehículo en millas/hora
      @param Desaceleracion, la tasa de desaceleración en un frenado en
            pies/segundo²
      @param NumEjes El número de ejes del vehículo, normalmente 2.
      @param NumLlantas El número de llantas del vehículo

*/
class vehiculo {
      private:
            std::string marca;
            std::string modelo;
            int anyo;
            int capacidad;
            std::string placa;
            float peso;
            float potencia;
            float largo;
            float ancho;
            float altura;
            float velocidad;
            float desaceleracion;
            int numEjes;
            int numLlantas;

      public:
            //Constructores
            /** Construir un vehículo con los datos determinados

            */
            vehiculo(std::string Marca, std::string Modelo, int year,
                  int Capacidad, std::string Placa, float Peso, float Potencia,
                  float Largo, float Ancho,float Altura, float Velocidad, float
                  Desaceleracion, int NumEjes = 2,int NumLlantas = 4) :
                  marca(Marca), modelo(Modelo), anyo(year),capacidad(Capacidad),
```

```
            placa(Placa),    peso(Peso),    potencia(Potencia),largo(Largo),
            ancho(Ancho), altura(Altura), velocidad(Velocidad),
            desaceleracion(Desaceleracion),numEjes(NumEjes),
            numLlantas(NumLlantas) {};

        vehiculo() : marca(""), modelo(""), anyo(1990), capacidad(5),
                     placa(""),  peso(4000.0),  potencia(80.0),  largo(19.0),
                     ancho(7.0), altura(4.25),desaceleracion(11.2),
                     velocidad(30.0), numEjes(2), numLlantas(4) {};


        virtual float aceleracion(float Velocidad);
        float areaFrontal() const;

        //Funciones set
        void setMarca(std::string Marca);
        void setModelo(std::string Modelo);
        void setAnyo(int year);
        void setCapacidad(int Capacidad);
        void setPlaca(std::string Placa);
        void setPeso(float Peso);
        void setPotencia(float Potencia);
        void setLargo(float Largo);
        void setAncho(float Ancho);
        void setAltura(float Altura);
        void setVelocidad(float Velocidad);
        void setDesaceleracion(float Desacel);
        void setNumEjes(int NumEjes);
        void setNumLlantas (int NumLlantas);

        /** Función que devuelve la división entre el peso y la potencia
              @return Peso/Potencia en libras/HP
        */
        float RelacionPesoPotencia();

        //Funciones get
        std::string getMarca() const;
        std::string getModelo() const;
        int getAnyo() const;
        int getCapacidad() const;
        float getPeso() const;
        float getPotencia() const;
        float getLargo() const;
        float getAncho() const;
        float getAltura() const;
        float getVelocidad() const;
        float getDesaceleracion() const;
        int getNumEjes() const;
        int getNumLlantas () const;

        virtual std::string to_string() const;

//-------------------------------------------------------------------------
         //CARACTERÍSTICAS DINÁMICAS

        /** Calcula la fuerza de resistencia del aire en libras que tiene un
              vehículo en movimiento, depende del área tranversal del vehículo,
              y el cuadrado de la velocidad
              @return la resistencia del aire en libras
        */
        virtual float Ra();

        /** Calcula la fuerza de resistencia debido a la pendiente del camino,
              que tiene un vehículo, es una función amiga, ya que usa el
              miembro pendiente de la clase camino
              @param via Una referencia a un objeto camino
```

```
            @return la fuerza de resistencia debido a la pendiente del camino
                        en libras
        */
        float Rp(camino& via);

        /** Calcula la fuerza de Resistencia al rodamiento para los automóviles
            sobre un pavimento liso, véase la fórmula 3.11
            depende del cuadrado de la velocidad y del peso del vehículo
            @return la fuerza de resistencia al rodamiento en libras
        */
        virtual float Rt();

        /** Calcula la fuerza de Resistencia a la curva para los automóviles,
            véase la fórmula 3.13 depende del cuadrado de la velocidad, del
            peso del vehículo, de la aceleración
            de la gravedad y del radio de curvatura
            @param via El camino
            @return la fuerza de resistencia a la curva en libras
        */
        float Rc(camino& via);

        /** Calcula la Potencia que produce el motor del automóvil para vencer
            las fuerzas de resistencia del aire, de la pendiente, de la
            curva, y de fricción para poner al vehículo en movimiento.
            1 H.P. = 550 pies-libras/segundo
            @param via El camino
            @return la potencia desarrollada por el vehículo en H.P.
        */
        virtual float Pot(camino& via);


        /** Calcula la distancia de frenado, se supone que la desaceleración
            es constante, en función de la velocidad del vehículo, el
            coeficiente  de fricción entre la superficie de rodamiento y las
            llantas, el peso (aceleración de la gravedad) y el peralte de la
            vía
            @param via El camino
            @param velocFinal La velocidad a la que se pretende llegar
                  durante la desaceleración
            @return la distancia de frenado en pies
        */
        float Db(camino& via, float velocFinal);

        /** Calcula la distancia S visual de paro, depende de la distancia
            recorrida en el tiempo de percepción - reacción del conductor
            y la distancia de frenado, cuando el vehículo se detiene
            completamente.
            @param via El camino
            @param chofer El conductor del vehículo
            @return la distancia S visual de paro en pies
        */
        float S(camino& via, conductor& chofer);

        /** Distancia Visual de Paro, es la distancia visual mínima requerida
            para que un conductor detenga su vehículo, después de ver un
            objeto en la trayectoria del vehículo sin impactarse con ese
        objeto
            @param via El camino
            @param chofer El conductor del vehículo
            @return la distancia visual de parada en pies.
        */
        float DVP(camino& via, conductor& chofer);

//-------------------------------------------------------------------------

        /** Funciones para calcular la distancia visual de rebase, que es la
```

```
                  Distancia visual mínima requerida en una carretera de dos
                  carriles y de dos sentidos, que le permitirá a un conductor
                  terminar una maniobra de rebase sin chocar con un vehículo que
                  venga en sentido contrario y sin cerrar el paso al vehículo
                  rebasado. En todos los casos la velocidad es del vehículo que
                  rebasa
            */

            /** Distancia recorrida en la maniobra inicial, en el tiempo de
                  percepción - reacción y durante la aceleración inicial, hasta el
                  punto en el cual el vehículo que rebasa entra justo al carril
                  izquierdo. Se usa interpolación
                  lineal para los valores intermedios. Véase la tabla 3.6 del libro
                  "Ingeniería de tránsito y carreteras" de Nicholas J. Garber y
                  Lester A. Hoel.
                  @return la distancia d1 recorrida por el vehículo en la maniobra
                        inicial en pies.
            */
            float d1();

            /** Distancia recorrida durante el tiempo que transcurre mientras que
                  elvehículo que rebasa recorre el carril izquierdo
                  @return la distancia recorrida en el carril izquierdo en pies
            */
            float d2();

            /** Distancia recorrida por el vehículo que rebasa y el vehículo que
                  viene en sentido contrario, al terminar la maniobra de rebase
                  @return la distancia d3 en pies
            */
            float d3();

            /** Distancia recorrida por el vehículo en sentido contrario durante
                  Dos tercios del tiempo, cuando el vehículo que rebasa está en el
                  carril      izquiuerdo (generalemente se toma como 2/3 d2)
                  @return la distancia d4 en pies
            */
            float d4();

            /** Distancia visual de rebase = d1 + d2 + d3 + d4
                  @return la diatancia visual de rebase en pies
            */
            float distVisualRebase();
};
//---------------------------------------------------------------------------
#endif
```

Y aquí está la implementación de la clase vehículo:

```
//---------------------------------------------------------------------------

#include "Vehículo.h"
/** Función que calcula el área frontal del vehículo
      @return Ancho por Altura en pies²
*/
float vehiculo::areaFrontal() const {
      return getAncho() * getAltura();
}

/** Función virtual, para redefinir en cada clase derivada
    @param Velocidad La velocidad del vehículo
      @return 0
```

```
*/
float vehiculo::aceleracion(float Velocidad) {
	return 0;
}

/** Función que pone la marca del vehículo
	@param Marca La marca del vehículo
*/
void vehiculo::setMarca(std::string Marca){
	marca = Marca;
}

/** Función que pone el modelo del vehículo
	@param Modelo El modelo del vehículo
*/
void vehiculo::setModelo(std::string Modelo){
	modelo = Modelo;
}

/** Función que pone el modelo del vehículo
	@param year El año de fabricación del vehículo
*/
void vehiculo::setAnyo(int year){
	anyo = year;
}

/** Función que pone la capacidad del vehículo como un entero
	@param Capacidad El número de personas que pueden viajar, incluye al conductor
*/
void vehiculo::setCapacidad(int Capacidad = 1){
	if(Capacidad < 1) capacidad = 1;
	else capacidad = Capacidad;
}

/** Función que pone la placa del vehículo
	@param Placa La identificación única del vehículo
*/
void vehiculo::setPlaca(std::string Placa){
	placa = Placa;
}

/** Función que coloca el peso del vehículo en libras
	@param Placa La identificación única del vehículo
*/
void vehiculo::setPeso(float Peso){
	if(Peso < 0.0) peso = 0.0;
	else peso = Peso;
}

/** Función que coloca la potencia del vehículo en Horse power
	@param Potencia La potencia máxima desarrollada por el automóvil en Horse
		Power
*/
void vehiculo::setPotencia(float Potencia){
	if(potencia < 0.0) potencia = 1.0;
	else potencia = Potencia;
}

/** Función que coloca el largo del vehículo
	@param Largo El largo del vehículo en pies
*/
void vehiculo::setLargo(float Largo){
	if(Largo < 0.0) largo = 0.0;
	else largo = Largo;
}
```

```
/** Función que coloca el ancho del vehículo
      @param Ancho El ancho del vehículo en pies
*/
void vehiculo::setAncho(float Ancho){
      if(Ancho < 0.0) ancho = 0.0;
      else ancho = Ancho;
}

/** Función que coloca la altura del vehículo
      @param Altura La altura del vehículo en pies
*/
void vehiculo::setAltura(float Altura){
      if(Altura < 0.0) altura = 0.0;
      else altura = Altura;
}

/** Función que coloca velocidad del vehículo
      @param Velocidad La velocidad del vehículo en pies/segundo
*/
void vehiculo::setVelocidad(float Velocidad){
      if(Velocidad < 0.0 && Velocidad > 240.0) velocidad = 0.0;
      else velocidad = Velocidad;
}

void vehiculo::setDesaceleracion(float Desacel) {
      if(Desacel < 0.0 && Desacel > 14.8)
            desaceleracion = 11.2;
      else desaceleracion = Desacel;
}

/** Función que coloca el número de ejes del vehículo
      @param Velocidad La velocidad del vehículo en pies/segundo
*/
void vehiculo::setNumEjes(int NumEjes){
      if(NumEjes < 2) numEjes = 2;
      else numEjes = NumEjes;
}

void vehiculo::setNumLlantas (int NumLlantas){
      if(NumLlantas < 2) numLlantas = 2;
      else numLlantas = NumLlantas;
}

/** Función que devuelve la marca del vehículo como un string
      @return La marca del vehículo
*/
std::string vehiculo::getMarca()const {
      return marca;
}

/** Función que devuelve el modelo del vehículo como un string
      @return El modelo del vehículo
*/
std::string vehiculo::getModelo()const {
      return modelo;
}

/** Función que devuelve el año de fabricación del vehículo como un int
      @return El año de fabricación del vehículo
*/
int vehiculo::getAnyo() const {
      return anyo;
}

/** Función que devuelve la capacidad en pasajeros del vehículo como un int
      @return La capacidad del vehículo, incluye al conductor
```

```
*/
int vehiculo::getCapacidad() const {
	return capacidad;
}

/** Función que devuelve el peso del vehículo como un float
	@return El peso bruto del vehículo en libras
*/
float vehiculo::getPeso() const {
	return peso;
}

/** Función que devuelve la potencia del vehículo como un float
	@return La potencia del vehículo en Horse Power
*/
float vehiculo::getPotencia() const {
	return potencia;
}

/** Función que devuelve el largo del vehículo como un float
	@return El largo del vehículo en pies
*/
float vehiculo::getLargo() const {
	return largo;
}

/** Función que devuelve el ancho del vehículo como un float
	@return El ancho del vehículo en pies
*/
float vehiculo::getAncho() const {
	return ancho;
}

/** Función que devuelve la altura del vehículo como un float
	@return La altura del vehículo en pies
*/
float vehiculo::getAltura() const {
	return altura;
}

/** Función que devuelve la velocidad del vehículo como un float
	@return La velocidad del vehículo en millas/hora
*/
float vehiculo::getVelocidad() const {
	return velocidad;
}

float vehiculo::getDesaceleracion() const {
	return desaceleracion;
}

/** Función que devuelve el número de ejes del vehículo como un int
	@return El número de ejes del vehículo
*/
int vehiculo::getNumEjes() const {
	return numEjes;
}

/** Función que devuelve el número de llantas del vehículo como un int
	@return El número de llantas del vehículo
*/
int vehiculo::getNumLlantas () const {
	return numLlantas;
}

float vehiculo::RelacionPesoPotencia() {
```

```
	return getPeso() / getPotencia();
}

std::string vehiculo::to_string() const {
	std::ostringstream sb;
	sb << "Características del Vehículo: " << std::endl
		<< "Marca: " << marca << std::endl
		<< "Modelo: " << modelo << std::endl
		<< "Año de fabricación: " << anyo << std::endl
		<< "Capacidad: " << capacidad << " personas\n"
		<< "Placa: " << placa << std::endl
		<< "Peso bruto: " << peso << " libras\n"
		<< "Potencia: " << potencia << " H.P.\n"
		<< "Largo: " << largo << " pies\n"
		<< "Ancho: " << ancho << " pies\n"
		<< "Altura: " << altura << "pies\n"
		<< "Velocidad: " << velocidad << " millas/hora\n"
		<< "Número de Ejes: " << numEjes << std::endl
		<< "Número de Llantas: " << numLlantas << std::endl;

	return sb.str();
}

//CARACTERÍSTICAS DINÁMICAS
/** Calcula la fuerza de resistencia del aire en libras que tiene un
	vehículo en movimiento, depende del área tranversal del vehículo,
	y del cuadrado de la velocidad. Véase la fórmula 3.10
	@return la resistencia del aire en libras
*/
float vehiculo::Ra() {
	return 0.5 * (mph_fps * mph_fps * rho * C_D * areaFrontal() * getVelocidad()
			* getVelocidad()) / g;
}

/** Calcula la fuerza de resistencia debido a la pendiente del camino,
	que tiene un vehículo,
	@param via Una referencia a un objeto camino
	@return la fuerza de resistencia debido a la pendiente del camino
		en libras
*/
float vehiculo::Rp(camino& via) {
	return getPeso() * via.getPendiente();
}

/** Calcula la fuerza de Resistencia al rodamiento para los automóviles
	sobre un pavimento liso, véase la fórmula 3.11
	depende del cuadrado de la velocidad y del peso del vehículo
	@return la fuerza de resistencia al rodamiento en libras
*/
float vehiculo::Rt(){
	return (C_rs + mph_fps * mph_fps * C_rv * getVelocidad() * getVelocidad()) *
getPeso();
}

/** Calcula la fuerza de Resistencia a la curva para los automóviles,
	véase la fórmula 3.13
	depende del cuadrado de la velocidad, del peso del vehículo, de la aceleración
	de la gravedad y del radio de curvatura
	@return la fuerza de resistencia a la curva en libras
*/
float vehiculo::Rc(camino& via){
	if(IsInfinite(via.getRadioGiro())) return 0.0;
	else return 0.5 * (mph_fps * mph_fps * getVelocidad() * getVelocidad() *
getPeso())
			/ g / via.getRadioGiro();
}
```

```
/** Calcula la Potencia que produce el motor del automóvil para vencer
      las fuerzas de resistencia del aire, de la pendiente, de la curva,
      y de fricción para poner al vehículo en movimiento.
      1 H.P. = 550 pies-libras/segundo
      @return la Potencia del automóvil en H.P.
*/
float vehiculo::Pot(camino& via) {
      return ( Ra() + Rp(via) + Rt() + Rc(via)) * getVelocidad() * mph_fps / 550.0;
}

/** Calcula la distancia de frenado, se supone que la desaceleración
      es constante, en función de la velocidad del vehículo, el coeficiente
      de fricción entre la superficie de rodamiento y las llantas, el peso
      (aceleración de la gravedad) y la pendiente de la vía
      @param via El camino
      @param velocFinal La velocidad a la que se pretende llegar durante la
      desaceleración
      @return la distancia de frenado en pies
*/
float vehiculo::Db(camino& via, float velocFinal) {
      return (getVelocidad() * getVelocidad() - velocFinal * velocFinal) * mph_fps
* mph_fps
            / (2.0 * g * (via.getCoeFriccion() + via.getPendiente()));
}

/** Calcula la distancia S visual de paro, depende de la distancia
      recorrida en el tiempo de percepción - reacción del conductor
      y la distancia de frenado, cuando el vehículo se detiene completamente.
      @param via El camino
      @param chofer El conductor del vehículo
      @return la distancia S visual de paro en pies
*/
float vehiculo::S(camino& via, conductor& chofer) {
    return mph_fps * getVelocidad() * chofer.getTiempoP_R() + Db(via, 0.0);
}

/** Distancia Visual de Paro, es la distancia visual mínima requerida
      para que un conductor detenga su vehículo, después de ver un objeto
      en la trayectoria del vehículo sin impactarse con ese objeto
      @param via El camino
      @param chofer El conductor del vehículo
      @return la distancia visual de parada en pies.
*/
float vehiculo::DVP(camino& via, conductor& chofer) {
      return mph_fps * getVelocidad() * chofer.getTiempoP_R() + getVelocidad() *
      getVelocidad()  *  mph_fps  *  mph_fps  /  (2.0*g*(getDesaceleracion()  /  g  +
via.getPendiente()));
}
//--------------------------------------------------------------------------

/** Funciones para calcular la distancia visual de rebase, que es la distancia
      visual mínima requerida en una carretera de dos carriles y de dos sentidos,
      que le permitirá a un conductor terminar una maniobra de rebase sin chocar
      con un vehículo que venga en sentido contrario y sin cerrar el paso al
      vehículo rebasado
*/

/** Distancia recorrida en la maniobra inicial, en el tiempo de percepción -
      reacción y durante la aceleración inicial, hasta el punto en el cual
      el vehículo que rebasa entra justo al carril izquierdo. Se usa interpolación
      lineal para los valores intermedios. Véase la tabla 3.6 del libro
      "Ingeniería de tránsito y carreteras" de Nicholas J. Garber y Lester A. Hoel.
      @return la distancia d1 recorrida por el vehículo en la maniobra inicial en
      pies.
*/
```

```
float vehiculo::d1() {
	float aceleracionPromedio;	//aceleración en millas/hora/segundo
	float tiempo1;			//en segundos
	const float m = 10.0; //Diferencia de velocidad entre el vehículo rebasado y
el que obstruye
	if(getVelocidad() < 30.0) {
		aceleracionPromedio = 1.2;
		tiempo1 = 3.0;
	}
	else if(30.0 <= getVelocidad() && getVelocidad() < 40.0) {
		aceleracionPromedio = 1.43 + (1.4 - 1.43) * (getVelocidad() - 40.0) /
(30.0 - 40.0);

		tiempo1 = 4.0 + (3.6 - 4.0) *(getVelocidad() - 40.0) / (30.0 - 40.0);
	}

	else if(40.0 <= getVelocidad() && getVelocidad() < 50.0) {
		aceleracionPromedio = 1.47 + (1.43 - 1.47) * (getVelocidad() - 50.0) /
(40.0 - 50.0);

		tiempo1 = 4.3 + (4.0 - 4.3) *(getVelocidad() - 50.0) / (40.0 - 50.0);
	}

	else if(50.0 <= getVelocidad() && getVelocidad() < 60.0) {
		aceleracionPromedio = 1.50 + (1.47 - 1.50) * (getVelocidad() - 60.0) /
(50.0 - 60.0);

		tiempo1 = 4.5 + (4.3 - 4.5) *(getVelocidad() - 60.0) / (50.0 - 60.0);
	}

	else if(60.0 <= getVelocidad() && getVelocidad() < 70.0) {
		aceleracionPromedio = 1.53 + (1.50 - 1.53) * (getVelocidad() - 70.0) /
(60.0 - 70.0);

		tiempo1 = 4.8 + (4.5 - 4.8) *(getVelocidad() - 70.0) / (60.0 - 70.0);
	}
	else {
		aceleracionPromedio = 1.58;
		tiempo1 = 5.2;
	}

	return mph_fps * tiempo1 * (getVelocidad() - m + aceleracionPromedio * tiempo1
/ 2.0);
}

/** Distancia recorrida durante el tiempo que transcurre mientras que el
	vehículo que rebasa recorre el carril izquierdo
	@return la distancia recorrida en el carril izquierdo en pies
*/
float vehiculo::d2() {
	float tiempo2;		//segundos
	if(getVelocidad() < 30.0) {
		tiempo2 = 8.8;
	}
	else if(30.0 <= getVelocidad() && getVelocidad() < 40.0) {
		tiempo2 = 10.0 + (9.3 - 10.0) *(getVelocidad() - 40.0) / (30.0 - 40.0);
	}

	else if(40.0 <= getVelocidad() && getVelocidad() < 50.0) {
		tiempo2 = 10.7 + (10.0 - 10.7) *(getVelocidad() - 50.0) / (40.0 -
50.0);
	}

	else if(50.0 <= getVelocidad() && getVelocidad() < 60.0) {
		tiempo2 = 11.3 + (10.7 - 11.3) *(getVelocidad() - 60.0) / (50.0 -
60.0);
```

```
	}

	else if(60.0 <= getVelocidad() && getVelocidad() < 70.0) {
		tiempo2 = 12.0 + (11.3 - 12.0) *(getVelocidad() - 70.0) / (60.0 -
70.0);
	}
	else {
		tiempo2 = 12.6;
	}

	return mph_fps * getVelocidad() * tiempo2;
}

/** Distancia recorrida por el vehículo que rebasa y el vehículo que viene
	en sentido contrario, al terminar la maniobra de rebase
	@return la distancia d3 en pies
*/
float vehiculo::d3() {
	if(getVelocidad() < 30.0)
		return 70.0;
	else if(30.0 <= getVelocidad() && getVelocidad() < 40.0) {
		return 180.0 + (100.0 - 180.0) *(getVelocidad() - 40.0) / (30.0 -
40.0);
	}

	else if(40.0 <= getVelocidad() && getVelocidad() < 50.0) {
		return 250.0 + (180.0 - 250.0) *(getVelocidad() - 50.0) / (40.0 -
50.0);
	}

	else if(50.0 <= getVelocidad() && getVelocidad() < 60.0) {
		return 300 + (250.0 - 300.0) *(getVelocidad() - 60.0) / (50.0 - 60.0);
	}

	else if(60.0 <= getVelocidad() && getVelocidad() < 70.0) {
		return 350.0 + (300.0 - 350.0) *(getVelocidad() - 70.0) / (60.0 -
70.0);
	}
	else {
		return 390.0;
	}
}

/** Distancia recorrida por el vehículo en sentido contrario durante dos
	tercios del tiempo, cuando el vehículo que rebasa está en el carril
	izquiuerdo (generalemente se toma como 2/3 d2)
	@return la distancia d4 en pies
*/
float vehiculo::d4() {
	return d2() * 2.0 / 3.0;
}

/** Distancia visual de rebase = d1 + d2 + d3 + d4
	@return la diatancia visual de rebase en pies
*/
float vehiculo::distVisualRebase() {
    return d1() + d2() + d3() + d4();
}
```

3.1.5. Clase automóvil

He aquí la clase automóvil

```
//---------------------------------------------------------------------------

#ifndef AutomovilH
#define AutomovilH
#include "C:\Operaciones de Transito\...\Vehiculo\Vehículo.h"


/** Clase que implementa un automóvil, se deriva de la clase vehículo
      @param Marca La marca del automóvil
      @param Modelo El modelo del automóvil
      @param year El año de fabricación del automóvil
      @param Capacidad El número de personas que pueden viajar, incluye al
            conductor
      @param Placa La identificación única del automóvil
      @param Peso El peso bruto del automóvil en libras
      @param Potencia La potencia máxima desarrollada por el automóvil en Horse
            Power
      @param Largo El largo del automóvil en pies
      @param Ancho El ancho del automóvil en pies
      @param Altura La altura del automóvil en pies
      @param WB_1 La distancia entre los ejes, comenzando del eje frontal
      @param Alfa Constante tasa máxima de aceleración, en pies/segundo²
      @param Beta Constante que se multiplica a la velocidad, junto con alfa nos
            Dan la aceleración (1/seg)
      @param RadioGiro El radio de giro, por defecto es el mínimo, en pies
      @param Velocidad La velocidad del automóvil en millas/hora
      @param NumEjes El número de ejes del automóvil, normalmente 2.
      @param NumLlantas El número de llantas del automóvil
      @param Simbolo El símbolo que indica que tipo de vehículo es (P automóvil
            de pasajeros)
*/
class automovil : public vehiculo {
      private:
            float WB1;
            float alfa;
            float beta;
            float radioGiro;
            char simbolo;

      public:
            //Constructores
            /** Construir un automóvil de pasajeros con los datos determinados


            */
            automovil(std::string Marca, std::string Modelo, int year, int
                  Capacidad, std::string Placa, float Peso, float Potencia, float
                  Largo, float Ancho, float Altura, float Velocidad, float
                  Desaceleracion, float WB_1, float Alfa, float Beta, float
                  RadioGiro, int NumEjes = 2, int NumLlantas = 4, char Simbolo =
```

```
            'P') : vehiculo(Marca, Modelo, year, Capacidad, Placa, Peso,
            Potencia, Largo, Ancho, Altura, Velocidad, Desaceleracion,
            NumEjes, NumLlantas), WB1(WB_1), alfa(Alfa), beta(Beta),
            radioGiro(RadioGiro), simbolo(Simbolo) {
            setVelocidad(Velocidad);
      };

      /** Construir un automóvil con campos predeterminados
            La mayoría de los datos se han sacado de la Tabla 3.2 Dimensiones
            de los      vehículos de diseño del libro "Ingeniería de
            tránsito y carreteras" de Nicholas J. Garber y
            Lester A. Hoel, para un automóvil de pasajeros
      */
      automovil(): vehiculo(), WB1(11.0), alfa(-1.0), beta(0),
            radioGiro(24.0), simbolo('P') {
                  setVelocidad(30.0);
      };

      /**Función que calcula la aceleración, depende de la velocidad y de
            la relación peso/potencia, usa las constantes alfa y beta para
            el cálculo
            @param Veloc La velocidad del automóvil en millas/hora
            @return la aceleración del automóvil en (pies/seg²)
      */
      float aceleracion(float Veloc);

      /** Calcula la tasa de aceleración máxima alfa
            @param Velocidad La velocidad en millas/hora o directamente el
                  valor de Alfa.
            @return alfa
      */
      void setAlfa(float Alfa);

      void setBeta(float Beta);
      void setWB1(float WB_1);


      std::string to_string() const;

      float getAlfa() const;
      float getBeta() const;
      float getWB1() const;


//------------------------------------------------------------------------
      //CARACTERÍSTICAS CINEMÁTICAS

      /** Calcula la velocidad en función del tiempo
            @param t Tiempo en el cual se calcula la velocidad
            @param t0 Tiempo para el cual la velocidad es u0 (velocidad
                  inicial)
```

```
                @return la velocidad en pies/segundo
            */
            float velocidad_t(float t, float t0);

            /** Calcula la posición en función del tiempo
                @param x0 La posición inicial del automóvil en el tiempo t0
                @param t Tiempo en el cual se calcula la velocidad
                @param t0 Tiempo para el cual la velocidad es u0 (velocidad
                      inicial)
                @return la posición en pies
            */
            float posicion_t(float x0, float t, float t0);

//-------------------------------------------------------------------------
            //CARACTERÍSTICAS DINÁMICAS

            /** Calcula la fuerza de resistencia del aire en libras que tiene un
                vehículo en movimiento, depende del área tranversal del
                vehículo, y el cuadrado de la velocidad
                @return la resistencia del aire en libras
            */
            float Ra();

        /** Calcula la fuerza de Resistencia al rodamiento para los automóviles
                sobre un pavimento liso, véase la fórmula 3.11 depende del
                cuadrado de la velocidad y del peso del vehículo
                @return la fuerza de resistencia al rodamiento en libras
            */
            float Rt();

            /** Calcula la Potencia que produce el motor del automóvil para vencer
                las fuerzas de resistencia del aire, de la pendiente, de la
                curva,
                y de fricción para poner al vehículo en movimiento.
                1 H.P. = 550 pies-libras/segundo
                @param via El camino
                @return la potencia desarrollada por el vehículo en H.P.
            */
            float Pot(camino& via);
};
//-------------------------------------------------------------------------
#endif
```

Mostramos enseguida la implementación en C++ de la clase automóvil

```
//-------------------------------------------------------------------------


#include "Automovil.h"
/** Calcula la aceleración en función de la velocidad
```

```
	@param Velocidad La velocidad, en este caso es una variable, en millas/hora
	@return La aceleración, en pies/segundo²
*/
float automovil::aceleracion(float Veloc) {

	return  getAlfa() - getBeta() * Veloc * mph_fps;
}

/** Función que calcula la tasa máxima de aceleración en función de la velocidad
	si no se da el parámetro Alfa o es un valor negativo.
	 vea la Fig. 3.4 a) del libro "Ingeniería de tránsito y carreteras"
	de Nicholas J. Garber y Lester A. Hoel. Se usa interpolación lineal.
	En caso contrario Alfa se asigna a alfa
	@param Alfa La constante alfa
*/
void automovil::setAlfa(float Alfa) {
	alfa = Alfa;
	if(alfa < 0.0) {
		if(0.0 <= getVelocidad() && getVelocidad() <= 10.0) {
			if(25.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
30.0) {
				alfa = 9.3 - (9.3 - 7.8)/5.0 *(RelacionPesoPotencia() -
25.0);
			}
			else if(30.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 35.0) {
				alfa = 7.8 - (7.8 - 6.75)/5.0 * (RelacionPesoPotencia()
- 30.0);
			}
			else alfa = 6.5;
		}
		else if(10.0 < getVelocidad() && getVelocidad() <= 20.0) {
			if(25.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
30.0) {
				alfa = 8.9 - (8.9 - 7.5)/5.0 *(RelacionPesoPotencia() -
25.0);
			}
			else if(30.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 35.0) {
				alfa = 7.5 - (7.5 - 6.5)/5.0 * (RelacionPesoPotencia() -
30.0);
			}
			else alfa = 6.25;
		}
		else if(20.0 < getVelocidad() && getVelocidad() <= 30.0) {
			if(25.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
30.0) {
				alfa = 8.5 - (8.5 - 7.2)/5.0 *(RelacionPesoPotencia() -
25.0);
			}
			else if(30.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
```

```
<= 35.0) {
                        alfa = 7.2 - (7.2 - 6.25)/5.0 * (RelacionPesoPotencia()
- 30.0);
                  }
                  else alfa = 6.0;
            }
            else if(30.0 < getVelocidad() && getVelocidad() <= 40.0) {
                  if(25.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
30.0) {
                        alfa = 8.2 - (8.2 - 6.8)/5.0 *(RelacionPesoPotencia() -
25.0);
                  }
                  else if(30.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 35.0) {
                        alfa = 6.8 - (6.8 - 5.8)/5.0 * (RelacionPesoPotencia() -
30.0);
                  }
                  else alfa = 5.75;
            }
            else if(40.0 < getVelocidad() && getVelocidad() <= 50.0) {
                  if(25.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
30.0) {
                        alfa = 7.75 - (7.75 - 6.5)/5.0 *(RelacionPesoPotencia()
- 25.0);
                  }
                  else if(30.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 35.0) {
                        alfa = 6.5 - (6.5 - 5.5)/5.0 * (RelacionPesoPotencia() -
30.0);
                  }
                  else alfa = 5.25;
            }
            //Este valor de alfa = 3.5 se asume
            else alfa = 3.5;
      }
}

void automovil::setBeta(float Beta) {
      if(Beta < 0) beta = 0.0;
      else beta = Beta;
}

void automovil::setWB1(float WB_1) {
      WB1 = WB_1;
}

float automovil::getAlfa() const {
    return alfa;
}

float automovil::getBeta() const {
```

```
      return beta;
}

float automovil::getWB1() const {
    return WB1;
}

std::string automovil::to_string() const {
    std::ostringstream sb;
       sb << "Características del Automóvil:\n"
            << vehiculo::to_string()
            << "Distancia entre ejes: " << getWB1() << " pies" << std::endl
            << "Tasa máxima de aceleración Alfa = " << getAlfa() << " pies/seg²"
<< std::endl
            << "Constante Beta para la aceleración = " << getBeta() << " (1/seg)"
<< std::endl;
      return sb.str();
}


//-------------------------------------------------------------------------
//CARACTERÍSTICAS CINEMÁTICAS
/** Calcula la velocidad en función del tiempo t
      @param t Tiempo en el cual se calcula la velocidad
      @param t0 Tiempo para el cual la velocidad es u0 (velocidad inicial)
      @return la velocidad en pies/segundo
*/
float automovil::velocidad_t(float t, float t0) {
      if(getBeta() == 0.0)  {
            float u0 = getVelocidad() * mph_fps;
            return u0 + aceleracion(getVelocidad()) * (t -t0);
      }
      else {
            //Se asume la velocidad actual como la velocidad inicial
            float u0 = getVelocidad() * mph_fps;
            //Ecuación 3.8 de "Ingeniería de tránsito y carreteras" de
            //Nicholas J. Garber y Lester A. Hoel.
            return getAlfa() / getBeta() + (u0 - getAlfa() / getBeta())*exp(-
getBeta()*(t - t0));
    }

}


float automovil::posicion_t(float x0, float t, float t0) {
      if(getBeta() == 0.0) {
            float u0 = getVelocidad() * mph_fps;  //La   velocidad   en   pies   /
segundo
            float X = x0 + u0 * (t - t0) + aceleracion(getVelocidad()) * (t - t0)
* (t - t0) /2.0;
            return X;
```

```
	}
	else {
		float u0 = getVelocidad() * mph_fps;
		//Ecuación 3.9 de "Ingeniería de tránsito y carreteras" de Nicholas
J. Garber y Lester A. Hoel.
		float X = x0 + getAlfa() / getBeta() * (t - t0) - getAlfa() /
getBeta()/getBeta() *
		 (1.0 - exp(-getBeta()*(t-t0))) + u0 / getBeta() * (1.0 - exp(-
getBeta()*(t-t0)));
		return X;
	}
}

//-------------------------------------------------------------------------
//CARACTERÍSTICAS DINÁMICAS
/** Calcula la fuerza de resistencia del aire en libras que tiene un
	vehículo en movimiento, depende del área transversal del vehículo,
	y del cuadrado de la velocidad. Véase la fórmula 3.10
	@return la resistencia del aire en libras
*/
float automovil::Ra() {
	return 0.5 * (mph_fps * mph_fps * rho * C_D * areaFrontal() * getVelocidad()
			* getVelocidad()) / g;
}

/** Calcula la fuerza de Resistencia al rodamiento para los automóviles
	sobre un pavimento liso, véase la fórmula 3.11
	depende del cuadrado de la velocidad y del peso del vehículo
	@return la fuerza de resistencia al rodamiento en libras
*/
float automovil::Rt(){
	return (C_rs + mph_fps * mph_fps * C_rv * getVelocidad() * getVelocidad())
* getPeso();
}

/** Calcula la Potencia que produce el motor del automóvil para vencer
	las fuerzas de resistencia del aire, de la pendiente, de la curva,
	y de fricción para poner al vehículo en movimiento.
	1 H.P. = 550 pies-libras/segundo
	@return la Potencia del automóvil en H.P.
*/
float automovil::Pot(camino& via) {
	return ( Ra() + Rp(via) + Rt() + Rc(via)) * getVelocidad() * mph_fps / 550.0;
}
```

3.1.6. Clase semiremolque

A continuación se muestra la clase

```
//---------------------------------------------------------------------------

#ifndef semiRemolqueH
#define semiRemolqueH

#include "C:\Operaciones de Transito\...\Vehiculo\Vehículo.h"

//C_Dt Coeficiente de arrastre aerodinámico 0.5 para camiones
const float C_Dt = 0.5;

//Ca, constante 0.02445 comúnmente para camiones
const float Ca = 0.02445;

//Cb constante, comunmente 0.00044 seg/pie para camiones
const float Cb = 0.00044;

class semiRemolque : public vehiculo {
      private:
            float WB1;
            float WB2;
            float alfa;
            float beta;
            float radioGiro;
            char* simbolo;

      public:
            //Constructores
            /** Construir un semiremolque con los datos determinados

            */
            semiRemolque(std::string Marca, std::string Modelo, int year, int
                  Capacidad, std::string Placa, float Peso, float Potencia, float
                  Largo, float Ancho, float Altura, float Velocidad, float
                  Desaceleracion, float WB_1, float WB_2, float Alfa, float Beta,
                  float RadioGiro, int NumEjes = 5, int NumLlantas = 18, char*
                  Simbolo = "WB-40") : vehiculo(Marca, Modelo, year, Capacidad,
                  Placa, Peso, Potencia, Largo, Ancho, Altura, Velocidad,
                  Desaceleracion, NumEjes, NumLlantas), WB1(WB_1), WB2(WB_2),
                  alfa(Alfa), beta(Beta), radioGiro(RadioGiro), simbolo(Simbolo)
                  {
                  setVelocidad(Velocidad);
            };

            semiRemolque() : vehiculo(), simbolo("WB-40"), WB1(12.5), WB2(27.5),
                  alfa(-1.0), beta(0), radioGiro(41.0) {
                  setPeso(80000.0);
                  setPotencia(620.0);
                  setLargo(45.5);
                  setAncho(8.0);
                  setAltura(13.5);
```

```
            setVelocidad(30.0);
            setNumEjes(5);
            setNumLlantas(18);
        };


        void setAlfa(float Alfa);

        void setBeta(float Beta);
        void setWB1(float WB_1);
        void setWB2(float WB_2);


        std::string to_string() const;

        float getAlfa() const;
        float getBeta() const;
        float getWB1() const;
        float getWB2() const;

        /**Función que calcula la aceleración, depende de la velocidad y de
            La relación peso/potencia usa las constantes alfa y beta
                para el cálculo
            @param Veloc La velocidad del semiremolque en millas/hora
            @return la aceleración del semiremolque en (pies/seg²)
        */
        float aceleracion(float Veloc);

//-------------------------------------------------------------------------
        //CARACTERÍSTICAS CINEMÁTICAS

        /** Calcula la velocidad en función del tiempo
            @param t Tiempo en el cual se calcula la velocidad
            @param t0 Tiempo para el cual la velocidad es u0 (velocidad
            inicial)
            @return la velocidad en pies/segundo
        */
        float velocidad_t(float t, float t0);

        /** Calcula la posición en función del tiempo
            @param x0 La posición inicial del automóvil en el tiempo t0
            @param t Tiempo en el cual se calcula la velocidad
            @param t0 Tiempo para el cual la velocidad es u0 (velocidad
            inicial)
            @return la posición en pies
        */
        float posicion_t(float x0, float t, float t0);

//-------------------------------------------------------------------------
```

```
            //CARACTERÍSTICAS DINÁMICAS

            /** Calcula la fuerza de resistencia del aire en libras que tiene un
                  vehículo en movimiento, depende del área tranversal del
                  vehículo, y el cuadrado de la velocidad
                  @return la resistencia del aire en libras
            */
            float Ra();

            /** Calcula la fuerza de Resistencia al rodamiento para los camiones
                  sobre un pavimento liso, véase la fórmula 3.12
                  depende del cuadrado de la velocidad y del peso del vehículo
                  @return la fuerza de resistencia al rodamiento en libras
            */
            float Rt();

            /** Calcula la Potencia que produce el motor del automóvil para vencer
                  las fuerzas de resistencia del aire, de la pendiente, de la
                  curva, y de fricción para poner al vehículo en movimiento.
                  1 H.P. = 550 pies-libras/segundo
                  @param via El camino
                  @return la potencia desarrollada por el vehículo en H.P.
            */
            float Pot(camino& via);
};
//---------------------------------------------------------------------------
#endif
```

Y a continuación su implementación:

```
//---------------------------------------------------------------------------
#include "semiRemolque.h"
//---------------------------------------------------------------------------

/** Función que calcula la tasa máxima de aceleración en función de la velocidad
      si no se da el parámetro Alfa o es un valor negativo.
       vea la Fig. 3.4 b) del libro "Ingeniería de tránsito y carreteras"
      de Nicholas J. Garber y Lester A. Hoel. Se usa interpolación lineal.
      En caso contrario Alfa se asigna a alfa
      @param Alfa La constante alfa
*/
void semiRemolque::setAlfa(float Alfa) {
      alfa = Alfa;
      if(alfa < 0.0) {
            if(0.0 <= getVelocidad() && getVelocidad() <= 10.0) {
                  if(100.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
200.0) {
                        alfa = 2.85 - (2.85 - 1.80)/100.0 *(RelacionPesoPotencia()
```

```
- 100.0);
                  }
                  else if(200.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 300.0) {
                        alfa = 1.8 - (1.8 - 1.3)/100.0 * (RelacionPesoPotencia()
- 200.0);
                  }
                  else if(300.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 400.0) {
                        alfa = 1.3 - (1.3 - 1.26)/100.0 * (RelacionPesoPotencia()
- 300.0);
                  }
                  else alfa = 1.25;
            }
            else if(10.0 < getVelocidad() && getVelocidad() <= 20.0) {
                  if(100.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
200.0) {
                        alfa = 2.28 - (2.28 - 1.62)/100.0 *(RelacionPesoPotencia()
- 100.0);
                  }
                  else if(200.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 300.0) {
                        alfa = 1.62 - (1.62 - 1.3)/100.0 * (RelacionPesoPotencia()
- 200.0);
                  }
                  else if(300.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 400.0) {
                        alfa = 1.3 - (1.3 - 1.20)/100.0 * (RelacionPesoPotencia()
- 300.0);
                  }
                  else alfa = 1.10;
            }
            else if(20.0 < getVelocidad() && getVelocidad() <= 30.0) {
                  if(100.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
200.0) {
                        alfa = 2.20 - (2.20 - 1.50)/100.0 *(RelacionPesoPotencia()
- 100.0);
                  }
                  else if(200.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 300.0) {
                        alfa = 1.50 - (1.50 - 1.2)/100.0 * (RelacionPesoPotencia()
- 200.0);
                  }
                  else if(300.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 400.0) {
                        alfa = 1.2 - (1.2 - 1.10)/100.0 * (RelacionPesoPotencia()
- 300.0);
                  }
                  else alfa = 1.05;
            }
            else if(30.0 < getVelocidad() && getVelocidad() <= 40.0) {
```

```
                  if(100.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
200.0) {
                        alfa = 2.0 - (2.0 - 1.2)/100.0 *(RelacionPesoPotencia() -
100.0);
                  }
                  else if(200.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 300.0) {
                        alfa = 1.2 - (1.2 - 1.1)/100.0 * (RelacionPesoPotencia()
- 200.0);
                  }
                  else if(300.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 400.0) {
                        alfa = 1.1 - (1.1 - 0.7)/100.0 * (RelacionPesoPotencia()
- 300.0);
                  }
                  else alfa = 0.5;
            }
            else if(40.0 < getVelocidad() && getVelocidad() <= 50.0) {
                  if(100.0 <= RelacionPesoPotencia() && RelacionPesoPotencia() <=
200.0) {
                        alfa = 1.6 - (1.6 - 0.98)/100.0 *(RelacionPesoPotencia()
- 100.0);
                  }
                  else if(200.0 < RelacionPesoPotencia() && RelacionPesoPotencia()
<= 300.0) {
                        alfa = 0.98 - (0.98 - 0.6)/100.0 * (RelacionPesoPotencia()
- 200.0);
                  }
                  else alfa = 0.5;
            }
            //Este valor de alfa = 0.4 se asume
            else alfa = 0.4;
      }
}

void semiRemolque::setBeta(float Beta) {
      if(Beta < 0) beta = 0.0;
      else beta = Beta;
}

void semiRemolque::setWB1(float WB_1) {
      WB1 = WB_1;
}

void semiRemolque::setWB2(float WB_2) {
      WB2 = WB_2;
}


float semiRemolque::getAlfa() const {
    return alfa;
```

```
}

float semiRemolque::getBeta() const {
	return beta;
}

float semiRemolque::getWB1() const {
	return WB1;
}

float semiRemolque::getWB2() const {
	return WB2;
}

/** Calcula la aceleración en función de la velocidad
	@param Velocidad La velocidad, en este caso es una variable, en millas/hora
	@return La aceleración, en pies/segundo²
*/
float semiRemolque::aceleracion(float Veloc) {

	return  getAlfa() - getBeta() * Veloc * mph_fps;
}

//-------------------------------------------------------------------------
//CARACTERÍSTICAS CINEMÁTICAS
/** Calcula la velocidad en función del tiempo t
	@param t Tiempo en el cual se calcula la velocidad
	@param t0 Tiempo para el cual la velocidad es u0 (velocidad inicial)
	@return la velocidad en pies/segundo
*/
float semiRemolque::velocidad_t(float t, float t0) {
	if(getBeta() == 0.0)  {
		float u0 = getVelocidad() * mph_fps;
		return u0 + aceleracion(getVelocidad()) * (t -t0);
	}
	else {
		//Se asume la velocidad actual como la velocidad inicial
		float u0 = getVelocidad() * mph_fps;
		//Ecuación 3.8 de "Ingeniería de tránsito y carreteras"
		//de Nicholas J. Garber y Lester A. Hoel.
		return getAlfa() / getBeta() + (u0 - getAlfa() / getBeta())*exp(-
getBeta()*(t - t0));
    }

}


float semiRemolque::posicion_t(float x0, float t, float t0) {
	if(getBeta() == 0.0) {
		float u0 = getVelocidad() * mph_fps;  //La velocidad en pies / segundo
		float X = x0 + u0 * (t - t0) + aceleracion(getVelocidad()) * (t - t0)
```

```
* (t - t0) /2.0;
			return X;
		}
		else {
			float u0 = getVelocidad() * mph_fps;
			//Ecuación 3.9 de "Ingeniería de tránsito y carreteras" de
			//Nicholas J. Garber y Lester A. Hoel.
			float X = x0 + getAlfa() / getBeta() * (t - t0) - getAlfa() /
getBeta()/getBeta()
			 * (1.0 - exp(-getBeta()*(t-t0))) + u0 / getBeta() * (1.0 - exp(-
getBeta()*(t-t0)));
			return X;
		}
}



std::string semiRemolque::to_string() const {
		std::ostringstream sb;
		sb << "Características del Semiremolque: " << std::endl
			<< vehiculo::to_string()
			<< "Distancia entre ejes 1 y 2: " << getWB1() << std::endl
			<< "Distancia entre ejes 2 y 3: " << getWB2() << std::endl
			<< "Tasa máxima de aceleración Alfa = " << getAlfa() << " pies/seg²"
<< std::endl
			<< "Constante Beta para la aceleración = " << getBeta() << " (1/seg)"
<< std::endl;


		return sb.str();
}

//CARACTERÍSTICAS DINÁMICAS

/** Calcula la fuerza de resistencia del aire en libras que tiene un
		vehículo en movimiento, depende del área transversal del vehículo,
		y el cuadrado de la velocidad
		@return la resistencia del aire en libras
*/
float semiRemolque::Ra() {
		return 0.5 * (mph_fps * mph_fps * rho * C_Dt * areaFrontal() *
					getVelocidad() * getVelocidad()) / g;
}

/** Calcula la fuerza de Resistencia al rodamiento para los automóviles
		sobre un pavimento liso, véase la fórmula 3.11
		depende del cuadrado de la velocidad y del peso del vehículo
		@return la fuerza de resistencia al rodamiento en libras
*/
float semiRemolque::Rt(){
		return (Ca + mph_fps * Cb * getVelocidad() ) * getPeso();
```

```
}

/** Calcula la Potencia que produce el motor del automóvil para vencer
      las fuerzas de resistencia del aire, de la pendiente, de la curva,
      y de fricción para poner al vehículo en movimiento.
      1 H.P. = 550 pies-libras/segundo
      @return la Potencia del automóvil en H.P.
*/
float semiRemolque::Pot(camino& via) {
      return ( Ra() + Rp(via) + Rt() + Rc(via)) * getVelocidad() * mph_fps / 550.0;
}
```

3.1.7. Clase camino

La clase camino es así de simple y lo mostramos a continuación:

```
//---------------------------------------------------------------------------

#ifndef CaminoH
#define CaminoH
#include <System.Math.hpp>
#include <math.h>

//factor de conversión de millas/hora a pies/segundo
const double mph_fps = 5280.0 /3600.0;

//Aceleración de la gravedad (32.2 pies / segundo²)
const float g = 32.2;

/** Clase que implementa un tramo de un camino que tiene una pendiente y un
      ancho constante en dicho tramo
      @param Pendiente La pendiente del camino en decimal
      @param Ancho El ancho del camino en pies
      @param Longitud La longitud horizontal del tramo que tiene una pendiente
             y un ancho constante en dicho tramo, en pies
      @param RadioGiro El radio de giro de la curva, si es recta, es infinito, en
            pies
      @param CoeFric El coeficiente de fricción entre el pavimento y las llantas
            del vehículo
      @param SobreElev La sobre elevación o el peralte de la sección transversal
            del camino
      @param VelocDisenyo La velocidad de diseño de la vía, debe ser mayor a 10
            millas/hora
*/
class camino {
      private:
            float pendiente;   //En tanto por uno
            float ancho;
```

```
            float longitud; //Distancia horizontal
            double radioGiro;
            float coeFriccion;
            float sobreElev;
            float velocidadDisenyo;

      public:
            //Constructores
            camino(float Pendiente, float Ancho, float Longitud, double RadioGiro,
                  float  CoeFric,  float  SobreElev,  float  VelocDisenyo)  :
                  pendiente(Pendiente),    ancho(Ancho),    longitud(Longitud),
                  radioGiro(RadioGiro), eFriccion(CoeFric),sobreElev(SobreElev),
                  velocidadDisenyo(VelocDisenyo) {};

            camino() : pendiente(0.02), ancho(12.0), longitud(300.0),
                  radioGiro(Infinity), coeFriccion(0.35), sobreElev(0.05),
                  velocidadDisenyo(30.0) {};

            //funciones set
            void setPendiente(float Pendiente);
            void setAncho(float Ancho);
            void setLongitud(float Longitud);
            void setRadioGiro(double RadioGiro);
            void setCoeFriccion(float CoeFric);
            void setSobreElev(float SobreElev);
            void setVelocidadDisenyo(float VelocDise);

            //funciones get
            float getPendiente() const;
            float getAncho() const;
            float getLongitud() const;
            double getRadioGiro() const;
            float getCoeFriccion() const;
            float getSobreElev() const;
            float getVelocidadDisenyo() const;

            //Otras funciones
            float coeFriccionLateral();
            float radioGiroMinimo();
};
//---------------------------------------------------------------------------
#endif
```

Y nuevamente la implementación, ahora de la clase camino:

```
#include "Camino.h"
//---------------------------------------------------------------------------
void camino::setPendiente(float Pendiente){
      pendiente = Pendiente;
}
```

```
void camino::setAncho(float Ancho) {
	ancho = Ancho;
}

void camino::setLongitud(float Longitud){
	if(Longitud < 0.0)
		longitud = 0.0;
	else longitud = Longitud;
}

/** Establece el radio de giro de la línea de centros del camino.
	Como mínimo es 21 pies para un automóvil
	@param RadioGiro El radio de giro en la línea de centros, en pies
*/
void camino::setRadioGiro(double RadioGiro){
	if (RadioGiro < 7.0 ) {
		radioGiro = 7.0;
	}
	else radioGiro = RadioGiro;
}

void camino::setCoeFriccion(float CoeFric){
	if(CoeFric < 0.0 && CoeFric >= 0.5)
		coeFriccion = 0.35;
	else coeFriccion = CoeFric;
}

void camino::setSobreElev(float SobreElev) {
	if(SobreElev < 0.0 && SobreElev > 0.1)
		sobreElev = 0.05;
	else sobreElev = SobreElev;
}


double camino::getRadioGiro() const {
	return radioGiro;
}

float camino::getPendiente() const {
	return pendiente;
}

float camino::getAncho() const {
	return ancho;
}

float camino::getLongitud() const {
    return longitud;
}
```

```
float camino::getCoeFriccion() const {
    return coeFriccion;
}


float camino::getSobreElev() const {
     return sobreElev;
}

/** Función que coloca la velocidad de diseño en millas por hora
     se supone una velocidad de diseño, cuando ésta sobresale de los límites ,
     de 30 millas/hora
*/
void camino::setVelocidadDisenyo(float VelocDise) {
     if(VelocDise < 0.0 && VelocDise > 240.0)
           velocidadDisenyo = 30.0;
     else velocidadDisenyo = VelocDise;
}

/** Función que devuelve la velocidad de diseño del camino
     @return la velocidad de diseño del camino en millas/hora
*/
float camino::getVelocidadDisenyo() const {
    return velocidadDisenyo;
}

/** Calcula el coeficiente de fricción lateral para diferentes velocidades
     de diseño. Véase la tabla 3.3 del libro "Ingeniería de tránsito y carreteras"
     de Nicholas J. Garber y Lester A. Hoel.
     Se usa interpolación lineal para los valores intermedios. Se supone un
     coeficiente  de fricción lateral de 0.08 para velocidades de diseño mayores
     a 70 millas/hora
     @ return el coeficiente de fricción lateral
*/
float camino::coeFriccionLateral() {
     if(getVelocidadDisenyo() <= 30.0)
           return 0.16;
     else if (getVelocidadDisenyo() > 30.0 && getVelocidadDisenyo() <= 40.0) {
           return 0.15 - 0.01 *(getVelocidadDisenyo() - 40.0)/10.0;
     }
     else if (getVelocidadDisenyo() > 40.0 && getVelocidadDisenyo() <= 60.0) {
           return 0.12 - 0.03 *(getVelocidadDisenyo() - 60.0)/20.0;
     }
     else if (getVelocidadDisenyo() > 60.0 && getVelocidadDisenyo() <= 70.0) {
           return 0.10 - 0.02 *(getVelocidadDisenyo() - 70.0)/10.0;
     }
     else return 0.08;
}

/** Función que calcula el radio de giro mínimo, en función de la velocidad,
     la sobreelevación y el coeficiente de fricción lateral del camino
```

```
     @return el radio de giro mínimo en pies
*/
float camino::radioGiroMinimo() {
     return getVelocidadDisenyo() * getVelocidadDisenyo() * mph_fps * mph_fps
      /     (g * ( getSobreElev() + coeFriccionLateral()));
}
//---------------------------------------------------------------------------
```

3.1.8. Prueba de los programas de las características del conductor, del peatón, del vehículo y del camino.

En la programación de software hay niveles de programación, el programador usuario no requiere saber cómo están implementados los programas de las clases o simplemente de los módulos, que algoritmos se han usado, etc. El simil con un conductor de un vehículo es bastante adecuado, ya que para conducir bien no se necesita conocer sobre la mecánica interna del vehículo, basta con conocer las funciones de las palancas y el volante y algo más, pero no más: lo único que se necesita saber es la parte de la estructura del programa, qué es lo que hace y cómo se crean los objetos de esas clases, creadas por otros programadores, programadores de aplicaciones, que crean bibliotecas de clases o funciones para que otros los utilicen; es así que se logra la reutilización del software y se mantiene alejados del código fuente a personas poco conocedoras, logrando así la seguridad del software y la facilidad de su uso por parte de personas poco entrenadas en estos asuntos, a esto se llama ocultación de la información. Dicho esto probaremos las clases creadas resolviendo algunos problemas y tal que se puedan comprobar los resultados con los obtenidos por otros métodos como el cálculo manual.

He aquí el programa de prueba fuente, que resuelve una multitud de problemas para este capítulo, y muchos otros más, pero ya es suficiente.

```
#include <conio.h>
#include "C:\Operaciones de Transito\Persona\persona.h"
#include "C:\Operaciones de Transito\Persona\persona.cpp"
#include "C:\Operaciones de Transito\...\Peaton\Peaton.h"
#include "C:\Operaciones de Transito\...\Peaton\Peaton.cpp"
#include "C:\Operaciones de Transito\...\Conductor\Conductor.h"
#include "C:\Operaciones de Transito\...\Conductor\Conductor.cpp"
#include "C:\Operaciones de Transito\...\Vehiculo\Vehículo.h"
#include "C:\Operaciones de Transito\...\Vehiculo\Vehículo.cpp"
#include "C:\Operaciones de Transito\...\Automovil\Automovil.h"
#include "C:\Operaciones de Transito\...\Automovil\Automovil.cpp"
#include "C:\Operaciones de Transito\...\Camino\Camino.h"
#include "C:\Operaciones de Transito\...\Camino\Camino.cpp"
#include "C:\Operaciones de Transito\...\TractorSemiRemolque\semiRemolque.h"
#include "C:\Operaciones de Transito\...\TractorSemiRemolque\semiRemolque.cpp"
```

```
int main()
{
	/*	Ejemplo 3.2 del libro Ingeniería de tránsito y de carreteras de Nicholas
		J. Garber y Lester A. Hoel.
		Mostramos el problema
	*/
	std::cout << "La aceleración de un vehículo puede representarse mediante\n"
		  << "la siguiente ecuación du/dt = 3.3 - 0.04*u, donde u es la\n"
		  << "velocidad del vehículo en pies / segundo. Si el vehículo está\n"
		  << "viajando a 45 millas/hora, determine su velocidad despues de\n"
		  << "5 segundos de aceleración y la distancia recorrida durante\n"
		  << "ese tiempo\n\n";

	/*	Creamos un objeto auto1 para resolver este problema, con valores
		predeterminados, lo mostramos en pantalla y luego modificamos algunos
		datos miembros de acuerdo al problema
	*/

	//Creamos el auto1
	automovil auto1;

	// Mostramos el auto1, con sus valores predeterminados
	std::cout <<"El automóvil  auto1\n" << auto1.to_string() << std::endl;

	//	Cambiamos algunos atributos de nuestro auto1
	//	Establecemos alpha a 3.3
	auto1.setAlfa(3.3);

	//	Establecemos beta a 0.04
	auto1.setBeta(0.04);

	//	Establecemos la velocidad a 45 millas/hora
	auto1.setVelocidad(45.0);

	//mostramos los cambios de los atributos al auto1
	std::cout << "El automóvil  auto1 con las nuevas características\n"
		  << auto1.to_string() << std::endl;


	//	Calculamos la velocidad despues de 5 segundos de aceleración y lo
	//	mostramos en pantalla
	std::cout << "La velocidad del auto después de 5 segundos de aceleración es\n"
		  << auto1.velocidad_t(5., 0.) << " pies/seg" << std::endl;

	//	Calculamos la distancia recorrida despues de 5 segundos de aceleración
	//	y lo mostramos en pantalla
	std::cout << "La distancia recorrida del auto después de 5 seg de aceleración es\n"
		  << auto1.posicion_t(0., 5., 0.) << " pies" << std::endl;

	/*	Ejemplo 3.3 del libro Ingeniería de tránsito y de carreteras de Nicholas
```

```
            J. Garber y Lester A. Hoel.
            Mostramos el problema
      */

      std::cout << "\n\nDetermine la potencia en caballos de fuerza producida por un\n"
                  << "automóvil de pasajeros que viaja a una velocidad de 65
millas/hora\n"
                  << "en un camino recto con una pendiente del 5 por ciento con un\n"
                  << "pavimento liso. Suponga que el peso del carro es de 4000 libras\n"
                  << "y el área transversal es de 40 pies²\n\n";

      //     Trabajamos con el mismo auto1, sólo le cambiamos algunas datos

      auto1.setPeso(4000.0);
      auto1.setAltura(4.0);
      auto1.setAncho(10.0);
      auto1.setVelocidad(65.0);


      //     Creamos un objeto calle13 de la clase camino para poner sus datos, de
      //  acuerdo al problema, ya que la función Pot(&camino) de potencia,
      //     requiere de la pendiente y del radio de giro, datos de un objeto camino

      camino calle13;

      // Ponemos la pendiente de calle13
      calle13.setPendiente(5.0/100.0);

      // Como es recto, su radio de giro es infinito
      calle13.setRadioGiro(Infinity);

      //     Con estos datos adicionales, podemos calcular la potencia ejercida por
      //     el automóvil auto1

      std::cout << "La potencia producida por el auto1 es = "
                  << auto1.Pot(calle13) << " H.P. " << std::endl;


      /*     Ejemplo 3.5 del libro Ingeniería de tránsito y de carreteras de Nicholas
            J. Garber y Lester A. Hoel.
            Mostramos el problema
      */

      std::cout << "\n\nUna conductora viaja a 65 millas/hora en un viaducto, trata\n"
                  << "de salir de la vialidad usando una rampa de salida con una\n"
                  << "velocidad máxima de 35 millas/hora ¿En qué punto del viaducto?\n"
                  << "debe pisar el freno para reducir la velocidad hasta el máximo\n"
                  << "permisible en la rampa, justo antes de entrar a la rampa, si\n"
                  << "esta sección del viaducto tiene una pendiente descendente de
3%\n\n";
```

```
	/* Con los mismos objetos automóvil auto1 y camino calle13, resolvemos
		este problema
	*/

	// Cambiamos la pendiente de calle13 a -3%
	calle13.setPendiente(-3.0/100.0);

	//	La velocidad de auto1 es 65 millas/hora
	auto1.setVelocidad(65.0);

	// Normalmente la desaceleración es de 11.2 pies/seg², pero nos aseguramos
	auto1.setDesaceleracion(11.2);

	// Calculamos la distancia de paro del automóvil auto1, se necesita de la
	//pendiente y de la velocidad final. Lo mostramos
	std:: cout << "Los frenos deben aplicarse por lo menos a = "
			<< auto1.Db(calle13, 35.0) << " pies \n";


	/*	Ejemplo 3.6 del libro Ingeniería de tránsito y de carreteras de Nicholas
		J. Garber y Lester A. Hoel.
		Mostramos el problema
	*/

	std::cout << "\n\nUn conductor que viaja a 55 millas/hora en una pendiente\n"
			<< "descendiente de 5 por ciento en una carretera observa un
accidente\n"
			<< "frente a él, en el cual interviene un camión volcado que bloquea\n"
			<< "completamente el camino. Si el conductor pudo detener su
vehículo\n"
			<< "a 30 pies del camión ¿a qué distancia del camión observó el \n"
			<< "accidente por primera vez? Suponga que el tiempo de percepción -
\n"
			<< "reacción es de 2.5 segundos\n\n";


	//Trabajamos con los mismos objetos, auto1 y calle13, cambiamos sus datos
	// Establecemos la nueva velocidad de auto1
	auto1.setVelocidad(55.0);

	//	Cambiamos la pendiente de calle13
	calle13.setPendiente(-5.0/100.0);

	//creamos un conductor y establecemos su tiempo de percepción - reacción
	conductor adolfo;

	// Establecemos el tiempo de percepción - reacción en 2.5 segundos
	// Aunque este valor es predeterminado, lo hacemos explicitamente
	adolfo.setTiempoPercReacc(2.5);

	//Calculamos la distancia de observación del accidente y lo mostramos
```

```
      // en pantalla
      std::cout << "La distancia de observación del accidente es "
                   << auto1.S(calle13, adolfo) + 30.0 << " pies\n";


      getche();

      return 0;
}
```

Y aquí está la salida del programa de prueba:

```
La aceleración de un vehículo puede representarse mediante
la siguiente ecuación du/dt = 3.3 - 0.04*u, donde u es la
velocidad del vehículo en pies / segundo. Si el vehículo está
viajando a 45 millas/hora, determine su velocidad despues de
5 segundos de aceleración y la distancia recorrida durante
ese tiempo

El automóvil  auto1
Características del Automóvil:
Características del Vehículo:
Marca:
Modelo:
Año de fabricación: 1990
Capacidad: 5 personas
Placa:
Peso bruto: 4000 libras
Potencia: 80 H.P.
Largo: 19 pies
Ancho: 7 pies
Altura: 4.25pies
Velocidad: 30 millas/hora
Número de Ejes: 2
Número de Llantas: 4
Distancia entre ejes: 11 pies
Tasa máxima de aceleración Alfa = -1 pies/seg²
Constante Beta para la aceleración = 0 (1/seg)

El automóvil  auto1 con las nuevas características
Características del Automóvil:
Características del Vehículo:
Marca:
Modelo:
Año de fabricación: 1990
Capacidad: 5 personas
Placa:
Peso bruto: 4000 libras
Potencia: 80 H.P.
```

Largo: 19 pies
Ancho: 7 pies
Altura: 4.25pies
Velocidad: 45 millas/hora
Número de Ejes: 2
Número de Llantas: 4
Distancia entre ejes: 11 pies
Tasa máxima de aceleración Alfa = 3.3 pies/seg²
Constante Beta para la aceleración = 0.04 (1/seg)

La distancia recorrida del auto después de 5 seg de aceleración es 337.726 pies

Determine la potencia en caballos de fuerza producida por un automóvil de pasajeros que viaja a una velocidad de 65 millas/hora en un camino recto con una pendiente del 5 por ciento con un pavimento liso. Suponga que el peso del carro es de 4000 libras y el área transversal es de 40 pies²

La potencia producida por el auto1 es = 77.0627 H.P.

Una conductora viaja a 65 millas/hora en un viaducto, trata de salir de la vialidad usando una rampa de salida con una velocidad máxima de 35 millas/hora ¿En qué punto del viaducto? debe pisar el freno para reducir la velocidad hasta el máximo permisible en la rampa, justo antes de entrar a la rampa, si esta sección del viaducto tiene una pendiente descendente de 3%

Los frenos deben aplicarse por lo menos a = 313.147 pies

Un conductor que viaja a 55 millas/hora en una pendiente descendiente de 5 por ciento en una carretera observa un accidente frente a él, en el cual interviene un camión volcado que bloquea completamente el camino. Si el conductor pudo detener su vehículo a 30 pies del camión ¿a qué distancia del camión observó el accidente por primera vez? Suponga que el tiempo de percepción - reacción es de 2.5 segundos

La distancia de observación del accidente es 568.474 pies

3.2. Estudios de Ingeniería de Tránsito

Necesitamos desarrollar las clases de las funciones especiales y las funciones estadísticas que nos servirán de insumos para el desarrollo de la tesis.

3.2.1. Función Gamma, Función Beta, Factoriales, Coeficientes Binomiales.

Esta es la rutina que implementa la función gamma.

```
Doub gammln(const Doub xx) {
      //devolver el valor ln (gamma (xx)) para xx > 0
      Int j;
      Doub x,tmp,y,ser;
      static const Doub cof[14]={57.1562356658629235,-59.5979603554754912,
      14.1360979747417471,-0.491913816097620199,.339946499848118887e-4,
      .465236289270485756e-4,-.983744753048795646e-4,.158088703224912494e-3,
      -.210264441724104883e-3,.217439618115212643e-3,-.164318106536763890e-3,
      .844182239838527433e-4,-.261908384015814087e-4,.368991826595316234e-5};
      if (xx <= 0) throw("mal argumento en gammln");
      y=x=xx;
      tmp = x+5.24218750000000000;     //Racional 671/128
      tmp = (x+0.5)*log(tmp)-tmp;
      ser = 0.999999999999997092;
      for (j=0;j<14;j++) ser += cof[j]/++y;
      return tmp+log(2.5066282746310005*ser/x);
}
```

Se muestra aquí la rutina factorial, hasta el factorial de 170.

```
Doub factrl(const Int n) {
      //retorna el valor de n! como un número de punto flotante
      static VecDoub a(171);
      static Bool init=true;
      if (init) {
            init = false;
            a[0] = 1.;
            for (Int i=1;i<171;i++) a[i] = i*a[i-1];
      }
      if (n < 0 || n > 170) throw("factrl fuera del rango");
      return a[n];
}
```

Más útil en la práctica es una función que devuelve el logaritmo de un factorial, que no tiene problemas de desbordamiento. El tamaño de la tabla de logaritmos es lo que usted puede permitirse el lujo en el espacio y el tiempo de inicialización. El valor NTOP = 2000 se debe aumentar si sus argumentos enteros son a menudo más grandes.

```
Doub factln(const Int n) {
      //Retorna ln(n!)
```

```
	static const Int NTOP=2000;
	static VecDoub a(NTOP);
	static Bool init=true;
	if (init) {
		init = false;
		for (Int i=0;i<NTOP;i++) a[i] = gammln(i+1.);
	}
	if (n < 0) throw("negative arg in factln");
	if (n < NTOP) return a[n];
	return gammln(n+1.);
}
```

Una rutina que se aprovecha de las tablas almacenadas en factrl y factln es bico.

Y aquí está la rutina que calcula el coeficiente binomial.

```
Doub bico(const Int n, const Int k) {
	//Retorna el coeficiente binomial (n k) como un número de punto flotante
	if (n<0 || k<0 || k>n) throw("bad args in bico");
	if (n<171) return floor(0.5+factrl(n)/(factrl(k)*factrl(n-k)));
	return floor(0.5+exp(factln(n)-factln(k)-factln(n-k)));
}
```

Por último, apartándose de las funciones combinatorias con argumentos de valores enteros, llegamos a la función beta,

```
Doub beta(const Doub z, const Doub w) {
	//Retorna el valor de la función beta B(z,w)
	return exp(gammln(z)+gammln(w)-gammln(z+w));
}
```

3.2.2. La Función Gamma Incompleta y la Función de Error

Antes de codificar estas funciones, tenemos esta estructura (clase)

```
struct Gauleg18 {
	/*	Abscisas y sus pesos para la cuadratura de Gauss - Legendre
	*/
	static const Int ngau = 18;
	static const Doub y[18];
	static const Doub w[18];
};
```

```
const Doub Gauleg18::y[18] = {0.0021695375159141994,
0.011413521097787704,0.027972308950302116,0.051727015600492421,
0.082502225484340941, 0.12007019910960293,0.16415283300752470,
0.21442376986779355, 0.27051082840644336, 0.33199876341447887,
0.39843234186401943, 0.46931971407375483, 0.54413605556657973,
0.62232745288031077, 0.70331500465597174, 0.78649910768313447,
0.87126389619061517, 0.95698180152629142};
const Doub Gauleg18::w[18] = {0.0055657196642445571,
0.012915947284065419,0.020181515297735382,0.027298621498568734,
0.034213810770299537,0.040875750923643261,0.047235083490265582,
0.053244713977759692,0.058860144245324798,0.064039797355015485,
0.068745323835736408,0.072941885005653087,0.076598410645870640,
0.079687828912071670,0.082187266704339706,0.084078218979661945,
0.085346685739338721,0.085983275670394821};
struct Gamma : Gauleg18 {
      /* Objeto para la función gamma incompleta. Gauleg18 proporciona
            coeficientes de la cuadratura de Gauss-Legendre
      */
      static const Int ASWITCH=100;
      static const Doub EPS;
      static const Doub FPMIN;
      Doub gln;

      Doub gammp(const Doub a, const Doub x) {
            //Retorna la función gamma incompleta P(a, x)
            if (x < 0.0 || a <= 0.0) throw("bad args in gammp");
            if (x == 0.0) return 0.0;
            else if ((Int)a >= ASWITCH) return gammpapprox(a,x,1);
            else if (x < a+1.0) return gser(a,x);
            else return 1.0-gcf(a,x);
      }

      Doub gammq(const Doub a, const Doub x) {
            //Retorna la función gamma incompleta Q(a, x) = 1. - P(a, x)
            if (x < 0.0 || a <= 0.0) throw("bad args in gammq");
            if (x == 0.0) return 1.0;
            else if ((Int)a >= ASWITCH) return gammpapprox(a,x,0);
            else if (x < a+1.0) return 1.0-gser(a,x);
            else return gcf(a,x);
      }

      Doub gser(const Doub a, const Doub x) {
            /*    Retorna la función gamma incompleta P(a, x) evalaudo por
                  la representación de serie
            */
            Doub sum,del,ap;
            gln=gammln(a);
            ap=a;
```

```
            del=sum=1.0/a;
            for (;;) {
                  ++ap;
                  del *= x/ap;
                  sum += del;
                  if (fabs(del) < fabs(sum)*EPS) {
                        return sum*exp(-x+a*log(x)-gln);
                  }
            }
      }

      Doub gcf(const Doub a, const Doub x) {
            /*    Retorna la función gamma incompleta P(a, x) evalaudo por
                  la representación de fracciones continuas
            */
            Int i;
            Doub an,b,c,d,del,h;
            gln=gammln(a);
            b=x+1.0-a;
            c=1.0/FPMIN;
            d=1.0/b;
            h=d;
            for (i=1;;i++) {
                  an = -i*(i-a);
                  b += 2.0;
                  d=an*d+b;
                  if (fabs(d) < FPMIN) d=FPMIN;
                  c=b+an/c;
                  if (fabs(c) < FPMIN) c=FPMIN;
                  d=1.0/d;
                  del=d*c;
                  h *= del;
                  if (fabs(del-1.0) <= EPS) break;
            }
            return exp(-x+a*log(x)-gln)*h;
      }
Doub gammpapprox(Doub a, Doub x, Int psig) {
            /*    Gamma incompleta por cuadratura. devuelve P (a, x) o
                  Q (a, x), cuando psig es 1 o 0, respectivamente. El usuario
                  no debe llamar directamente
            */
            Int j;
            Doub xu,t,sum,ans;
            Doub a1 = a-1.0, lna1 = log(a1), sqrta1 = sqrt(a1);
            gln = gammln(a);
            if (x > a1) xu = MAX(a1 + 11.5*sqrta1, x + 6.0*sqrta1);
            else xu = MAX(0.,MIN(a1 - 7.5*sqrta1, x - 5.0*sqrta1));
            sum = 0;
```

```
            for (j=0;j<ngau;j++) {
                  t = x + (xu-x)*y[j];
                  sum += w[j]*exp(-(t-a1)+a1*(log(t)-lna1));
            }
            ans = sum*(xu-x)*exp(a1*(lna1-1.)-gln);
            return (psig?(x>a1? 1.0-ans:-ans):(x>a1? ans:1.0+ans));
      }

      Doub invgammp(Doub p, Doub a);

};
const Doub Gamma::EPS = numeric_limits<Doub>::epsilon();
const Doub Gamma::FPMIN = numeric_limits<Doub>::min()/EPS;
```

3.2.3. Inversa de la Función Gamma Incompleta

Una implementación es:

```
Doub Gamma::invgammp(Doub p, Doub a) {
      //Retorna x tal que P(a, x) = p para un argumento p entre 0 y 1
      Int j;
      Doub x,err,t,u,pp,lna1,afac,a1=a-1;
      const Doub EPS=1.e-8;
      gln=gammln(a);
      if (a <= 0.) throw("a must be pos in invgammap");
      if (p >= 1.) return MAX(100.,a + 100.*sqrt(a));
      if (p <= 0.) return 0.0;
      if (a > 1.) {
            lna1=log(a1);
            afac = exp(a1*(lna1-1.)-gln);
            pp = (p < 0.5)? p : 1. - p;
            t = sqrt(-2.*log(pp));
            x = (2.30753+t*0.27061)/(1.+t*(0.99229+t*0.04481)) - t;
            if (p < 0.5) x = -x;
            x = MAX(1.e-3,a*pow(1.-1./(9.*a)-x/(3.*sqrt(a)),3));
      } else {
            t = 1.0 - a*(0.253+a*0.12);
            if (p < t) x = pow(p/t,1./a);
            else x = 1.-log(1.-(p-t)/(1.-t));
      }
      for (j=0;j<12;j++) {
            if (x <= 0.0) return 0.0;
            err = gammp(a,x) - p;
            if (a > 1.) t = afac*exp(-(x-a1)+a1*(log(x)-lna1));
            else t = exp(-x+a1*log(x)-gln);
            u = err/t;
```

```
            x -= (t = u/(1.-0.5*MIN(1.,u*((a-1.)/x - 1))));
            if (x <= 0.) x = 0.5*(x + t);
            if (fabs(t) < EPS*x ) break;
        }
        return x;
    }
```

3.2.4. Función de Error

La función de error y la función de error complementaria son casos especiales de la función gamma incompleta y se obtienen moderadamente eficientemente por los procedimientos anteriores.

```
struct Erf {
        //Objeto de la función de error y funciones relacionadas
        static const Int ncof=28;
        static const Doub cof[28];

        inline Doub erf(Doub x) {
            if (x >=0.) return 1.0 - erfccheb(x);
            else return erfccheb(-x) - 1.0;
        }

        inline Doub erfc(Doub x) {
            if (x >= 0.) return erfccheb(x);
            else return 2.0 - erfccheb(-x);
        }

        Doub erfccheb(Doub z){
            /* Evaluar la ecuación (2.2.5.7) utilizando los coeficientes de
                Chebyshev almacenados. El usuario no debe llamar
                directamente
            */
            Int j;
            Doub t,ty,tmp,d=0.,dd=0.;
            if (z < 0.) throw("erfccheb requires nonnegative argument");
            t = 2./(2.+z);
            ty = 4.*t - 2.;
            for (j=ncof-1;j>0;j--) {
                tmp = d;
                d = ty*d - dd + cof[j];
                dd = tmp;
            }
            return t*exp(-z*z + 0.5*(cof[0] + ty*d) - dd);
        }

        Doub inverfc(Doub p) {
            /* Inverso de la función de error complementaria.
                Devuelve x tal que erfc (x) = p para el argumento p entre 0
                y 2
            */
            Doub x,err,t,pp;
            if (p >= 2.0) return -100.;
            if (p <= 0.0) return 100.;
            pp = (p < 1.0)? p : 2. - p;
```

```
            t = sqrt(-2.*log(pp/2.));
            x = -0.70711*((2.30753+t*0.27061)/(1.+t*(0.99229+t*0.04481)) -
t);
            for (Int j=0;j<2;j++) {
                  err = erfc(x) - pp;
                  x += err/(1.12837916709551257*exp(-SQR(x))-x*err);
            }
            return (p < 1.0? x : -x);
      }

      inline Doub inverf(Doub p) {return inverfc(1.-p);}
      /*    Inverso de la función de error complementaria. Volver x tal que
            erfc (x) = p para el argumento p entre -1 y 1
      */

};

const Doub Erf::cof[28] = {-1.3026537197817094, 6.4196979235649026e-1,
      1.9476473204185836e-2,-9.561514786808631e-3,-9.46595344482036e-4,
      3.66839497852761e-4,4.2523324806907e-5,-2.0278578112534e-5,
      -1.624290004647e-6,1.303655835580e-6,1.5626441722e-8,-8.5238095915e-8,
      6.529054439e-9,5.059343495e-9,-9.91364156e-10,-2.27365122e-10,
      9.6467911e-11, 2.394038e-12,-6.886027e-12,8.94487e-13, 3.13092e-13,
      -1.12708e-13,3.81e-16,7.106e-15,-1.523e-15,-9.4e-17,1.21e-16,-2.8e-17};
Doub erfcc(const Doub x)
{
      /*    Devuelve la  función de error complementaria erfc(x) con el error
            fraccional en todas partes menor de 1.2e-7
      */
      Doub t,z=fabs(x),ans;
      t=2./(2.+z);
      ans=t*exp(-z*z-1.26551223+t*(1.00002368+t*(0.37409196+t*(0.09678418+
            t*(-0.18628806+t*(0.27886807+t*(-1.13520398+t*(1.48851587+
            t*(-0.82215223+t*0.17087277)))))))));
      return (x >= 0.0 ? ans : 2.0-ans);
}
```

3.2.5. Función Beta Incompleta

```
struct Beta : Gauleg18 {
      /*    Objeto para la función beta incompleta. Gauleg18 provee los
            coeficientes para la cuadratura de Gauss-Legendre
      */
      static const Int SWITCH=3000;
      static const Doub EPS, FPMIN;

      Doub betai(const Doub a, const Doub b, const Doub x) {
            /*    Retorna la función beta incompleta I_x(a,B) para a y b
                  positivos y x entre 0 y 1
            */
            Doub bt;
            if (a <= 0.0 || b <= 0.0) throw("Bad a or b in routine betai");
            if (x < 0.0 || x > 1.0) throw("Bad x in routine betai");
```

```
	if (x == 0.0 || x == 1.0) return x;
	if (a > SWITCH && b > SWITCH) return betaiapprox(a,b,x);
	bt=exp(gammln(a+b)-gammln(a)-gammln(b)+a*log(x)+b*log(1.0-x));
	if (x < (a+1.0)/(a+b+2.0)) return bt*betacf(a,b,x)/a;
	else return 1.0-bt*betacf(b,a,1.0-x)/b;
}

Doub betacf(const Doub a, const Doub b, const Doub x) {
	/*	Evalua la fracción continua para la función beta incompleta
		por el método modificado de Lentz. El usuario no puede llamar
		directamengte
	*/
	Int m,m2;
	Doub aa,c,d,del,h,qab,qam,qap;
	qab=a+b;
	qap=a+1.0;
	qam=a-1.0;
	c=1.0;
	d=1.0-qab*x/qap;
	if (fabs(d) < FPMIN) d=FPMIN;
	d=1.0/d;
	h=d;
	for (m=1;m<10000;m++) {
		m2=2*m;
		aa=m*(b-m)*x/((qam+m2)*(a+m2));
		d=1.0+aa*d;
		if (fabs(d) < FPMIN) d=FPMIN;
		c=1.0+aa/c;
		if (fabs(c) < FPMIN) c=FPMIN;
		d=1.0/d;
		h *= d*c;
		aa = -(a+m)*(qab+m)*x/((a+m2)*(qap+m2));
		d=1.0+aa*d;
		if (fabs(d) < FPMIN) d=FPMIN;
		c=1.0+aa/c;
		if (fabs(c) < FPMIN) c=FPMIN;
		d=1.0/d;
		del=d*c;
		h *= del;
		if (fabs(del-1.0) <= EPS) break;
	}
	return h;
}

Doub betaiapprox(Doub a, Doub b, Doub x) {
	/*	Beta incompleta por cuadratura. Retorna I_x(a,b). El usuario
		no puede llamar directamente
	*/
```

```
            Int j;
            Doub xu,t,sum,ans;
            Doub a1 = a-1.0, b1 = b-1.0, mu = a/(a+b);
            Doub lnmu=log(mu),lnmuc=log(1.-mu);
            t = sqrt(a*b/(SQR(a+b)*(a+b+1.0)));
            if (x > a/(a+b)) {
                  if (x >= 1.0) return 1.0;
                  xu = MIN(1.,MAX(mu + 10.*t, x + 5.0*t));
            } else {
                  if (x <= 0.0) return 0.0;
                  xu = MAX(0.,MIN(mu - 10.*t, x - 5.0*t));
            }
            sum = 0;
            for (j=0;j<18;j++) {
                  t = x + (xu-x)*y[j];
                  sum += w[j]*exp(a1*(log(t)-lnmu)+b1*(log(1-t)-lnmuc));
            }
            ans          =          sum*(xu-x)*exp(a1*lnmu-gammln(a)+b1*lnmuc-
gammln(b)+gammln(a+b));
            return ans>0.0? 1.0-ans : -ans;
      }

      Doub invbetai(Doub p, Doub a, Doub b) {
            /*    La inversa de la función beta incompleta. Retorna x tal que
                  I_x(a,b) = p para el argumento p entre 0 y 1
            */
            const Doub EPS = 1.e-8;
            Doub pp,t,u,err,x,al,h,w,afac,a1=a-1.,b1=b-1.;
            Int j;
            if (p <= 0.) return 0.;
            else if (p >= 1.) return 1.;
            else if (a >= 1. && b >= 1.) {
                  pp = (p < 0.5)? p : 1. - p;
                  t = sqrt(-2.*log(pp));
                  x = (2.30753+t*0.27061)/(1.+t*(0.99229+t*0.04481)) - t;
                  if (p < 0.5) x = -x;
                  al = (SQR(x)-3.)/6.;
                  h = 2./(1./(2.*a-1.)+1./(2.*b-1.));
                  w  =  (x*sqrt(al+h)/h)-(1./(2.*b-1)-1./(2.*a-1.))*(al+5./6.-
2./(3.*h));
                  x = a/(a+b*exp(2.*w));
            } else {
                  Doub lna = log(a/(a+b)), lnb = log(b/(a+b));
                  t = exp(a*lna)/a;
                  u = exp(b*lnb)/b;
                  w = t + u;
                  if (p < t/w) x = pow(a*w*p,1./a);
                  else x = 1. - pow(b*w*(1.-p),1./b);
```

```
            }
            afac = -gammln(a)-gammln(b)+gammln(a+b);
            for (j=0;j<10;j++) {
                  if (x == 0. || x == 1.) return x;
                  err = betai(a,b,x) - p;
                  t = exp(a1*log(x)+b1*log(1.-x) + afac);
                  u = err/t;
                  x -= (t = u/(1.-0.5*MIN(1.,u*(a1/x - b1/(1.-x)))));
                  if (x <= 0.) x = 0.5*(x + t);
                  if (x >= 1.) x = 0.5*(x + t + 1.);
                  if (fabs(t) < EPS*x && j > 0) break;
            }
            return x;
      }

};
const Doub Beta::EPS = numeric_limits<Doub>::epsilon();
const Doub Beta::FPMIN = numeric_limits<Doub>::min()/EPS;
```

3.2.6. Funciones estadísticas

Mostramos en este párrafo el código fuente en C++ de las funciones estadísticas.

3.2.6.1. Distribución Normal (o de Gauss)

La siguiente estructura implementa las relaciones anteriores.

```
struct Normaldist : Erf {
//Distribución Normal, se deriva de la función de error Erf

      Doub mu, sig;
      Normaldist(Doub mmu = 0., Doub ssig = 1.) : mu(mmu), sig(ssig) {
      /* Constructor. Inicializado con mu y sig. El constructor por defecto
            sin argumentos is N(0,1)
      */
            if (sig <= 0.) throw("bad sig in Normaldist");
      }
      Doub p(Doub x) {
            //Retorna la función de densidad de probabilidad
            return (0.39894228040143268/sig)*exp(-0.5*SQR((x-mu)/sig));
      }
      Doub cdf(Doub x) {
            //Retorna la función de distribución acumulativa
            return 0.5*erfc(-0.70710678118654752*(x-mu)/sig);
      }
      Doub invcdf(Doub p) {
        //Retorna la inversa de la función de distribución acumulativa
            if (p <= 0. || p >= 1.) throw("bad p in Normaldist");
            return -1.41421356237309505*sig*inverfc(2.*p)+mu;
      }
};
```

3.2.6.2. Distribución de Cauchy

La distribución de Cauchy a veces se llama la distribución de Lorentz.

```
struct Cauchydist {
	//Distribución de Cauchy
	Doub mu, sig;
	Cauchydist(Doub mmu = 0., Doub ssig = 1.) : mu(mmu), sig(ssig) {
		/*	Constructor. Inicializado con mu y sig. El constructor
			por defecto tiene los argumentos Cauchy(0, 1)
		*/
		if (sig <= 0.) throw("bad sig in Cauchydist");
	}
	Doub p(Doub x) {
		//Retorna la función de densidad de probabilidad
		return 0.318309886183790671/(sig*(1.+SQR((x-mu)/sig)));
	}
	Doub cdf(Doub x) {
		//Retorna la función de distribución acumulativa
		return 0.5+0.318309886183790671*atan2(x-mu,sig);
	}
	Doub invcdf(Doub p) {
		//Retorna la inversa de la función de distribución
acumulativa
		if (p <= 0. || p >= 1.) throw("bad p in Cauchydist");
		return mu + sig*tan(3.14159265358979324*(p-0.5));
	}
};
```

3.2.6.3. Distribución t de Student

```
struct Studenttdist : Beta {
	//Distribución t de Student, derivado de la función beta Beta
	Doub nu, mu, sig, np, fac;
	Studenttdist(Doub nnu, Doub mmu = 0., Doub ssig = 1.)
	: nu(nnu), mu(mmu), sig(ssig) {
		//Constructor. Inicializado con nu, mu y sig. Por defecto con
		// un argumento es Student(nu, 0, 1)
		if (sig <= 0. || nu <= 0.) throw("bad sig,nu in Studentdist");
		np = 0.5*(nu + 1.);
		fac = gammln(np)-gammln(0.5*nu);
	}
	Doub p(Doub t) {
		//Retorna la función de densidad de probabilidad
		return exp(-np*log(1.+SQR((t-mu)/sig)/nu)+fac)
			/(sqrt(3.14159265358979324*nu)*sig);
	}
	Doub cdf(Doub t) {
		//Retorna la función de distribución acumulativa
```

```
            Doub p = 0.5*betai(0.5*nu, 0.5, nu/(nu+SQR((t-mu)/sig)));
            if (t >= mu) return 1. - p;
            else return p;
        }
        Doub invcdf(Doub p) {
            //Retorna la inversa de la función de distribución
            //acumulativa
            if (p <= 0. || p >= 1.) throw("bad p in Studentdist");
            Doub x = invbetai(2.*MIN(p,1.-p), 0.5*nu, 0.5);
            x = sig*sqrt(nu*(1.-x)/x);
            return (p >= 0.5? mu+x : mu-x);
        }
        Doub aa(Doub t) {
            //Retorna la cdf de dos colas A(t|v)
            if (t < 0.) throw("bad t in Studentdist");
            return 1.-betai(0.5*nu, 0.5, nu/(nu+SQR(t)));
        }
        Doub invaa(Doub p) {
            //Retorna la inversa, es decir t tal que p = A(t|v)
            if (p < 0. || p >= 1.) throw("bad p in Studentdist");
            Doub x = invbetai(1.-p, 0.5*nu, 0.5);
            return sqrt(nu*(1.-x)/x);
        }
};
```

3.2.6.4. La Distribución Chi – Cuadrado

Codificación de la distribución chi-cuadrado (o χ^2) tiene un solo parámetro $\nu > 0$ que controla tanto la ubicación y ancho de su pico. En la mayoría de aplicaciones ν es un número entero y se refiere como el *número de grados de libertad*.

```
struct Chisqdist : Gamma {
    /*    Distribución chi-cuadrado, derivado de la función gamma Gamma
    */
    Doub nu,fac;
    Chisqdist(Doub nnu) : nu(nnu) {
        //Constructor. Se inicializa con ν
        if (nu <= 0.) throw("bad nu in Chisqdist");
        fac = 0.693147180559945309*(0.5*nu)+gammln(0.5*nu);
    }
    Doub p(Doub x2) {
      //Retorna la función de densidad de probabilidad
        if (x2 <= 0.) throw("bad x2 in Chisqdist");
        return exp(-0.5*(x2-(nu-2.)*log(x2))-fac);
    }
    Doub cdf(Doub x2) {
        //Retorna la función de distribución acumulativa
        if (x2 < 0.) throw("bad x2 in Chisqdist");
        return gammp(0.5*nu,0.5*x2);
    }
    Doub invcdf(Doub p) {
      //Retorna la inversa de la función de distribución acumulativa
```

```
            if (p < 0. || p >= 1.) throw("bad p in Chisqdist");
            return 2.*invgammp(p,0.5*nu);
      }
};
```

3.2.6.5. Distribución Gamma

```
struct Gammadist : Gamma {
      //Distribución gamma, derivado de la función gamma Gamma
      Doub alph, bet, fac;
      Gammadist(Doub aalph, Doub bbet = 1.) : alph(aalph), bet(bbet) {
            if (alph <= 0. || bet <= 0.) throw("bad alph,bet in Gammadist");
            fac = alph*log(bet)-gammln(alph);
      }
      Doub p(Doub x) {
            //Retorna la función de densidad de probabilidad
            if (x <= 0.) throw("bad x in Gammadist");
            return exp(-bet*x+(alph-1.)*log(x)+fac);
      }
      Doub cdf(Doub x) {
            //Retorna la función de distribución acumulativa
            if (x < 0.) throw("bad x in Gammadist");
            return gammp(alph,bet*x);
      }
      Doub invcdf(Doub p) {
            //Retorna la inversa de la función de distribución acumulativa
            if (p < 0. || p >= 1.) throw("bad p in Gammadist");
            return invgammp(p,alph)/bet;
      }
};
```

3.2.6.6. Distribución – F

Un uso frecuente de la distribución F es para probar si dos muestras observadas tienen la misma varianza

```
struct Fdist : Beta {
      //Distribución F, derivada de la función beta Beta
      Doub nu1,nu2;
      Doub fac;
      Fdist(Doub nnu1, Doub nnu2) : nu1(nnu1), nu2(nnu2) {
            //Constructor. Inicializado con v_1 y v_2
            if (nu1 <= 0. || nu2 <= 0.) throw("bad nu1,nu2 in Fdist");
            fac = 0.5*(nu1*log(nu1)+nu2*log(nu2))+gammln(0.5*(nu1+nu2))
                  -gammln(0.5*nu1)-gammln(0.5*nu2);
      }
      Doub p(Doub f) {
            //Retorna la función de densidad de probabilidad
            if (f <= 0.) throw("bad f in Fdist");
            return exp((0.5*nu1-1.)*log(f)-0.5*(nu1+nu2)*log(nu2+nu1*f)+fac);
      }
      Doub cdf(Doub f) {
            //Retorna la función de distribución acumulativa
```

```
            if (f < 0.) throw("bad f in Fdist");
            return betai(0.5*nu1,0.5*nu2,nu1*f/(nu2+nu1*f));
      }
      Doub invcdf(Doub p) {
            //Retorna la inversa de la función de distribución acumulativa
            if (p <= 0. || p >= 1.) throw("bad p in Fdist");
            Doub x = invbetai(p,0.5*nu1,0.5*nu2);
            return nu2*x/(nu1*(1.-x));
      }
};
```

3.2.6.7. Distribución Beta

```
struct Betadist : Beta {
      //Distribución Beta, derivado de la función beta Beta.
      Doub alph, bet, fac;
      Betadist(Doub aalph, Doub bbet) : alph(aalph), bet(bbet) {
            //Constructor. Inicializado con alpha y beta
            if (alph <= 0. || bet <= 0.) throw("bad alph,bet in Betadist");
            fac = gammln(alph+bet)-gammln(alph)-gammln(bet);
      }
      Doub p(Doub x) {
            //Retorna la función de densidad de probabilidad
            if (x <= 0. || x >= 1.) throw("bad x in Betadist");
            return exp((alph-1.)*log(x)+(bet-1.)*log(1.-x)+fac);
      }
      Doub cdf(Doub x) {
            //Retorna la función de distribución acumulativa
            if (x < 0. || x > 1.) throw("bad x in Betadist");
            return betai(alph,bet,x);
      }
      Doub invcdf(Doub p) {
            //Retorna la inversa de la función de distribución acumulativa
            if (p < 0. || p > 1.) throw("bad p in Betadist");
            return invbetai(p,alph,bet);
      }
};
```

3.2.6.8. La Distribución de Kolmogorov – Smirnov

El código para las funciones KS y sus inversas son:

```
struct KSdist {
      /* Función de Distribución acumulativa de Kolgomorov - Smirnov y sus
         inversas
      */
      Doub pks(Doub z) {
            //Retorna la función de distribución acumulativa
            if (z < 0.) throw("bad z in KSdist");
            if (z < 0.042) return 0.;
            if (z < 1.18) {
                  Doub y = exp(-1.23370055013616983/SQR(z));
```

```
                return 2.25675833419102515*sqrt(-log(y))
                        *(y + pow(y,9) + pow(y,25) + pow(y,49));
        } else {
                Doub x = exp(-2.*SQR(z));
                return 1. - 2.*(x - pow(x,4) + pow(x,9));
        }
    }
    Doub qks(Doub z) {
        //Retorna la función de distribución acumulativa complementaria
        if (z < 0.) throw("bad z in KSdist");
        if (z == 0.) return 1.;
        if (z < 1.18) return 1.-pks(z);
        Doub x = exp(-2.*SQR(z));
        return 2.*(x - pow(x,4) + pow(x,9));
    }
    Doub invqks(Doub q) {
        /*    Retorna la inversa de la función de distribución acumulativa
              complementaria
        */
        Doub y,logy,yp,x,xp,f,ff,u,t;
        if (q <= 0. || q > 1.) throw("bad q in KSdist");
        if (q == 1.) return 0.;
        if (q > 0.3) {
                f = -0.3926990816987241 55*SQR(1.-q);
                y = invxlogx(f);
                do {
                        yp = y;
                        logy = log(y);
                        ff = f/SQR(1.+ pow(y,4)+ pow(y,12));
                        u = (y*logy-ff)/(1.+logy);
                        y = y - (t=u/MAX(0.5,1.-0.5*u/(y*(1.+logy))));
                } while (abs(t/y)>1.e-15);
                return 1.57079632679489662/sqrt(-log(y));
        } else {
                x = 0.03;
                do {
                        xp = x;
                        x = 0.5*q+pow(x,4)-pow(x,9);
                        if (x > 0.06) x += pow(x,16)-pow(x,25);
                } while (abs((xp-x)/x)>1.e-15);
                return sqrt(-0.5*log(x));
        }
    }
    Doub invpks(Doub p) {return invqks(1.-p);}
    //Retorna la inversa de la función de distribución acumulativa
};
```

3.2.6.9. Distribución de Poisson

```
struct Poissondist : Gamma {
	//Distribución de Poisson, derivado de la función gamma Gamma
	Doub lam;
	Poissondist(Doub llam) : lam(llam) {
		//Constructor. Inicializado con lambda
		if (lam <= 0.) throw("bad lam in Poissondist");
	}
	Doub p(Int n) {
		//Retorna la función de densidad de probabilidad
		if (n < 0) throw("bad n in Poissondist");
		return exp(-lam + n*log(lam) - gammln(n+1.));
	}
	Doub cdf(Int n) {
		//Retorna la función de distribución acumulativa
		if (n < 0) throw("bad n in Poissondist");
		if (n == 0) return 0.;
		return gammq((Doub)n,lam);
	}
	Int invcdf(Doub p) {
		/* Dando el argumento P, retorna el entero n, tal que
			P(<n) <= P <= P(n+1)
		*/
		Int n,nl,nu,inc=1;
		if (p <= 0. || p >= 1.) throw("bad p in Poissondist");
		if (p < exp(-lam)) return 0;
		n = (Int)MAX(sqrt(lam),5.);
		if (p < cdf(n)) {
			do {
				n = MAX(n-inc,0);
				inc *= 2;
			} while (p < cdf(n));
			nl = n; nu = n + inc/2;
		} else {
			do {
				n += inc;
				inc *= 2;
			} while (p > cdf(n));
			nu = n; nl = n - inc/2;
		}
		while (nu-nl>1) {
			n = (nl+nu)/2;
			if (p < cdf(n)) nu = n;
			else nl = n;
		}
		return nl;
```

```
	}
};
```

3.2.6.10. Distribución Binomial

```
struct Binomialdist : Beta {
	//Distribución binomial, derivada de la función beta Beta
	Int n;
	Doub pe, fac;
	Binomialdist(Int nn, Doub ppe) : n(nn), pe(ppe) {
		/* Constructor. Inicializado con n (tamaño de la muestra) y
		   p (probabilidad del evento)
		*/
		if (n <= 0 || pe <= 0. || pe >= 1.) throw("bad args in
Binomialdist");
		fac = gammln(n+1.);
	}
	Doub p(Int k) {
		//Retorna la función de densidad de probabilidad
		if (k < 0) throw("bad k in Binomialdist");
		if (k > n) return 0.;
		return exp(k*log(pe)+(n-k)*log(1.-pe)
			+fac-gammln(k+1.)-gammln(n-k+1.));
	}
	Doub cdf(Int k) {
		//Retorna la función de distribución acumulativa
		if (k < 0) throw("bad k in Binomialdist");
		if (k == 0) return 0.;
		if (k > n) return 1.;
		return 1. - betai((Doub)k,n-k+1.,pe);
	}
	Int invcdf(Doub p) {
		/* Dado el argumento P, retorna el entero n tal que
		P( < n) <= P <= P( < n + 1)
		*/
		Int k,kl,ku,inc=1;
		if (p <= 0. || p >= 1.) throw("bad p in Binomialdist");
		k = MAX(0,MIN(n,(Int)(n*pe)));
		if (p < cdf(k)) {
			do {
				k = MAX(k-inc,0);
				inc *= 2;
			} while (p < cdf(k));
			kl = k; ku = k + inc/2;
		} else {
```

```
                do {
                        k = MIN(k+inc,n+1);
                        inc *= 2;
                } while (p > cdf(k));
                ku = k; kl = k - inc/2;
        }
        while (ku-kl>1) {
                k = (kl+ku)/2;
                if (p < cdf(k)) ku = k;
                else kl = k;
        }
        return kl;
    }
};
```

3.3. DESCRIPCIÓN ESTADÍSTICA DE LOS DATOS

3.3.1. Momentos de una Distribución: Media, Varianza, Asimetría y así sucesivamente

```
void moment(VecDoub_I &data, Doub &ave, Doub &adev, Doub &sdev, Doub &var,
      Doub &skew, Doub &curt) {
      /** Dada una matriz de datos data[0, ... n-1], esta rutina devuelve su
            media o promedio aritmético ave, desviación media o desviación media
            absoluta adev, desviación estándar sdev, varianza var,asimetría o
            sesgo skew y curtosis curt.
      */
      Int j,n=data.size();
      Doub ep=0.0,s,p;
      if (n <= 1) throw("n must be at least 2 in moment");
      s=0.0;
      for (j=0;j<n;j++) s += data[j];
      ave=s/n;
      adev=var=skew=curt=0.0;
      for (j=0;j<n;j++) {
            adev += abs(s=data[j]-ave);
            ep += s;
            var += (p=s*s);
            skew += (p *= s);
            curt += (p *= s);
      }
      adev /= n;
      var=(var-ep*ep/n)/(n-1);
      sdev=sqrt(var);
      if (var != 0.0) {
            skew /= (n*var*sdev);
            curt=curt/(n*var*var)-3.0;
      } else throw("No skew/kurtosis when variance = 0 (in moment)");
}
```

3.3.2. Determinación de las características de velocidad a partir de un conjunto de datos de velocidad. Programa de Prueba.

Se muestra aquí el programa que procesa un conjunto de datos de velocidad par obtener sus características más relevantes. El problema está dado como un ejemplo en el libro de “Ingeniería de Tránsito y Carreteras” de Nicholas J. Garber y Lester A. Joel. Ejemplo 4-2, página 87.

```
#include "C:\nr_c304\code\nr3.h"
```

```
#include "C:\nr_c304\code\moment.h"
#include "C:\nr_c304\code\sort.h"
#include "C:\nr_c304\code\interp_1d.h"
#include "C:\nr_c304\code\interp_linear.h"

#include <iostream>
#include <conio.h>
#include <iomanip>
#include <map>
#include <vector>
#include <algorithm>

void imprime(const pair<const double, int>& r);
void moda(std::map <const double, int>& r, double& veloc, int& frec);

int main()
{
	/** Un conjunto de datos de velocidad en el sitio, separados por comas,
		representado como un arreglo de tipos double. Se supone en
		millas / hora
	*/
	double a[] = {	35.1,  44.0,  45.8,  44.3,  36.3,  54.0,  42.1,  50.1,
			51.8,  50.8,  38.3,  44.6,  45.2,  41.1,  55.1,  50.2,
			54.3,  45.4,  55.2,  45.7,  54.1,  54.0,  46.1,  54.2,
			52.3,  57.3,  46.8,  57.8,  36.8,  55.8,  43.3,  55.3,
			39.0,  53.7,  40.8,  54.5,  51.6,  51.7,  50.3,  59.8,
			40.3,  55.1,  45.0,  48.3,  47.8,  47.1,  34.8,  52.4,
			49.1,  37.1,  65.0,  49.5,  52.2,  48.4,  42.8,  49.5,
			48.6,  41.2,  48.0,  58.0,  49.0,  41.8,  48.3,  45.9,
			44.7,  49.5,  56.0,  49.1,  49.2,  56.4,  48.5,  45.4,
			48.6,  52.0,  49.8,  63.4,  60.1,  48.8,  52.1,  48.7,
			61.8,  56.6,  48.2,  62.1,  53.3,  53.4
		};

	const int numDatos = sizeof(a) / sizeof(a[0]);

	VecDoub_I DatosVelocidad(numDatos, a);
	double ave;
	double adev;
	double sdev;
	double var;
	double skew;
	double curt;

	/** Dada una matriz de datos de velocidad DatosVelocidad[0, ... n-1], esta
		rutina devuelve su media o promedio aritmético ave, desviación media o
		desviación media absoluta adev, desviación estándar sdev, varianza var,
		asimetría o sesgo skew y curtosis curt.
	*/
	moment(DatosVelocidad, ave, adev, sdev, var, skew, curt);

	std::cout << setprecision(2);
	std::cout << resetiosflags(ios_base::fixed);
	std::cout << fixed << setw(6);
```

```
    std::cout << "Número de datos de velocidad:\t " << numDatos << std::endl;
      std::cout << "Velocidad Media:\t\t " << ave << " millas/hora" << std::endl;
      std::cout << "Desviación Media Absoluta:\t " << adev << " millas/hora" <<
std::endl;
      std::cout << "Variancia:\t\t\t " << var << " millas²/hora²" << std::endl;
      std::cout << "Desviación Estándar:\t\t " << sdev << " millas/hora" << std::endl;
      std::cout << "Asimetría:\t\t\t " << skew << std::endl;
      std::cout << "Curtosis:\t\t\t " << curt << std::endl;

      /** Cálculo de la velocidad mediana y otros quantiles, se necesita ordenar
            los datos de las velocidades para hallar estos estadísticos.
      */

      VecDoub DatosOrdenados(numDatos, a);

      /** La rutina shell ordena un array aa[0, ..., n-1] en orden numérico
            ascendente por el método de Shell. aa se sustituye en
            la salida por su reordenación ordenada. Normalmente, el argumento
            opcional m debe ser omitido, pero si se establece en un valor positivo,
            entonces sólo los primeros m elementos de aa serán ordenados
      */
      shell(DatosOrdenados, -1);

      /** Se calcula la velocidad mediana con los datos de las velocidades
            previamente ordenadas
      */
      double velocidadMediana;
      //Si el número de datos es impar,
      if(numDatos % 2) {
            //entonces
            velocidadMediana = DatosOrdenados[numDatos / 2] ;
      }
      else velocidadMediana = 0.5 * (DatosOrdenados[numDatos / 2 - 1] +
                                                      DatosOrdenados[numDatos /
2]);

      std::cout << "Velocidad Mediana:\t\t " << velocidadMediana
                        << " millas/hora"  << std::endl;

      /** Cálculo de la velocidad del percentil 85%, que es el que se usa en
            muchos de los estudios de la Ingeniería de Tránsito.  Se usa
            interpolación lineal para calcular dicho valor.
      */
      double velocidad85;

      VecDoub xx(numDatos), yy(numDatos);
      xx = DatosOrdenados;

      for(int i = 0; i < numDatos; i++) {
            yy[i] = i + 1;
      }

      Linear_interp myfunc(yy, xx);
```

```
	double x ;

	x = 85.0 * numDatos / 100.0;

	velocidad85 = myfunc.interp(x);

	std::cout << "Velocidad percentil 85%:\t " << velocidad85 << " millas/hora"
			<< std::endl;

	/** Cálculo de la moda, que es la velocidad que con más frecuencia se repite

	*/

	std::vector<double> veloc(numDatos);
	for(int i = 0; i < numDatos; i++)
		veloc[i] = DatosOrdenados[i];
	/** Cuenta la frecuencia de cada velocidad
	*/
	std::map<const double, int> histograma;
	for(int i = 0; i < numDatos; i++)
		histograma[veloc[i]]++;


	double velocidadModa;
	int mayorFrec;

	/*for_each(histograma.begin(), histograma.end(), imprime);

	std::cout << "Ingrese, de los resultados anteriores, la velocidad que usted"
			<< "\nconsidere la moda" << std::endl;
	std::cin >> velocidadModa;  */

	moda(histograma, velocidadModa, mayorFrec);
	std::cout << "Velocidad de Moda:\t\t " << velocidadModa << " millas/hora "
	<< "con una frecuencia de: " << mayorFrec << std::endl;

	getche();

	return 0;
}

/** Se invoca a cada elemento del map, que es una secuencia de pares, para hallar
	el valor modal
*/
void imprime(const pair<const double, int>& r) {
	std::cout << r.first << ' ' << r.second << '\n';
}

void moda(std::map<const double, int>& r, double& veloc, int& mayor) {
	mayor = 0;
	typedef std::map<const double, int>::iterator ic;
	for(ic p = r.begin(); p != r.end(); ++p) {
		if(mayor > p->second) ;
		else {
```

```
                    mayor = p->second;
                    veloc = p->first;
              }
        }

 }
```

Esta es la salida del programa.

```
Número de datos de velocidad:    86
Velocidad Media:                 49.39 millas/hora
Desviación Media Absoluta:       5.06 millas/hora
Variancia:                       42.45 millas²/hora²
Desviación Estándar:             6.52 millas/hora
Asimetría:                       -0.07
Curtosis:                        -0.22
Velocidad Mediana:               49.15 millas/hora
Velocidad percentil 85%:         55.35 millas/hora
Velocidad de Moda:               49.50 millas/hora con una frecuencia
de: 3
```

3.3.3. ¿Tienen Dos Distribuciones las Mismas Medias o Varianzas?

Desarrollamos los códigos fuentes en C++, para las pruebas estadísticas más usadas.

3.3.3.1. Prueba t de Student para Diferencias de Medias Significativas

Como una rutina, tenemos

```
void ttest(VecDoub_I &data1, VecDoub_I &data2, Doub &t, Doub &prob)
{
      /**   La rutina ttest, dado los vectores de data1[0 ... n1-1] y
            Data2[0 ... n2-1], devuelve t de Student como t, y su valor p como
            prob, valores pequeños de prob que indican que los vectores tienen
            diferentes medias significativamente. Los vectores de datos se
            supone que se elaborarán a partir de poblaciones con la misma
            varianza verdadera.
      */
      Beta beta;
      Doub var1,var2,svar,df,ave1,ave2;
      Int n1=data1.size(), n2=data2.size();
      avevar(data1,ave1,var1);
      avevar(data2,ave2,var2);
      df=n1+n2-2;
      svar=((n1-1)*var1+(n2-1)*var2)/df;
      t=(ave1-ave2)/sqrt(svar*(1.0/n1+1.0/n2));
```

```
	prob=beta.betai(0.5*df,0.5,df/(df+t*t));
}
```

que hace uso de la siguiente rutina para calcular la media y la varianza de un conjunto de números.

```
void avevar(VecDoub_I &data, Doub &ave, Doub &var) {
	/**	La rutina avevar. Dado la serie de datos data[0 ... n-1], devuelve
		su media como ave y su varianza como var
	*/
	Doub s,ep;
	Int j,n=data.size();
	ave=0.0;
	for (j=0;j<n;j++) ave += data[j];
	ave /= n;
	var=ep=0.0;
	for (j=0;j<n;j++) {
		s=data[j]-ave;
		ep += s;
		var += s*s;
	}
	var=(var-ep*ep/n)/(n-1);
}
```

El siguiente caso a considerar es que las dos distribuciones tienen significativamente diferentes varianzas, pero sin embargo se quiere saber si sus medias son las mismas o diferentes.

La rutina es

```
void tutest(VecDoub_I &data1, VecDoub_I &data2, Doub &t, Doub &prob) {
	/**	La rutina tutest, dado los vectores data1[0 ... n1-1] y
		data2[0 ... n2-1], devuelve t de Student como t, y su valor p como
		prob, valores pequeños de prob que indican que los vectores tienen
		diferentes medias significativamente. Los vectores de datos se
		supone que se elaborarán a partir de poblaciones con varianzas
		desiguales.
	*/
	Beta beta;
	Doub var1,var2,df,ave1,ave2;
	Int n1=data1.size(), n2=data2.size();
	avevar(data1,ave1,var1);
	avevar(data2,ave2,var2);
	t=(ave1-ave2)/sqrt(var1/n1+var2/n2);
	df=SQR(var1/n1+var2/n2)/(SQR(var1/n1)/(n1-1)+SQR(var2/n2)/(n2-1));
```

```
	prob=beta.betai(0.5*df,0.5,df/(df+SQR(t)));
}
```

Nuestro ejemplo final de la prueba t de Student es el caso de muestras pareadas. La rutina es

```
void tptest(VecDoub_I &data1, VecDoub_I &data2, Doub &t, Doub &prob) {
	/**	La rutina  tptest, dados los vectores pareados data1[0 ... n1-1]
		y data2[0 ... n2-1], Esta rutina devuelve la t de Student para datos
		apareados como t, y su p-valor como prob, valores pequeños de prob
		que indica una diferencia significativa de las medias.
	*/
	Beta beta;
	Int j, n=data1.size();
	Doub var1,var2,ave1,ave2,sd,df,cov=0.0;
	avevar(data1,ave1,var1);
	avevar(data2,ave2,var2);
	for (j=0;j<n;j++) cov += (data1[j]-ave1)*(data2[j]-ave2);
	cov /= (df=n-1);
	sd=sqrt((var1+var2-2.0*cov)/n);
	t=(ave1-ave2)/sd;
	prob=beta.betai(0.5*df,0.5,df/(df+t*t));
}
```

3.3.3.2. La Prueba F para Varianzas Significativamente Diferentes

Estas consideraciones y la ecuación (2.2.6.3) dan la rutina ftest.

```
void ftest(VecDoub_I &data1, VecDoub_I &data2, Doub &f, Doub &prob) {
	/**	La rutina ftest, dadas las matrices data1[0 ... n1-1] y
		 data2[0 ... n2-1], esta rutina devuelve el valor de f, y su valor
		p como prob. Los valores pequeños de prob indican que las dos
		matrices tienen significativamente diferentes varianzas.
	*/
	Beta beta;
	Doub var1,var2,ave1,ave2,df1,df2;
	Int n1=data1.size(), n2=data2.size();
	avevar(data1,ave1,var1);
	avevar(data2,ave2,var2);
	if (var1 > var2) {
		f=var1/var2;
		df1=n1-1;
		df2=n2-1;
	} else {
		f=var2/var1;
		df1=n2-1;
		df2=n1-1;
```

```
        }
        prob = 2.0*beta.betai(0.5*df2,0.5*df1,df2/(df2+df1*f));
        if (prob > 1.0) prob=2.-prob;
}
```

3.3.4. ¿Son Dos Distribuciones Diferentes?

Se generaliza la pregunta anterior sobre las medias y varianzas, y se trata ahora de comparar dos distribuciones o muestras de datos.

3.3.4.1. La Prueba Chi – Cuadrado

Tenemos, entonces, el siguiente programa:

```
void chsone(VecDoub_I &bins, VecDoub_I &ebins, Doub &df,
    /** La rutina chsone; dada la matriz bins [0 ... nbins-1] que contienen los
      números observados de eventos, y una matriz ebins [0 ... nbins-1] que
      contiene los números esperados de los acontecimientos, y dado el número
      de restricciones knstrn (normalmente uno ), esta rutina retorna (trivial)
      el número de grados de libertad df, y (no trivial) la chsq chi-cuadrado
      y el valor de p prob . Un valor pequeño de prob indica una diferencia
      significativa entre las distribuciones bins y ebins. Tenga en cuenta que
      bins y ebins son ambos matrices (vectores) de números doubles, aunque
      bins normalmente contendrá datos enteros.
      */
      Doub &chsq, Doub &prob, const Int knstrn=1) {
      Gamma gam;
      Int j,nbins=bins.size();
      Doub temp;
      df=nbins-knstrn;
      chsq=0.0;
      for (j=0;j<nbins;j++) {
            if (ebins[j]<0.0 || (ebins[j]==0. && bins[j]>0.))
                  throw("Bad expected number in chsone");
            if (ebins[j]==0.0 && bins[j]==0.0) {
                  --df;
            } else {
                  temp=bins[j]-ebins[j];
                  chsq += temp*temp/ebins[j];
            }
      }
      prob=gam.gammq(0.5*df,0.5*chsq);
}
```

Seguidamente consideramos el caso de la comparación de dos conjuntos de datos agrupados.

El programa es chstwo.

```
void chstwo(VecDoub_I &bins1, VecDoub_I &bins2, Doub &df,
      /** Dadas las matrices bins1[0 ... nbins1-1] y bins2[0 ... nbins2-1],
            Que contiene dos conjuntos de datos agrupados, y dado el número de
            restricciones knstrn (normalmente 1 o 0), esta rutina devuelve el
            número de grados de libertad df, el chi-cuadrado chsq , y el p-
            valor pro. Un valor pequeño de prob indica una diferencia
            significativa entre las distribuciones bins1 y bins2. Tenga en
            cuenta que bins1 y bins2 son ambos matrices de doubles, aunque
            normalmente contendrán valores enteros.
      */
      Doub &chsq, Doub &prob, const Int knstrn=1) {
      Gamma gam;
      Int j,nbins=bins1.size();
      Doub temp;
      df=nbins-knstrn;
      chsq=0.0;
      for (j=0;j<nbins;j++)
            if (bins1[j] == 0.0 && bins2[j] == 0.0)
                  --df;
            else {
                  temp=bins1[j]-bins2[j];
                  chsq += temp*temp/(bins1[j]+bins2[j]);
            }
      prob=gam.gammq(0.5*df,0.5*chsq);
}
```

3.3.4.2. La Prueba de Kolmogorov - Smirnov

Aquí esta la rutina para el caso de una distribución: ksone

```
void ksone(VecDoub_IO &data, Doub func(const Doub), Doub &d, Doub &prob)
{
    /** Dado un vector data[0 ... n-1], y dada una función suministrada por el
            usuario de una sola variable func que es una función de distribución
            acumulada que va de 0 (para los valores más pequeños de su
            argumento) a 1 (para valores mayores de su argumento), esta rutina
            devuelve la estadística K-S d y el p - valor prob . Los valores
            pequeños de prob muestran que la función de distribución acumulativa
            de los datos es significativamente diferente de func. Los datos de
```

```
            la matriz data se modifica al ser ordenados en orden ascendente.
      */
      Int j,n=data.size();
      Doub dt,en,ff,fn,fo=0.0;
      KSdist ks;
      sort(data);
      en=n;
      d=0.0;
      for (j=0;j<n;j++) {
            fn=(j+1)/en;
            ff=func(data[j]);
            dt=MAX(abs(fo-ff),abs(fn-ff));
            if (dt > d) d=dt;
            fo=fn;
      }
      en=sqrt(en);
      prob=ks.qks((en+0.12+0.11/en)*d);
}
```

Mientras que la estadística K-S esta destinada para su uso con una distribución continua, también puede ser utilizada para una distribución discreta. Este refinamiento se incluye en la rutina kstwo.

```
void kstwo(VecDoub_IO &data1, VecDoub_IO &data2, Doub &d, Doub &prob)
{
      /** Dado un vector data1[0 ... n1-1] y array data2[0 ... n2-1], esta
            Rutina devuelve la estadística K-S d y el valor de p prob para la
            Hipótesis nula de que los conjuntos de datos se han extraído de
            la misma distribución. Los valores pequeños de prob muestran que
            la función de distribución acumulativa de data1 es
            significativamente diferente de la de data2. El data1 y data2 se
            modifican al ser ordenados en orden ascendente.
      */
      Int j1=0,j2=0,n1=data1.size(),n2=data2.size();
      Doub d1,d2,dt,en1,en2,en,fn1=0.0,fn2=0.0;
      KSdist ks;
      sort(data1);
      sort(data2);
      en1=n1;
      en2=n2;
      d=0.0;
      while (j1 < n1 && j2 < n2) {
            if ((d1=data1[j1]) <= (d2=data2[j2]))
                  do
                        fn1=++j1/en1;
```

```
                while (j1 < n1 && d1 == data1[j1]);
            if (d2 <= d1)
                do
                    fn2=++j2/en2;
                while (j2 < n2 && d2 == data2[j2]);
            if ((dt=abs(fn2-fn1)) > d) d=dt;
        }
        en=sqrt(en1*en2/(en1+en2));
        prob=ks.qks((en+0.12+0.11/en)*d);
}
```

3.3.5. Comparación de dos velocidades medias. Uso de la prueba de Kolmogorov – Smirnov para probar la normalidad de datos de velocidad. Programa de Prueba.

A continuación se muestra un programa de prueba que resuelve 2 problemas tipos; la primera prueba si hay o no una diferencia significativa entre las medias de dos conjuntos de velocidades para un nivel de confianza dado, se usa la prueba de t Student. El segundo problema prueba si un conjunto de velocidades se adapta o no a una distribución normal, usando la prueba de normalidad de Kolmogorov – Smirnov.

```
#include "C:\nr_c304\code\nr3.h"
#include "C:\nr_c304\code\moment.h"
#include "C:\nr_c304\code\gamma.h"
#include "C:\nr_c304\code\incgammabeta.h"
#include "C:\nr_c304\code\sort.h"
#include "C:\nr_c304\code\erf.h"
#include "C:\nr_c304\code\stattests.h"
#include "C:\nr_c304\code\ksdist.h"
#include "C:\nr_c304\code\kstests.h"

#include <iostream>
#include <conio.h>

Doub func(const Doub);
double media;
double var;
double desvEst;             //Desviación estándar

int main()
{

        double veloc1[] = {
                40.,    35.,    38.,    37.,    33.,    30.,    28.,    35.,    35.,
                40.,    33.,    35.,    36.,    36.,    40.,    38.,    35.,            30.,
                30.,    38.,    39.,    35.,    36.,    34.,    33.,    31.,    36.,
                35.,    33.,    39.
        };
```

```
	double veloc2[] = {
		23.,	33.,	25.,	36.,	37.,	34.,	23.,	28.,	24.,
		31.,	24.,	20.,	21.,	28.,	35.,	25.,	21.,	35.,
		30.,	33.,	21.,	28.,	23.,	24.,	27.,	20.,	20.,
		30.,	32.,	33.
	};

	std::cout << "Para los dos conjuntos de datos de velocidad, probar si hay\n"
			<< " o no una diferencia significativa de las velocidades, con una\n"
			<< " probabilidad de error del 5%\n\n";

	const int numDatos = sizeof(veloc1)/ sizeof(veloc1[0]);

	VecDoub_I velocidad1(numDatos, veloc1);
	VecDoub_I velocidad2(numDatos, veloc2);
	double t;
	double prob;
	ttest(velocidad1, velocidad2, t, prob);

	std::cout << "El valor de la variable t Student's es = " << t << std::endl;
	std::cout << "La probabilidad es = " << prob << std::endl;

	if(prob < 0.05)
		std::cout << "Para un nivel de confianza del 95% hay una diferencia\n"
				<< "significativa de las velocidades medias\n";
	else
		std::cout << "Para un nivel de confianza del 95%  no hay una diferencia\n"
				<< "significativa de las velocidades medias\n";
	getche();




	std::cout << "\n\nEl gerente de transporte de una municipalidad realiza una\n"
			<< "prueba de hipótesis para probar que μ = 50 contra μ ≠ 50, donde\n"
			<< "μ representa la velocidad media en un tramo de una calle en esa\n"
			<< "ciudad. En los datos que se presentan en la tabla 11.2 se dan \n"
			<< "las velocidades en millas/hora de 28 vehículos. Para esta \n"
			<< "prueba de hipótesis, usando la distribución t de Student, se\n"
			<< "supone que los datos empleados para la prueba han sido tomados\n"
			<< "de una población distribuida normalmente. Esta suposición de \n"
			<< "normalidad se puede probar con una prueba de normalidad de \n"
			<<  "Kolmogorov - Smirnov, para un nivel de significancia
convencional\n"
			<< "α = 0.05. Si la suposición de normalidad no se rechaza, entonces\n"
			<< "se procede a realizar la prueba de hipótesis en que μ = 50 \n"
			<< "contra μ ≠ 50 empleando α = 0.05\n\n";

	double tabla11_2[] = {
		68.,	49.,	45.,	76.,	65.,	50.,
		54.,	92.,	24.,	36.,	60.,	66.,
		57.,	74.,	52.,	75.,	36.,	40.,
		62.,	56.,	94.,	57.,	64.,
		72.,	65.,	59.,	45.,	33.
	};


	const int numDat = sizeof(tabla11_2)/ sizeof(tabla11_2[0]);

	VecDoub_IO velocid(numDat, tabla11_2);
```

```
	/** La rutina avevar. Dado la serie de datos data [0 ... n-1], devuelve su
		media como media y su varianza como var
	*/
	avevar(velocid, media, var);

	desvEst = sqrt(var);
	std::cout << "La velocidad media es = " << media << " millas/hora\n"
		  << "La varianza es = " << var << " millas²/hora²\n"
		  << "La desviación estándar es = " << desvEst << " millas/hora\n";

	std::cout << "Con estos datos, media y desviación estándar, crearemos una\n"
		  << "distribución normal, que se acomode a los datos de velocidad\n";

	std::cout << "Ahora llamamos a la rutina ksone con la función de \n"
		  << "distribución acumulativa de la distribución normal \n"
		  << "velocNormal, double cdf(double); que se pasa como argumento\n"
		  << "a la rutina ksone, para la funcion func\n";

	Doub d;
	Doub probks;

	ksone(velocid, func, d, probks);

	std::cout << "El valor de la variable K-S d = " << d << std::endl;
	std::cout << "La probabilidad es = " << probks << std::endl;

	if(probks < 0.05) {
		std::cout << "Para un nivel de confianza del 95% hay una diferencia\n"
				<< "significativa de la función de distribución acumulativa
de\n"
				<< "los datos de velocidad con el de la función func\n";

		std::cout << "Por tanto ya no se realiza la prueba t de Student\n";
	}
	else {
		std::cout << "Para un nivel de confianza del 95%  no hay una diferencia\n"
				<< "significativa de la función de distribución acumulativa
de\n"
				<< "los datos de velocidad con el de la función func\n";

	//realizar student

	}



	getche();


	return 0;
}


Doub func(const Doub x) {
	Normaldist velocNormal(media, desvEst);
	return velocNormal.cdf(x);
}
```

Esta es la salida:

Para los dos conjuntos de datos de velocidad, probar si hay

```
 o no una diferencia significativa de las velocidades, con una
 probabilidad de error del 5%

El valor de la variable t Student's es = 6.61589
La probabilidad es = 1.29088e-08
Para un nivel de confianza del 95% hay una diferencia
significativa de las velocidades medias


El gerente de transporte de una municipalidad realiza una
prueba de hipótesis para probar que μ = 50 contra μ ≠ 50, donde
μ representa la velocidad media en un tramo de una calle en esa
ciudad. En los datos que se presentan en la tabla 11.2 se dan
las velocidades en millas/hora de 28 vehículos. Para esta
prueba de hipótesis, usando la distribución t de Student, se
supone que los datos empleados para la prueba han sido tomados
de una población distribuida normalmente. Esta suposición de
normalidad se puede probar con una prueba de normalidad de
Kolmogorov - Smirnov, para un nivel de significancia convencional
α = 0.05. Si la suposición de normalidad no se rechaza, entonces
se procede a realizar la prueba de hipótesis en que μ = 50
contra μ ≠ 50 empleando α = 0.05

La velocidad media es = 58.0714 millas/hora
La varianza es = 275.921 millas²/hora²
La desviación estándar es = 16.6109 millas/hora
Con estos datos, media y desviación estándar, crearemos una
distribución normal, que se acomode a los datos de velocidad
Ahora llamamos a la rutina ksone con la función de
distribución acumulativa de la distribución normal
velocNormal, double cdf(double); que se pasa como argumento
a la rutina ksone, para la funcion func
El valor de la variable K-S d = 0.0687921
La probabilidad es = 0.999023
Para un nivel de confianza del 95%  no hay una diferencia
significativa de la función de distribución acumulativa de
los datos de velocidad con el de la función func
```

3.3.6. Análisis por Tabla de Contingencia de Dos Distribuciones

Para variables nominales, se usa el análisis de contingencia o de análisis de tabla de contingencia.

3.3.6.1. Medidas de Asociación Basadas en Chi-Cuadrado

```
void cntab(MatInt_I &nn, Doub &chisq, Doub &df, Doub &prob, Doub &cramrv,
	Doub &ccc)
{
	/** Dado una tabla de contingencia bidimensional en forma de una matriz
		nn [0 ... ni-1] [0 ... nj-1] de enteros, esta rutina devuelve el
		chi-cuadrado chisq , el número de grados de libertad df, el valor - p,
		prob  (valores pequeños que indica una asociación significativa), y dos
		medidas de asociación, V de Cramer (cramrv) y el coeficiente de
		contingencia C (ccc)
	*/
	const Doub TINY=1.0e-30;
	Gamma gam;
	Int i,j,nnj,nni,minij,ni=nn.nrows(),nj=nn.ncols();
	Doub sum=0.0,expctd,temp;
	VecDoub sumi(ni),sumj(nj);
	nni=ni;
	nnj=nj;
	for (i=0;i<ni;i++) {
		sumi[i]=0.0;
		for (j=0;j<nj;j++) {
			sumi[i] += nn[i][j];
			sum += nn[i][j];
		}
		if (sumi[i] == 0.0) --nni;
	}
	for (j=0;j<nj;j++) {
		sumj[j]=0.0;
		for (i=0;i<ni;i++) sumj[j] += nn[i][j];
		if (sumj[j] == 0.0) --nnj;
	}
	df=nni*nnj-nni-nnj+1;
	chisq=0.0;
	for (i=0;i<ni;i++) {
		for (j=0;j<nj;j++) {
			expctd=sumj[j]*sumi[i]/sum;
			temp=nn[i][j]-expctd;
			chisq += temp*temp/(expctd+TINY);
		}
	}
	prob=gam.gammq(0.5*df,0.5*chisq);
	minij = nni < nnj ? nni-1 : nnj-1;
	cramrv=sqrt(chisq/(sum*minij));
	ccc=sqrt(chisq/(chisq+sum));
}
```

A continuación se muestra un programa de prueba y su salida para el análisis de una tabla de contingencia por chi-cuadrado. (del Libro *Estadística* de Murray R. S. y Larry J. S. 2009, McGraw - Hill)

Tabla 12.27

		Edad del conductor				
		21-30	31-40	41-50	51-60	61-70
Número de accidentes	0	748	821	786	720	672
	1	74	60	51	66	50
	2	31	25	22	16	15
	>2	9	10	6	5	7

```
#include <iostream>
#include <conio.h>

#include "C:\nr_c304\code\nr3.h"
#include "C:\nr_c304\code\sort.h"
#include "C:\nr_c304\code\moment.h"
#include "C:\nr_c304\code\gamma.h"
#include "C:\nr_c304\code\erf.h"
#include "C:\nr_c304\code\incgammabeta.h"
#include "C:\nr_c304\code\stattests.h"


int main()
{


	std::cout << "En la tabla 12.27 se muestran los resultados de una encuesta\n"
		  << "realizada con objeto de determinar si la edad de un
		  conductor\n"
		  << "de 21 años o más tiene alguna relación con la cantidad de\n"
		  << "accidentes automovilísticos en los que se ve implicado\n"
		  << "(incluyendo accidentes menores). Al nivel de significancia:\n"
		  << "a). 0.05 b).0.01, probar la hipótesis de que la cantidad de\n"
		  << "accidentes es independiente de la edad del conductor.
		  ¿Cuáles\n"
		  << "pueden ser las fuentes de dificultad en la técnica de
		  muestreo,\n"
		  << "así como otras consideraciones, que puedan afectar los\n"
		  << "resultados\n";


	int tabla12_27[] = {
		748,   821,   786,   720,   672,
		74 ,   60,    51,    66,    50,
		31,    25,    22,    16,    15,
		9,     10,    6,     5,     7
	};


	MatInt_I nn(4, 5, tabla12_27);
	double chisq;
	double df;
	double prob;
	double cramrv;
	double ccc;

	cntab(nn, chisq, df, prob, cramrv, ccc);
```

```
	std::cout << "La probabilidad es prob = " << prob << std::endl;
	std::cout << "La V de Cramer es = " << cramrv << std::endl;
    std::cout << "El coeficiente de contingencia C = " << ccc << std::endl;

	if (prob < 0.05) {
		std::cout << "Hay una asociación significativa entre la edad del \n"
			<< "conductor y los accidentes provocados, con un 95% de
				\n"
			<< "probabilidad\n";
	}
	else
		std::cout << "No hay una asociación significativa entre la edad
			del \n"
			<< "conductor y los accidentes provocados, con un 95% de
			\n"
			<< "probabilidad\n";

	getche();

	return 0;
}
```

La salida es:

```
En la tabla 12.27 se muestran los resultados de una encuesta
realizada con objeto de determinar si la edad de un conductor
de 21 años o más tiene alguna relación con la cantidad de
accidentes automovilísticos en los que se ve implicado
(incluyendo accidentes menores). Al nivel de significancia:
a). 0.05 b). 0.01, probar la hipótesis de que la cantidad de
accidentes es independiente de la edad del conductor. ¿Cuáles
pueden ser las fuentes de dificultad en la técnica de muestreo,
así como otras consideraciones, que puedan afectar los
resultados
La probabilidad es prob = 0.276181
La V de Cramer es = 0.0338248
El coeficiente de contingencia C = 0.058486
No hay una asociación significativa entre la edad del
conductor y los accidentes provocados, con un 95% de
probabilidad
```

3.3.6.2. Correlación Lineal

Pasamos ahora a las medidas de asociación entre las variables que son ordinales o continuas, en lugar de la variable nominal. Más ampliamente usado es el coeficiente de correlación lineal. La rutina que calcula el coeficiente de correlación lineal o de Pearson es:

```
void pearsn(VecDoub_I &x, VecDoub_I &y, Doub &r, Doub &prob, Doub &z)
{
	/** Dadas dos matrices x[0 ... n-1] e y[0 ... n-1], esta rutina calcula
		Su coeficiente de correlación r (devuelto como r), el p-valor en el
		cual la hipótesis nula de correlación cero es refutada (prob cuyo
		pequeño indica una correlación significativa), y z de Fisher
		(devuelto como z), cuyo valor puede ser utilizado en otras pruebas
		estadísticas como se describe anteriormente.
	*/
	const Doub TINY=1.0e-20;
```

```
	Beta beta;
	Int j,n=x.size();
	Doub yt,xt,t,df;
	Doub syy=0.0,sxy=0.0,sxx=0.0,ay=0.0,ax=0.0;
	for (j=0;j<n;j++) {
		ax += x[j];
		ay += y[j];
	}
	ax /= n;
	ay /= n;
	for (j=0;j<n;j++) {
		xt=x[j]-ax;
		yt=y[j]-ay;
		sxx += xt*xt;
		syy += yt*yt;
		sxy += xt*yt;
	}
	r=sxy/(sqrt(sxx*syy)+TINY);
	z=0.5*log((1.0+r+TINY)/(1.0-r+TINY));
	df=n-2;
	t=r*sqrt(df/((1.0-r+TINY)*(1.0+r+TINY)));
	prob=beta.betai(0.5*df,0.5,df/(df+t*t));
	// prob=erfcc(abs(z*sqrt(n-1.0))/1.4142136);
}
```

3.3.7. La correlación No Paramétrica o Rango

3.3.7.1. Coeficiente de Spearman de Correlación orden de rango

Esta es la rutina que calcula el Coeficiente de Correlación de Spearman.

```
void spear(VecDoub_I &data1, VecDoub_I &data2, Doub &d, Doub &zd, Doub &probd,
	Doub &rs, Doub &probrs)
{
    /** Dados dos matrices, data1 [0 ... n-1] y data2 [0 ... n-1], esta rutina
		vuelve la suma los cuadrados de la diferencia de rangos como D, el
		número de desviaciones estándar por el cual D se desvía de hipótesis
		nula valor esperado como zd, el valor de p bilateral de esta
		desviación como probd, rs correlación de Spearman como rs, y el
		valor de p bilateral de su desviación de cero como probrs. La rutina
		crank externa (abajo) y sort2 se utilizan. Un pequeño valor de
		cualquiera de probd o probrs indica una correlación significativa
		(rs positivo) o anticorrelación (rs negativo)
```

```
	*/
	Beta bet;
	Int j,n=data1.size();
	Doub vard,t,sg,sf,fac,en3n,en,df,aved;
	VecDoub wksp1(n),wksp2(n);
	for (j=0;j<n;j++) {
		wksp1[j]=data1[j];
		wksp2[j]=data2[j];
	}
	sort2(wksp1,wksp2);
	crank(wksp1,sf);
	sort2(wksp2,wksp1);
	crank(wksp2,sg);
	d=0.0;
	for (j=0;j<n;j++)
		d += SQR(wksp1[j]-wksp2[j]);
	en=n;
	en3n=en*en*en-en;
	aved=en3n/6.0-(sf+sg)/12.0;
	fac=(1.0-sf/en3n)*(1.0-sg/en3n);
	vard=((en-1.0)*en*en*SQR(en+1.0)/36.0)*fac;
	zd=(d-aved)/sqrt(vard);
	probd=erfcc(abs(zd)/1.4142136);
	rs=(1.0-(6.0/en3n)*(d+(sf+sg)/12.0))/sqrt(fac);
	fac=(rs+1.0)*(1.0-rs);
	if (fac > 0.0) {
		t=rs*sqrt((en-2.0)/fac);
		df=en-2.0;
		probrs=bet.betai(0.5*df,0.5,df/(df+t*t));
	} else
		probrs=0.0;
}

void crank(VecDoub_IO &w, Doub &s)
{
	/*	Dada una matriz ordenada w[0 ... n-1], sustituye a los elementos
		por su rango, incluyendo clasificación media de los vínculos,
		y regresa S como la suma de f * f * f - f, donde f es el número
		de elemento en cada vínculo.
	*/
	Int j=1,ji,jt,n=w.size();
	Doub t,rank;
	s=0.0;
	while (j < n) {
		if (w[j] != w[j-1]) {
			w[j-1]=j;
			++j;
		} else {
```

```
                for (jt=j+1;jt<=n && w[jt-1]==w[j-1];jt++);
                rank=0.5*(j+jt-1);
                for (ji=j;ji<=(jt-1);ji++)
                      w[ji-1]=rank;
                t=jt-j;
                s += (t*t*t-t);
                j=jt;
          }
      }
      if (j == n) w[n-1]=n;
}
```

3.3.7.2. La Correlación No Paramétrica Tau de Kendall

Tau de Kendall es aún más no paramétrico de r_s de Spearman o D. Sin embargo, si usted está dispuesto a compatimentar sus datos en un número moderado de contenedores, sigue leyendo.

```
void kendl1(VecDoub_I &data1, VecDoub_I &data2, Doub &tau, Doub &z, Doub &prob)
{
      /** kendal1 Teniendo en cuenta los datos de arrays data1 [0 ... n-1] y
            data2 [0 ... n-1], este programa vuelve tau de Kendall como tau,
            su de desviaciones estándar desde cero como z, p-valor bilateral
            como prob. Los valores pequeños de prob indican una correlación
            significativa (tau positivo) o anticorrelación (tau negativo)
      */
      Int is=0,j,k,n2=0,n1=0,n=data1.size();
      Doub svar,aa,a2,a1;
      for (j=0;j<n-1;j++) {
            for (k=j+1;k<n;k++) {
                  a1=data1[j]-data1[k];
                  a2=data2[j]-data2[k];
                  aa=a1*a2;
                  if (aa != 0.0) {
                        ++n1;
                        ++n2;
                        aa > 0.0 ? ++is : --is;
                  } else {
                        if (a1 != 0.0) ++n1;
                        if (a2 != 0.0) ++n2;
                  }
            }
      }
      tau=is/(sqrt(Doub(n1))*sqrt(Doub(n2)));
      svar=(4.0*n+10.0)/(9.0*n*(n-1.0));
```

```
	z=tau/sqrt(svar);
	prob=erfcc(abs(z)/1.4142136);
}
```

A veces sucede que hay sólo unos pocos valores posibles para cada *x* e *y*. En ese caso, los datos se pueden grabar como una tabla de contingencia que da el número de puntos de datos para cada contingencia de *x* e *y*.

```
void kendl2(MatDoub_I &tab, Doub &tau, Doub &z, Doub &prob)
{
	/** Dado un tabla de dos dimensiones tab[0 ... i-1] [0 ... j-1], de tal
	    manera que tab[k] [l] contiene el número de eventos que caen en el
	    compartimento k de una variable y compartimento l de otro, este
	    programa vuelve tau de Kendall como tau, su número de desviaciones
	    estándar a partir de cero como z, y su valor p de dos caras como
	    prob. Los valores pequeños de prob indican una correlación
	    significativa (tau positivo) o anticorrelación (tau negativa) entre
	    las dos variables. Aunque tab es una matriz de doubles , lo habitual
	    es que contiene valores enteros.
	*/
	Int k,l,nn,mm,m2,m1,lj,li,kj,ki,i=tab.nrows(),j=tab.ncols();
	Doub svar,s=0.0,points,pairs,en2=0.0,en1=0.0;
	nn=i*j;
	points=tab[i-1][j-1];
	for (k=0;k<=nn-2;k++) {
		ki=(k/j);
		kj=k-j*ki;
		points += tab[ki][kj];
		for (l=k+1;l<=nn-1;l++) {
			li=l/j;
			lj=l-j*li;
			mm=(m1=li-ki)*(m2=lj-kj);
			pairs=tab[ki][kj]*tab[li][lj];
			if (mm != 0) {
				en1 += pairs;
				en2 += pairs;
				s += (mm > 0 ? pairs : -pairs);
			} else {
				if (m1 != 0) en1 += pairs;
				if (m2 != 0) en2 += pairs;
			}
		}
	}
	tau=s/sqrt(en1*en2);
	svar=(4.0*points+10.0)/(9.0*points*(points-1.0));
```

```
        z=tau/sqrt(svar);
        prob=erfcc(abs(z)/1.4142136);
}
```

3.3.8. Programa de Prueba, Correlación Lineal, Correlación Orden de Rango No Paramétrica de Spearman y Tau de Kendall

Se muestra un programa de prueba de las Correlaciones Paramétricas y No Paramétricas, para probar si existe correlación o no. [4]

Tabla 6.1 Observaciones de velocidad y densidad en un camino rural

(a) Cálculos para el ejemplo 6.2

Velocidad, u, (mi/h) y_i	*Densidad, k (veh/mi) x_i*	$x_i y_i$	x_i^2
53.2	20	1 064.0	400
48.1	27	1 298.7	729
44.8	35	1 568.0	1 225
40.1	44	1 764.4	1 936
37.3	52	1 939.6	2 704
35.2	58	2 041.6	3 364
34.1	60	2 046.0	3 600
27.2	64	1 740.8	4 096
20.4	70	1 428.0	4 900
17.5	75	1 312.5	5 625
14.6	82	1 197.2	6 724
13.1	90	1 179.0	8 100
11.2	100	1 120.0	10 000
8.0	115	920.0	13 225
$\Sigma = 404.8$	$\Sigma = 892$	$\Sigma = 20\,619.8$	$\Sigma = 66\,628.0$
$\bar{y} = 28.91$	$\bar{x} = 63.71$		

```
#include <iostream>
#include <conio.h>

#include "C:\nr_c304\code\nr3.h"
#include "C:\nr_c304\code\sort.h"
#include "C:\nr_c304\code\moment.h"
#include "C:\nr_c304\code\gamma.h"
#include "C:\nr_c304\code\erf.h"
#include "C:\nr_c304\code\incgammabeta.h"
#include "C:\nr_c304\code\stattests.h"


int main()
{
```

```
	std::cout << "Utilicemos los datos mostrados en la tabla 6.1 para demostrar\n"
			<< "el uso del coeficiente de correlación lineal r o de Pearson\n"
			<< "y veamos si hay o no asociación significativa entre estas dos\n"
			<<  "variables  (Velocidad  en  millas/hora)  y  (Densidad  k
veh/milla)\n";

	double y[] = { //velocidades en millas por hora
		53.2,  48.1,  44.8,  40.1,  37.3,  35.2,  34.1,  27.2,  20.4,
		17.5,  14.6,  13.1,  11.2,  8.0
	};

	double x[] = {//densidad en vehículos por milla
		20.,   27.,   35.,   44.,   52.,   58.,   60.,   64.,   70.,
		75.,   82.,   90.,   100.,  115.
	};

	const int numDatos = sizeof(x) / sizeof(x[0]);

	VecDoub_I k(numDatos, x);
	VecDoub_I veloc(numDatos, y);
	double r;    //Coeficiente r de Pearson
	double prob;
	double z;

	pearsn(k, veloc, r, prob, z);
	std::cout <<"COEFICIENTE DE CORRELACION LINEAL O DE PEARSON: r\n\n";
	std::cout << "El coeficiente de correlación lineal o de Pearson \nr  es = "
			<< r << std::endl;

	std::cout << "La z de Fisher es z = " << z << std::endl;

	std::cout << "La probabilidad es = " << prob << std::endl;
	std::cout << "Un pequeño valor de prob indica una significativa correlación\n";

	if (prob < 0.01) {
		std::cout << "Hay una correlación significativa entre la velocidad \n"
				<< "y la densidad, con una probabilidad mayor a 99% \n";

	}
	else
			std::cout  <<  "No  hay  una  correlación  significativa  entre
velocidad\n"
				<< "y la densidad, con una probabilidad mayor a 99% de \n";

	getche();


	std::cout << "Hemos visto que el Coeficiente de Correlación Lineal o de \n"
			<< "Pearson, no nos dice sobre si la correlación es significativa\n"
			<< "Por tanto para los mismos datos usaremos el Coeficiente de \n"
			<< "Correlación Orden de Rango No Paramétrica de Spearman\n";

	double d;
```

```
	double zd;
	double probd;
	double rs;
	double probrs;


	spear(k, veloc, d, zd, probd, rs, probrs);

	std::cout <<"\nCOEFICIENTE DE CORRELACION NO PARAMETRICA DE SPEARMAN: rs\n\n";
	std::cout << "El coeficiente de correlación de Spearman \nrs  es = "
		<< rs << std::endl;

	std::cout << "La suma de los cuadrados de la diferencia de los rangos \nD = "
		 << d << std::endl;

	std::cout << "El número de desviaciones estándar por el que D se desvía de\n"
		<< " la hipótesis nula zd = " << zd << std::endl;

	std::cout << "La probabilidad bilateral de esta desviación como \n probd = "
		<< probd << std::endl;

	std::cout << "La probabilidad bilateral de la desviación de rs \n de cero "
		<< "como probrs = " << probrs << std::endl;

	std::cout << "Un pequeño valor de probd o probrs indica una significativa \n"
		<< "correlación\n";

	if (probd < 0.01) {
		std::cout << "Hay una correlación significativa entre la velocidad \n"
			<< "y la densidad, con una probabilidad mayor a 99% \n";

	}
	else
		std::cout << "No hay una correlación significativa entre
velocidad\n"
			<< "y la densidad, con una probabilidad mayor a 99% de \n";

	getche();


		/* Volvemos a asignar los datos originales, ya que la anterior rutina
			spear(k, veloc, d, zd, probd, rs, probrs), reordenó los datos
		*/
		VecDoub_I densidad(numDatos, x);
		VecDoub_I velocidad(numDatos, y);
		double tau;
		//double z;  ya se definió para calcular r
		//double prob;  ya se definió para calcular r


		 kendl1(densidad, velocidad, tau, z, prob);

		 std::cout <<"\nCORRELACION NO PARAMETRICA TAU DE KENDALL : TAU\n\n";
```

```
              std::cout << "El tau de Kendal es tau = " << tau << std::endl;

             std::cout << "El número de desviaciones estándar por el que tau se \n"
                    << " desvía de cero z  = " << z << std::endl;

      std::cout << "La probabilidad bilateral de esta desviación como \n prob = "
                    << prob << std::endl;

      std::cout << "Un pequeño valor de prob indica una significativa \n"
                    << "correlación\n";

      if (probd < 0.01) {
             std::cout << "Hay una correlación significativa entre la velocidad \n"
                           << "y la densidad, con una probabilidad mayor a 99% \n";

      }
      else
                    std::cout   <<   "No   hay   una   correlación   significativa   entre
velocidad\n"
                           << "y la densidad, con una probabilidad mayor a 99% de \n";

      getche();


      return 0;
}
```

Esta es la salida del programa anterior

```
Utilicemos los datos mostrados en la tabla 6.1 para demostrar
el uso del coeficiente de correlación lineal r o de Pearson
y veamos si hay o no asociación significativa entre estas dos
variables (Velocidad en millas/hora) y (Densidad k veh/milla)
COEFICIENTE DE CORRELACION LINEAL O DE PEARSON: r

El coeficiente de correlación lineal o de Pearson
r  es = -0.973062
La z de Fisher es z = -2.1469
La probabilidad es = 5.20587e-09
Un pequeño valor de prob indica una significativa correlación
Hay una correlación significativa entre la velocidad
y la densidad, con una probabilidad mayor a 99%
Hemos visto que el Coeficiente de Correlación Lineal o de
Pearson, no nos dice sobre si la correlación es significativa
Por tanto para los mismos datos usaremos el Coeficiente de
Correlación Orden de Rango No Paramétrica de Spearman

COEFICIENTE DE CORRELACION NO PARAMETRICA DE SPEARMAN: rs

El coeficiente de correlación de Spearman
rs  es = -1
```

```
La suma de los cuadrados de la diferencia de los rangos
D = 910
El número de desviaciones estándar por el que D se desvía de
 la hipótesis nula zd = 3.60555
La probabilidad bilateral de esta desviación como
 probd = 0.000311491
La probabilidad bilateral de la desviación de rs
 de cero como probrs = 0
Un pequeño valor de probd o probrs indica una significativa
correlación
Hay una correlación significativa entre la velocidad
y la densidad, con una probabilidad mayor a 99%

CORRELACION NO PARAMETRICA TAU DE KENDALL : TAU

El tau de Kendal es tau = -1
El número de desviaciones estándar por el que tau se
 desvía de cero z  = -4.98179
La probabilidad bilateral de esta desviación como
 prob = 6.30005e-07
Un pequeño valor de prob indica una significativa
correlación
Hay una correlación significativa entre la velocidad
y la densidad, con una probabilidad mayor a 99%
```

Capítulo IV. DESARROLLO DEL TRABAJO DE LA TESIS. MODELAMIENTO DE DATOS O MODELOS DE GENERACIÓN DE VIAJES

4.1. Ajustando Datos a una Línea Recta

Un ejemplo concreto hará que las consideraciones de la sección anterior más significativa. Consideramos que el problema de la instalación de un conjunto de N puntos de datos (x_i, y_i), i = 0,…, N - 1 a un modelo de línea recta.

$$y(x) = y(x|a,b) = a + b \cdot x \qquad (4.1.1)$$

Este problema a menudo se llama *regresión lineal*, una terminología que se originó, hace mucho tiempo, en las ciencias sociales. Suponemos que el σ_i es la incertidumbre asociada a cada yi medida se conoce, y que (valores de la variable dependiente) de la xi se conocen con exactitud.

Para medir qué tan bien el modelo de acuerdo con los datos, se utiliza la función de mérito chi-cuadrado (15.1.6), que en este caso es

$$\chi^2(a,b) = \sum_{i=0}^{N-1} \left(\frac{y_i - a - b \cdot x_i}{\sigma_i} \right)^2 \qquad (4.1.2)$$

Si los errores de medición se distribuyen normalmente, entonces esta función de mérito dará estimaciones de parámetros de máxima verosimilitud de a y b; si los errores no se distribuyen normalmente, entonces las estimaciones no son de máxima verosimilitud, pero todavía puede ser útil en un sentido práctico.

La ecuación (4.1.2) se minimiza para determinar a y b. En su mínimo, derivados de chi-cuadrado (a, b) con respecto a la a, b se hacen cero:

$$0 = \frac{\partial \chi^2}{\partial} = -2 \sum_{i=0}^{N-1} \frac{y_i - a - b \cdot x_i}{\sigma_i^2}$$

$$0 = \frac{\partial \chi^2}{\partial} = -2 \sum_{i=0}^{N-1} \frac{x_i \cdot (y_i - a - b \cdot x_i)}{\sigma_i^2} \qquad (4.1.3)$$

Estas condiciones se pueden reescribir en una forma conveniente si definimos las siguientes sumas:

$$S \equiv \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \qquad S_x \equiv \sum_{i=0}^{N-1} \frac{x_i}{\sigma_i^2} \qquad S_y \equiv \sum_{i=0}^{N-1} \frac{y_i}{\sigma_i^2} \qquad (4.1.4)$$

$$S_x \equiv \sum_{i=0}^{N-1} \frac{x_i^2}{\sigma_i^2} \qquad S_x = \sum_{i=0}^{N-1} \frac{x_i y_i}{\sigma_i^2}$$

Con estas definiciones (4.1.3) se convierte en

$$\begin{aligned} a \cdot S + b \cdot S_x &= S_y \\ a \cdot S_x + b \cdot S_x &= S_x \end{aligned} \qquad (4.1.5)$$

La solución de estas dos ecuaciones con dos incógnitas se calcula como

$$\Delta = S \cdot S_x - (S_x)^2$$

$$a = \frac{S_x \cdot S_y - S_x \cdot S_x}{\Delta} \qquad (4.1.6)$$

$$b = \frac{S \cdot S_x - S_x \cdot S_y}{\Delta}$$

La ecuación (4.1.6) da la solución para los parámetros del modelo que mejor se ajuste a y b.

No hemos terminado, sin embargo. Debemos estimar las incertidumbres probables en las estimaciones de a y b, ya que, obviamente, los errores de medición en los datos deben introducir cierta incertidumbre en la determinación de estos parámetros. Si los datos son independientes, entonces cada uno contribuye su propio bit de incertidumbre de los parámetros. Consideración de propagación de errores muestra que el σ_f^2 varianza en el valor de cualquier función será

$$\sigma_f^2 = \sum_{i=0}^{N-1} \sigma_i^2 \left(\frac{\partial}{\partial y_i}\right)^2 \qquad (4.1.7)$$

Para la línea recta, los derivados de a y b con respecto a yi pueden ser evaluados directamente de la solución:

$$\frac{\partial}{\partial y_i} = \frac{S_x - S_x \cdot x_i}{\sigma_i^2 \cdot \Delta} \qquad (4.1.8)$$

$$\frac{\partial}{\partial y_i} = \frac{S \cdot x_i - S_x}{\sigma_i^2 \cdot \Delta}$$

Sumando sobre los puntos que en (4.1.7), obtenemos

$$\sigma_a^2 = {}^{S_x}/_{\Delta}$$

(4.1.9)

$$\sigma_b^2 = {}^{S}/_{\Delta}$$

que son las varianzas en las estimaciones de a y b, respectivamente. Veremos que también se necesita un número adicional para caracterizar adecuadamente la incertidumbre probable de la estimación de parámetros. Ese número es la covarianza de a y b, y (como veremos más adelante) está dada por

$$C \quad (a, b) = \; - {}^{S_x}/_{\Delta} \qquad (4.1.10)$$

El coeficiente de correlación entre la incertidumbre en a y la incertidumbre en b, que es un número entre -1 y 1, se sigue de (4.1.10)

$$r_a \quad = \frac{-S_x}{\sqrt{S \square S_x}} \qquad (4.1.11)$$

Un valor positivo de r_a indica que los errores en a y b es probable que tengan el mismo signo, mientras que un valor negativo indica los errores son anticorrelacionados, propensos a tener signos opuestos.

Todavía no hemos terminado. Debemos estimar la bondad de ajuste de los datos al modelo. ¡En ausencia de esta estimación, no tenemos la más mínima indicación de que los parámetros a y b en el modelo tienen ningún sentido en absoluto! La probabilidad Q de que un valor de chi-cuadrado tan pobre como el valor (4.1.2) debe ocurrir por casualidad es

Q = 1- Chisqdist (N-2) .cdf (chi-cuadrado) (4.1.12)

Aquí Chisqdist es nuestro objeto que realiza la función de distribución de chi cuadrado y cdf es su función de distribución acumulativa. Si Q es mayor que, digamos 0,1, entonces la bondad del ajuste, es creíble. Si es mayor que, digamos, 0.001, a continuación, el ajuste puede ser aceptable si los errores son no normal o han sido subestimados moderadamente. Si Q es inferior a 0.001, entonces el modelo y / o procedimiento de estimación de razón puede ser puesto en duda. En este último caso, recurrir a las estimaciones robustas para seguir adelante.

Si usted no sabe los errores de medición de los puntos individuales, σ_i, y está procediendo (peligrosamente) para utilizar la ecuación del chi-cuadrado para la estimación de estos errores, entonces aquí está el procedimiento para la estimación de las incertidumbres probables de los parámetros a y b: conjunto σ_i = 1 en toda ecuación a (4.1.6), y multiplicar σ_a y σ_b, tal como se obtiene de la ecuación (4.1.9), por el factor adicional $\sqrt{\chi^2/(N-2)}$, donde χ^2 se calcula (4.1.2) usando los parámetros ajustados a y b. Como se mencionó anteriormente, este procedimiento es equivalente a suponer un

buen ajuste, por lo que no obtiene una probabilidad Q independiente de bondad de ajuste. En el § 2.3.7.1 prometimos una relación entre el coeficiente de correlación lineal r (ecuación (2.3.7.1.1)) y una medida de bondad de ajuste, chi-cuadrado (ecuación 4.1.2). Para los datos no ponderados (todo $\sigma_i = 1$), esa relación es:

$$\chi^2 = (1 - r^2) \sum_{i=0}^{N-1} (y_i - \bar{y})^2 \qquad (4.1.13)$$

Para los datos con diferentes errores σ_i, las ecuaciones anteriores siguen siendo válidas si las sumas en las ecuaciones (4.1.13) y (2.3.7.1.1) se ponderan por $1/\sigma_i^2$.

El siguiente objeto, Fitab, lleva a cabo exactamente las operaciones que hemos discutido. Usted llama a su constructor, ya sea con o sin errores σ_i. Si no se dispone de los σ_i de son conocidos, la rutina asume valores iguales de σ_i para cada punto y supone un buen ajuste, como se discute en los Estimadores de Máxima Verosimilitud.

Las fórmulas (4.1.6) son susceptibles de error de redondeo. De acuerdo con ello, volvemos a escribir la siguiente manera: Definir

$$t_i = \frac{1}{\sigma_i}\left(x_i - \frac{S_x}{S}\right), \qquad i = 0, 1, \ldots, N-1 \qquad (4.1.14) \text{ y}$$

$$S_t = \sum_{i=0}^{N-1} t_i^2 \qquad (4.1.15)$$

Entonces, como se puede verificar por sustitución directa,

$$b = \frac{1}{S_t} \sum_{i=0}^{N-1} \frac{t_i \cdot y_i}{\sigma_i} \qquad (4.1.16)$$

$$a = \frac{S_y - S_x \cdot b}{S} \qquad (4.1.17)$$

$$\sigma_a^2 = \frac{1}{S}\left(1 + \frac{S_x^2}{S \cdot S_t}\right) \qquad (4.1.18)$$

$$\sigma_b^2 = \frac{1}{S_t} \qquad (4.1.19)$$

$$Cov(a, b) = -\frac{S_x}{S \cdot S_t} \qquad (4.1.20)$$

$$r_{ab} = \frac{Cov(a,b)}{\sigma_a \cdot \sigma_b} \qquad (4.1.21)$$

```
struct Fitab {
      /** Objeto para el ajuste de una línea recta y = a + bx a un conjunto
            De puntos (xi, yi), con o sin errores sigma_i disponible. Llamar
            a uno de los dos constructores para calcular el ajuste. Las
            respuestas son entonces disponibles como las variables a, b,
            siga, sigb, chi2, y q o sigdat.
      */
      Int ndata;
```

```
      Doub a, b, siga, sigb, chi2, q, sigdat;
      VecDoub_I &x, &y, &sig;

      Fitab(VecDoub_I &xx, VecDoub_I &yy, VecDoub_I &ssig)
      : ndata(xx.size()), x(xx), y(yy), sig(ssig), chi2(0.), q(1.),
sigdat(0.)  {
            /*    Constructor. Dado un conjunto de puntos de datos x[0 ...
                  ndata-1], y[0 ... ndata-1] con desviaciones estándar
                  individuales sig [0 ... ndata-1], establece a, b y su
                  respectiva probable incertidumbres siga y sigb, el chi-
                  cuadrado, y la probabilidad q de bondad de ajuste (que el
                  ajuste tendría χ² este grande o mayor)
            */

            Gamma gam;
            Int i;
            Doub ss=0.,sx=0.,sy=0.,st2=0.,t,wt,sxoss;
            b=0.0;
            for (i=0;i<ndata;i++) {
                  wt=1.0/SQR(sig[i]);
                  ss += wt;
                  sx += x[i]*wt;
                  sy += y[i]*wt;
            }
            sxoss=sx/ss;
            for (i=0;i<ndata;i++) {
                  t=(x[i]-sxoss)/sig[i];
                  st2 += t*t;
                  b += t*y[i]/sig[i];
            }
            b /= st2;
            a=(sy-sx*b)/ss;
            siga=sqrt((1.0+sx*sx/(ss*st2))/ss);
            sigb=sqrt(1.0/st2);
            for (i=0;i<ndata;i++) chi2 += SQR((y[i]-a-b*x[i])/sig[i]);
            if (ndata>2) q=gam.gammq(0.5*(ndata-2),0.5*chi2);
      }

      Fitab(VecDoub_I &xx, VecDoub_I &yy)
      : ndata(xx.size()), x(xx), y(yy), sig(xx), chi2(0.), q(1.), sigdat(0.)
{
            /*    Constructor. Como el anterior, pero sin errores conocidos
                  sig no se utiliza). Las incertidumbres SIGSA y SIGB son
                  estimadas asumiendo errores iguales para todos los puntos,
                  y que una línea recta es una buena opción. q se devuelve
                  como 1,0, la normalización de chi2 es la desviación
                  estándar unidad en todos los puntos, y sigdat se establece
                  en el error estimado de cada punto.
            */
            Int i;
            Doub ss,sx=0.,sy=0.,st2=0.,t,sxoss;
            b=0.0;
            for (i=0;i<ndata;i++) {
                  sx += x[i];
                  sy += y[i];
            }
            ss=ndata;
            sxoss=sx/ss;
            for (i=0;i<ndata;i++) {
                  t=x[i]-sxoss;
                  st2 += t*t;
                  b += t*y[i];
```

```
            }
            b /= st2;
            a=(sy-sx*b)/ss;
            siga=sqrt((1.0+sx*sx/(ss*st2))/ss);
            sigb=sqrt(1.0/st2);
            for (i=0;i<ndata;i++) chi2 += SQR(y[i]-a-b*x[i]);
            if (ndata > 2) sigdat=sqrt(chi2/(ndata-2));
            siga *= sigdat;
            sigb *= sigdat;
      }
};
```

4.1.1. Programa de Prueba de Regresión Lineal

El siguiente programa de prueba se usa para realizar una regresión lineal o un ajuste de los pares de datos a una recta; no se conocen los errores de medición, por tanto se usa el segundo constructor de la estructura o clase Fitab. Este problema está propuesto en el libro de “Ingeniería de Tránsito” de Cal y Mayor.

```
#include <iostream>
#include <conio.h>

#include "C:\nr_c304\code\nr3.h"
#include "C:\nr_c304\code\gamma.h"
#include "C:\nr_c304\code\incgammabeta.h"
#include "C:\nr_c304\code\fitab.h"


int main()
{

      double anyo[4] = {
            2003.,      2004.,      2005.,      2006.
      };

      double volumen[4] = {
            4731.,      5207.,      5501.,      5605.
      };

      std::cout << "En la tabla 8.11, para una carretera de dos carriles, se
\n"
                  << "presentan para diferentes años los volúmenes de
tránsito \n"
                  << "promedio diario TPDS, adapte los datos a una línea
recta.\n"
                  << "Pronostique el TPDS para el año 2010\n\n";

      VecDoub_I xx(4, anyo);        //años
      VecDoub_I yy(4, volumen);     //volumen

      Fitab regreL(xx, yy);

      std::cout << " a = " << regreL.a << std::endl;
      std::cout << " b = " << regreL.b << std::endl;
      std::cout << " y  = " << regreL.a << " + " << regreL.b << " * x " <<
std::endl;
```

```
      std::cout << "Desviación estándar de a = " << regreL.siga << std::endl;
      std::cout << "Desviación estándar de b = " << regreL.sigb<< std::endl;
      std::cout << "Chi Cuadrado = " << regreL.chi2 << std::endl;
      std::cout << "Probabilidad de bondad de ajuste q = " << regreL.q <<
std::endl;
      std::cout << "Error estimado de cada punto = " << regreL.sigdat <<
std::endl;

      std::cout << "El TPDS para el año 2010 habría sido = " <<
                  regreL.a + regreL.b * 2010 << std::endl;

      getche();

      return 0;
}
```

Cuyos resultados son:

```
En la tabla 8.11, para una carretera de dos carriles, se
presentan para diferentes años los volúmenes de tránsito
promedio diario TPDS, adapte los datos a una línea recta.
Pronostique el TPDS para el año 2010

 a = -579251
 b = 291.6
 y  = -579251 + 291.6 * x

Desviación estándar de a = 117907
Desviación estándar de b = 58.8211
Chi Cuadrado = 34599.2
Probabilidad de bondad de ajuste q = 1
Error estimado de cada punto = 131.528
El TPDS para el año 2010 habría sido = 6864.8
```

4.1.2. Mínimos Cuadrados Lineales Generalizados

Una generalización inmediata de la regresión lineal es encajar un conjunto de puntos de datos (x_i, y_i) a un modelo que no es sólo una combinación lineal de 1 y *x* (es decir, *a* + *b***x*), sino más bien una combinación lineal de cualquiera M funciones especificados de *x*. Por ejemplo, las funciones podrían ser *1, x, x²,* x^{M-1}, en cuyo caso sus combinaciones lineales generales,

$$y(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_{M-1} \cdot x^{M-1} \qquad (4.1.2.1)$$

es un polinomio de grado M - 1. O, las funciones podrían ser senos y cosenos, en cuyo caso su combinación lineal general es una serie de Fourier. La forma general de este tipo de modelo es

$$y(x) = \sum_{k=0}^{M-1} a_k \cdot X_k(x) \qquad (4.1.2.2)$$

donde las cantidades $X_0(x), \ldots, X_{M-1}(x)$ son funciones fijas arbitrarias de *x*, llamadas las funciones de base.

Tenga en cuenta que las funciones $X_K(x)$ pueden ser tremendamente funciones no lineales de x. En esta discusión, "lineal" se refiere sólo a la dependencia del modelo en sus parámetros a_K

Para estos modelos lineales generalizamos la discusión de la sección anterior mediante la definición de una función de mérito

$$\chi^2 = \sum_{i=0}^{N-1} \left[\frac{y_i - \sum_{K=0}^{M-1} a_K X_K(x_i)}{\sigma_i} \right]^2 \qquad (4.1.2.3)$$

Como antes, σ_i es el error de medición (desviación estándar) del punto de datos i-ésimo, supuesto conocido. Si no se conocen los errores de medición, pueden todos (como se discute en el extremo de § 15.1) establecerse en el valor constante σ = 1.

Una vez más, vamos a elegir como mejores parámetros aquellos que minimizan χ^2. Hay varias técnicas diferentes disponibles para la búsqueda de este mínimo. Dos de ellos son particularmente útiles, y vamos a discutir tanto en esta sección. Para introducirlos y dilucidar su relación, necesitamos un poco de notación.
Sea **A** una matriz cuyos N * M componentes están construidos a partir M de funciones de base evaluados en las N abscisas x_i, y desde los N errores de medición σ_i, por la prescripción

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i} \qquad (4.1.2.4)$$

La matriz **A** se llama la matriz de diseño del problema de montaje. Nótese que **A** en general tiene más filas que columnas, N > = M, ya que debe haber más puntos de datos que los parámetros del modelo que hay que resolver. (¡Puede ajustar una línea recta a dos puntos, pero no una ecuación de quinto grado muy significativo!) La matriz de diseño se muestra esquemáticamente en la figura 15.4.1

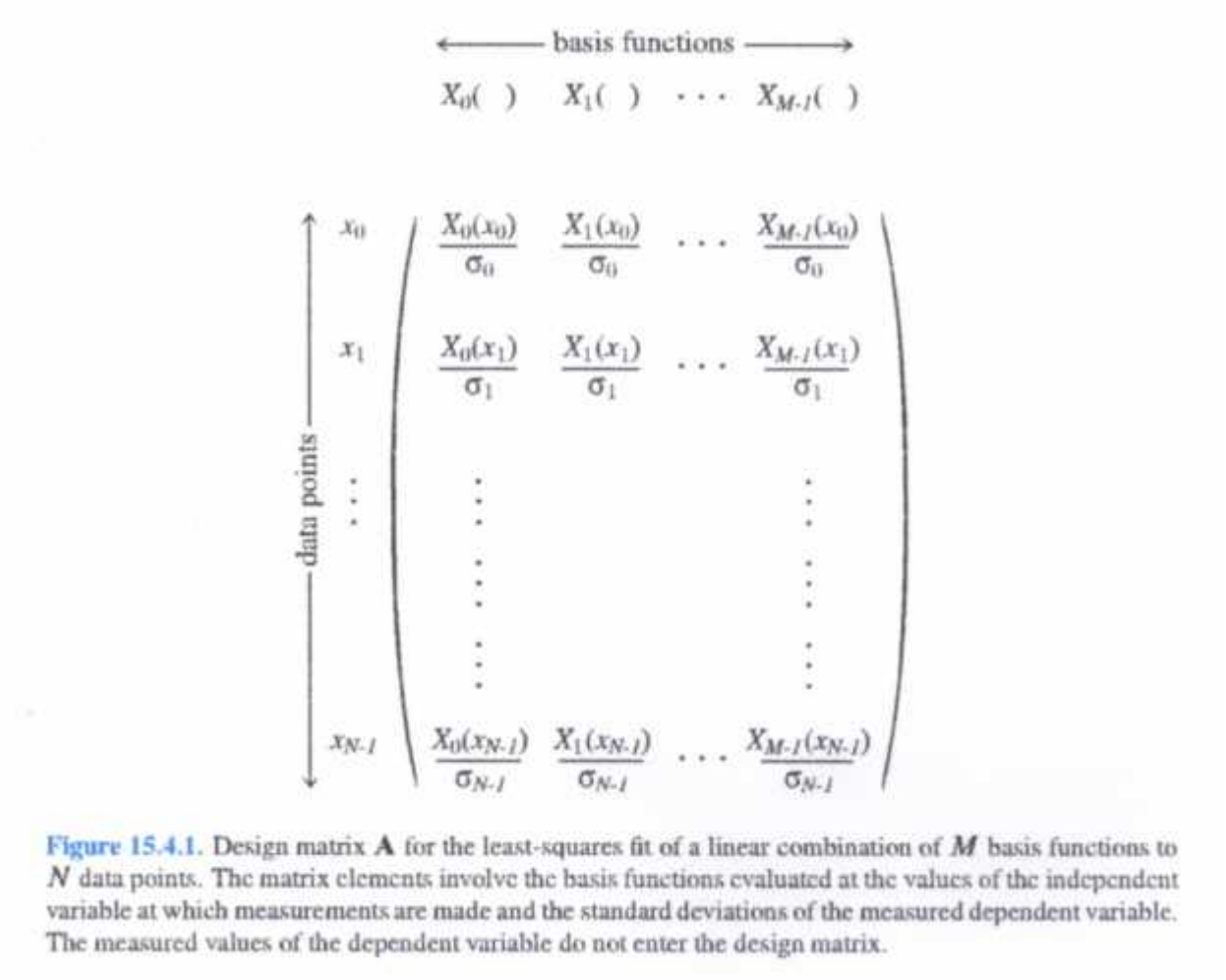


Figure 15.4.1. Design matrix **A** for the least-squares fit of a linear combination of M basis functions to N data points. The matrix elements involve the basis functions evaluated at the values of the independent variable at which measurements are made and the standard deviations of the measured dependent variable. The measured values of the dependent variable do not enter the design matrix.

Figura 15. Diseño de la matriz A. [14]

También se define un vector **b** de longitud N por

$$b_i = \frac{y_i}{\sigma_i} \qquad (4.1.2.5)$$

y denotar el vector cuyos *M* componentes son los parámetros que puedan montarse,$a_0, \ldots, a_{M-1}$, por **a.**

4.1.2.1. Solución Por el Uso de las Ecuaciones Normales

El mínimo de (4.1.2.3) se produce cuando la derivada de χ^2 con respecto a todos los *M* parámetros a_k se iguala a cero. Especializando la ecuación (15.1.8) para el caso del modelo (4.1.2.2), esta condición produce las *M* ecuaciones

$$0 = \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=0}^{M-1} a_j \, X_j(x_i) \right] X_k(x_i) \qquad k = 0, \ldots, M-1 \qquad (4.1.2.6)$$

Intercambiando el orden de las sumatorias, podemos escribir (4.1.2.6) como la ecuación matricial

$$\sum_{j=0}^{M-1} \alpha_k \; a_j = \beta_k \qquad (4.1.2.7)$$

Donde

$$\alpha_k = \sum_{i=0}^{N-1} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2} \quad o, s \quad e \qquad \boldsymbol{\alpha} = \mathbf{A}^{\mathbf{T}} \mathbf{A} \qquad (4.1.2.8)$$

Una matriz de *M* * *M,* y

$$\beta_k = \sum_{i=0}^{N-1} \frac{y_i X_k(x_i)}{\sigma_i^2} \quad 0, s \quad e \qquad \boldsymbol{\beta} = \mathbf{A}^{\mathbf{T}} \mathbf{b} \qquad (4.1.2.9)$$

Un vector de longitud *M*.

Las ecuaciones (4.1.2.6) o (4.1.2.7) son llamadas las *ecuaciones normales* del problema de los mínimos cuadrados.

Ellos pueden ser resueltos para el vector de parámetros **a** por el método estándar del álgebra lineal, especialmente la descomposición LU con sustitución hacia atrás, la descomposición Cholesky o la eliminación de Gauss-Jordan. En forma matricial, las ecuaciones normales se pueden escribir como cualquiera de éstos

$$\boldsymbol{\alpha} \mathbf{a} = \boldsymbol{\beta} \quad \text{o como} \quad (\mathbf{A}^{\mathbf{T}} \mathbf{A}) \mathbf{a} = \mathbf{A}^{\mathbf{T}} \mathbf{b} \qquad (4.1.2.10)$$

La matriz inversa $\mathbf{C} = \boldsymbol{\alpha}^{-1}$ llamada la matriz de covarianza, está estrechamente relacionada con las incertidumbres probables (o, más precisamente, estándar) de los parámetros **a** estimados. Para estimar estas incertidumbres, considere esto

$$a_j = \sum_{k=0}^{M-1} \boldsymbol{\alpha}_j^{-1} \beta_k = \sum_{k=0}^{M-1} C_j \left[\sum_{i=0}^{N-1} \frac{y_i X_k(x_i)}{\sigma_i^2} \right] \qquad (4.1.2.11)$$

y que la varianza asociada con la estimación de a_j se puede encontrar en (15.2.7) a partir de

$$\sigma^2(a_j) = \sum_{i=0}^{N-1} \sigma_i^2 \left(\frac{\partial a_j}{\partial y_i} \right)^2 \qquad (4.1.2.12)$$

Tenga en cuenta que a_k es independiente de y_i, así

$$\frac{\partial a_j}{\partial y_i} = \sum_{k=0}^{M-1} \frac{C_j X_k(x_i)}{\sigma_i^2} \qquad (4.1.2.13)$$

En consecuencia, nos encontramos con que

$$\sigma^2(a_i) = \sum_{k=0}^{M-1} \sum_{l=0}^{M-1} C_j C_j \left[\sum_{i=0}^{N-1} \frac{X_k(x_i) X_l(x_i)}{\sigma_i^2} \right] \qquad (4.1.2.14)$$

El término final entre paréntesis es la matriz **α**. Dado que esta es la matriz inversa de **C**, (4.1.2.14) reduce inmediatamente a

$$\sigma^2(a_i) = C_{jj} \qquad (4.1.2.15)$$

En otras palabras, los elementos diagonales de **C** son las varianzas (incertidumbres al cuadrado) de los parámetros buscado **a**. No debería sorprender al enterarse de que los elementos fuera de la diagonal C_{jk} son las covarianzas entre a_j y a_k; pero vamos a aplazar el debate de estos para § 15.6

Ahora vamos a dar una rutina que implementa las fórmulas anteriores para el problema general lineal de mínimos cuadrados, por el método de las ecuaciones normales. Puesto que deseamos calcular no sólo la solución del vector **a**, sino también la matriz de covarianza **C**, es más conveniente utilizar la eliminación de Gauss-Jordan (rutina gaussj) para realizar el álgebra lineal. El recuento de operación en esta solicitud no es más grande que la de la descomposición *LU*. Si usted no tiene ninguna necesidad de la matriz de covarianza, sin embargo, puede ahorrarse un factor de 3 en el álgebra lineal por el cambio a la descomposición *LU*, sin el cálculo de la matriz inversa. En teoría, ya que $\mathbf{A}^T \cdot \mathbf{A}$ es definida positiva, la descomposición de Cholesky es la forma más eficiente para resolver las ecuaciones normales. Sin embargo, en la práctica, la mayor parte del tiempo de cálculo se gastan en bucles a través de los datos para formar las ecuaciones, y el método de la eliminación de Gauss-Jordan es bastante adecuado.

Debemos advertir que la solución de un problema de mínimos cuadrados directamente de las ecuaciones normales es bastante susceptible a errores de redondeo, ya que el número de condición de la matriz de **α** es el cuadrado del número de condición de **A**. Una alternativa, y que se prefiere, es la técnica que consiste en la descomposición *QR* de la matriz de diseño **A**. Esto es esencialmente lo que hicimos al final de §15.2 para los datos de ajuste a una línea recta, pero sin llamar a todo la maquinaria de *QR* para derivar las fórmulas necesarias. Más adelante en esta sección, discutiremos otras dificultades en el problema de mínimos cuadrados, para el que la cura es la descomposición de valor singular (SVD), de la que se da una aplicación.

```
struct Fitlin {
      /*    Objeto para el ajuste de los mínimos cuadrados lineales
            generalizados, resolviendo las ecuaciones normales, incluyendo
            también la capacidad de mantener los parámetros especificados en,
            valores especificados fijos. Llamar al Constructor para enlazar
            vectores de datos y funciones de ajuste. Luego llame a cualquier
            combinación de funciones, gratuitamente y espera y en forma tan a
            menudo como se desee, los conjuntos de ajuste dan las cantidades de
            salida en a, covar y chsq.
      */
      Int ndat, ma;
      VecDoub_I &x,&y,&sig;
      VecDoub (*funcs)(const Doub);
      VecBool ia;

      VecDoub a;
      MatDoub covar;
      Doub chisq;

      Fitlin(VecDoub_I &xx, VecDoub_I &yy, VecDoub_I &ssig, VecDoub funks(const
Doub))
      : ndat(xx.size()), x(xx), y(yy), sig(ssig), funcs(funks) {
            /* Constructor. Vincula referencias a las matrices de datos xx,
```

```
            yy, ssig, y las funciones suministradas por el usuario
            funks(x) que devuelve un VecDoub que contiene ma funciones de
            base evaluados en x = x.
            Inicializa todos los parámetros como libres (no retenidos)
      */
      ma = funcs(x[0]).size();
      a.resize(ma);
      covar.resize(ma,ma);
      ia.resize(ma);
      for (Int i=0;i<ma;i++) ia[i] = true;
}

void hold(const Int i, const Doub val) {ia[i]=false; a[i]=val;}
void free(const Int i) {ia[i]=true;}
/*    Funciones opcionales para la consagración de un parámetro,
      identificados por un valor en el rango de 0, ..., ma-1,
      fija en el valor val, o para liberar un parámetro que se consagró
      previamente fijada, las funciones hold y free pueden ser llamados
      por cualquier número de parámetros antes de llamar a la función
      fit para calcular los valores de mejor ajuste para los restantes
      (no atados) parámetros, y el proceso puede repetirse varias veces.
       Alternativamente, se puede establecer el vector booleano ia
       directamente , antes de llamar a la función fit.
*/
void fit() {
      /*    Resolver las ecuaciones normales para la minimización de χ²
            para
            encajar algunos o todos los coeficientes a[0, ..., ma-1]
            de una función que depende linealmente de a,
            y = Σ_j a_j * funks_j (x). Los valores de respuesta son
            establecidos para a[0, ..., ma-1], χ² = chisq, y la matriz
            de covarianza covar[0, ..., ma-1 [0, ..., ma-1]

      */
      Int i,j,k,l,m,mfit=0;
      Doub ym,wt,sum,sig2i;
      VecDoub afunc(ma);
      for (j=0;j<ma;j++) if (ia[j]) mfit++;
      if (mfit == 0) throw("lfit: no parameters to be fitted");
      MatDoub temp(mfit,mfit,0.),beta(mfit,1,0.);
      for (i=0;i<ndat;i++) {
            afunc = funcs(x[i]);
            ym=y[i];
            if (mfit < ma) {
                  for (j=0;j<ma;j++)
                        if (!ia[j]) ym -= a[j]*afunc[j];
            }
            sig2i=1.0/SQR(sig[i]);
            for (j=0,l=0;l<ma;l++) {
                  if (ia[l]) {
                        wt=afunc[l]*sig2i;
                        for (k=0,m=0;m<=l;m++)
                              if (ia[m]) temp[j][k++] += wt*afunc[m];
                        beta[j++][0] += ym*wt;
                  }
            }
      }
      for (j=1;j<mfit;j++) for (k=0;k<j;k++) temp[k][j]=temp[j][k];
      gaussj(temp,beta);
      for (j=0,l=0;l<ma;l++) if (ia[l]) a[l]=beta[j++][0];
      chisq=0.0;
      for (i=0;i<ndat;i++) {
```

```
                  afunc = funcs(x[i]);
                  sum=0.0;
                  for (j=0;j<ma;j++) sum += a[j]*afunc[j];
                  chisq += SQR((y[i]-sum)/sig[i]);
            }
            for (j=0;j<mfit;j++) for (k=0;k<mfit;k++) covar[j][k]=temp[j][k];
            for (i=mfit;i<ma;i++)
                  for (j=0;j<i+1;j++) covar[i][j]=covar[j][i]=0.0;
            k=mfit-1;
            for (j=ma-1;j>=0;j--) {
                  if (ia[j]) {
                        for (i=0;i<ma;i++) SWAP(covar[i][k],covar[i][j]);
                        for (i=0;i<ma;i++) SWAP(covar[k][i],covar[j][i]);
                        k--;
                  }
            }
      }
};
```

De inmediato probamos el algoritmo anterior para unos conjuntos de datos muy simple, un ejemplo de juguete, una variable es el cubo de la otra. Veremos los resultados y la función modelada, note que las funciones básicas pueden ser polinomios u otras funciones trascendentales que el usuario puede dar.

```
#include <iostream>
#include <conio.h>

#include "C:\nr_c304\code\nr3.h"
#include "C:\nr_c304\code\gaussj.h"
#include "C:\nr_c304\code\fitlin.h"

VecDoub cubicfit(const double x);

int main()
{
      double x[4] = { 2, 3, 1, 4};
      double y[4] = { 8, 26, 1, 68};
      double sigm[4] = { 1, 1, 1, 1};

      VecDoub_I xx(4, x), yy(4, y), ssig(4, sigm);
      Fitlin myfit(xx, yy, ssig, cubicfit);
      myfit.fit();
   /**

   */

   for(int i = 0; i < 4; i++)
      std::cout << myfit.a[i] <<std::endl;

   std::cout << "y = "
            << myfit.a[0] << " + " << myfit.a[1] << " * x  + "
            << myfit.a[2] << " * x * x + " << myfit.a[3] << " * x * x * x \n";

   std::cout << "El valor, para x = 5,  de y = "
            << myfit.a[0]+ myfit.a[1]*5.0 + myfit.a[2]*25.0 + myfit.a[3]*125.0;

   std::cout << "\nLa matriz de covarianza es = \n"
```

```
            << myfit.covar ;

   std::cout << "\nEl chi-cuadrado es = " << myfit.chisq;

   getche();
      return 0;
}


VecDoub cubicfit(const double x) {
      VecDoub ans(4);
      ans[0] = 1.;
      for(int i = 1; i < 4; i++) ans[i] = x*ans[i-1];
      return ans;
}
```

La salida es:

```
-8
14.3333
-7.5
2.16667
y = -8 + 14.3333 * x  + -7.5 * x * x + 2.16667 * x * x * x
El valor, para x = 5,  de y = 147
La matriz de covarianza es =
69      -104.167        45      -5.83333
-104.167        161.389 -70.8333        9.27778
45      -70.8333        31.5    -4.16667
-5.83333        9.27778 -4.16667        0.555556

El chi-cuadrado es = 5.76058e-26
```

4.1.2.2. Solución Por el Uso de la Descomposición del Valor Singular

En algunas aplicaciones, las ecuaciones normales son perfectamente adecuadas para los problemas lineales de mínimos cuadrados. Sin embargo, en muchos otros casos, las ecuaciones normales están muy cerca de una matriz singular. Un elemento pivote cero se pueden encontrar durante la solución de las ecuaciones lineales (por ejemplo, en la rutina gaussj), en el que se no se obtiene alguna solución. O se puede producir un pequeño pivote, en cuyo caso se suele obtener el parámetro a_k equipada con grandes magnitudes que son delicadamente (y de forma inestable) equilibrados para cancelar casi precisamente cuando se evalúa la función ajustada.
¿Por qué ocurre esto comúnmente? Una razón matemática es que el número de condición de la matriz de **α** es el cuadrado del número de condición de **A**. Pero y una razón adicional es que, más a menudo que a los experimentadores les gustaría admitir, los datos no distinguen claramente entre dos o más de las funciones bases que proporcionan. Si dos de estas funciones, o dos combinaciones diferentes de funciones, pasan a ajustar los datos igual de bien - o igual de mal - entonces la matriz **α**, es incapaz de distinguir entre ellos, prolijamente pliega su tienda de campaña y se convierte en una matriz singular. Hay una cierta ironía matemática en el hecho de que los problemas de mínimos cuadrados son tanto sobredeterminados (el número de puntos de datos es más grande que el número de parámetros) y subdeterminado (existen combinaciones ambiguas de parámetros); pero así es como a menudo se da. Las ambigüedades pueden

ser extremadamente difíciles de detectar a priori en problemas complicados.

Introduzca la descomposición de valores singulares (SVD). Este sería un buen momento para que revisar las técnicas poderosas para tratar con conjuntos de ecuaciones o matrices que son o bien singulares o bien numéricamente muy cerca del singular, esta técnica es conocida como *descomposición de valores singulares*. En el caso de un sistema sobredeterminado, SVD produce una solución que es la mejor aproximación en el sentido de mínimos cuadrados. Eso es exactamente lo que queremos. En el caso de un sistema subdeterminado, SVD produce una solución cuyos valores (para nosotros, de la a_k) son más pequeños en el sentido de mínimos cuadrados. Eso también es lo que queremos: Cuando alguna combinación de función de base es irrelevante para el ajuste, esa combinación será conducido a un valor más pequeño, inofensivo, que empujó hasta la cancelación delicadamente de infinitos.

En términos de la matriz de diseño **A** (ecuación 4.1.2.4) y el vector **b** (ecuación 4.1.2.5), la minimización de χ^2 en (4.1.2.3) se puede escribir como

$$\chi^2 = |\mathbf{A} \cdot \mathbf{a} - \mathbf{b}|^2 \qquad (4.1.2.16)$$

Comparando a la ecuación (2.6.9), vemos que este es precisamente el problema que las rutinas en el objeto SVD están diseñados para resolver. La solución, que se da por la ecuación (2.6.12), se puede reescribir de la siguiente manera: Si **U** y **V** entran en la descomposición SVD de **A** según la ecuación (2.6.1), calculado por SVD, a continuación, dejar que los vectores $\mathbf{U}_{(i)}$ i = 0, ..., M-1 denotan las columnas de **U** (cada uno un vector de longitud N), y deja que los vectores $\mathbf{V}_{(i)}$ i= 0, ..., M-1 denotan las columnas de **V** (cada uno un vector de longitud M). A continuación, la solución (2.6.12) del problema de mínimos cuadrados (4.1.2.16) se puede escribir como

$$\mathbf{a} = \sum_{i=0}^{M-1} \left(\frac{\mathbf{U}_{(i)} \cdot \mathbf{b}}{w_i} \right) \mathbf{V}_{(i)} \qquad (4.1.2.17)$$

donde los w_i son, como §2.6, los valores singulares calculados por SVD.

La ecuación (4.1.2.17) dice que los parámetros equipada **a** son combinaciones lineales de las columnas de **V**, con coeficientes obtenidos mediante la formación de productos puntos de las columnas de **U** con el vector de datos ponderada (4.1.2.5). A pesar de que está fuera de nuestro alcance de probar aquí, resulta que los errores estándar (vagamente, "probables") en los parámetros de ajuste también son combinaciones lineales de las columnas de **V**. De hecho, la ecuación (4.1.2.17) se puede escribir en una forma de mostrar estos errores como

$$\mathbf{a} = \left[\sum_{i=0}^{M-1} \left(\frac{\mathbf{U}_{(i)} \cdot \mathbf{b}}{w_i} \right) \cdot \mathbf{V}_{(i)} \right] \pm \frac{1}{w_0} \mathbf{V}_{(0)} \pm \cdots \pm \frac{1}{w_{M-1}} \mathbf{V}_{(M-1)} \qquad (4.1.2.18)$$

Aquí cada ± seguido de una desviación estándar. El hecho sorprendente es que, descompuesto de esta manera, las desviaciones estándar son todos independientes entre sí (no correlacionada). Por lo tanto, se pueden sumar a la moda de la raíz cuadrada de la media. Lo que está pasando es que el vector $\mathbf{V}_{(i)}$ son los principales ejes del elipsoide de error de los parámetros ajustados **a** (ver §15.6).

De ello se desprende que la varianza en la estimación de un parámetro a_j está dada por

$$\sigma^2(a_j) = \sum_{i=0}^{M-1} \frac{1}{w_i^2} \left[V_{(j)}\right]_j^2 = \sum_{i=0}^{M-1} \left(\frac{V_j}{w_i}\right)^2 \qquad (4.1.2.19)$$

cuyo resultado debe ser idéntica a (4.1.2.14). Al igual que antes, no debe ser sorprendido en la fórmula para las covarianzas, aquí se dan sin pruebas,

$$C \quad (a_j, a_k) = \sum_{i-0}^{M-1} \left(\frac{V_j \, V_k}{w_i^2}\right) \qquad (4.1.2.20)$$

Hemos introducido este apartado señalando que las ecuaciones normales pueden fallar por encontrarse con un pivote cero. Aún sin embargo, no hemos mencionado cómo la SVD supera este problema. La respuesta es: Si algún valor singular w_i es cero, su recíproco en la ecuación (4.1.2.18) se debe establecer en cero, no en infinito. (Compárese la discusión anterior ecuación 2.6.7.) Esto corresponde a la adición de los parámetros equipada **a** a un múltiplo de cero, en lugar de algún múltiplo de un gran número aleatorio, de cualquier combinación lineal de funciones de base que están degeneradas en el ajuste.
Por otra parte, si un valor singular w_i es distinto de cero, pero muy pequeño, también debe definir su recíproco a cero, ya que su valor aparente es probablemente un artefacto de error de redondeo, no es un número significativo. Una respuesta plausible a la pregunta "¿cómo pequeño es pequeño?" es editar de esta forma todos los valores singulares cuya relación con el mayor valor singular es inferior a N veces la precisión ε de la máquina. (Esta es una recomendación más conservadora que el por defecto, en el que se escala como $N^{1/2}$).
Hay otra razón para la edición incluso de valores singulares adicionales, los suficientemente grandes tal que error de redondeo no es una pregunta. La descomposición de valor singular le permite identificar las combinaciones lineales de variables que acaba de pasar y no contribuir mucho a reducir el χ^2 de su conjunto de datos. La edición de estos a veces puede reducir los errores de error probables en sus coeficientes de forma significativa, al tiempo que aumenta el χ^2 mínimo sólo insignificantemente. Vamos a aprender más acerca de la identificación y el tratamiento de estos casos en el § 15.6.
En general, se recomienda que siempre utilice técnicas SVD en lugar de utilizar las ecuaciones normales. La única desventaja significativa de SVD es que requiere más espacio de almacenamiento de orden N * M de la matriz de diseño y su descomposición. El almacenamiento también se requiere para la matriz **V** de M * M, pero esto es en lugar de la matriz del mismo tamaño para los coeficientes de las ecuaciones normales; Sin embargo, su gran ventaja, que (en teoría) no puede fallar, más que compensa la desventaja de velocidad.

El siguiente objeto, Fitsvd, tiene una interfaz casi idéntica a Fitlin, de arriba. Un parámetro opcional adicional en el constructor establece el umbral para la edición de valores singulares.

```
struct Fitsvd {
      /*    Clase para los mínimos cuadrados lineales generalizados, ajustados
            mediante la descomposición de valor singular. Llame al constructor
            con los vectores de datos de vinculación y las funciones de ajuste.
            A continuación, llame en forma que fijó la cantidad de salida
            a, covar y chisq
      */
```

```
	Int ndat, ma;
	Doub tol;
	VecDoub_I *x,&y,&sig;   //(¿Por qué es x un puntero? Está explicado
abajo)
	VecDoub (*funcs)(const Doub); //Los valores de salida.
	VecDoub a;              // a es el vector de coeficientes encajados,
	MatDoub covar;          // covar es su matriz de covarianza,
	Doub chisq;              //y chisq es el valor de chi-cuadrado para el
ajuste.

	Fitsvd(VecDoub_I &xx, VecDoub_I &yy, VecDoub_I &ssig,
	VecDoub funks(const Doub), const Doub TOL=1.e-12)
	: ndat(yy.size()), x(&xx), xmd(NULL), y(yy), sig(ssig),
	funcs(funks), tol(TOL) {}
	/*    Constructor. Vincula referencias a las matrices de datos xx , yy,
	      y ssig, y las funciones suministradas por el usuario funks(x) que
	      devuelve un VecDoub de ma funciones de base que contiene, evaluados
	      en x = x. Si TOL es positivo, es el umbral (en relación con el mayor
	      valor singular) para descartar pequeños valores singulares. Si es
	      <= 0, se utiliza el valor por defecto en SVD.
	*/

	void fit() {
		/* Resuelve por la descomposición de valor singular la minimización
		      De χ² que se ajusta para los coeficientes de a[0 ... ma-1]
		      de una función que depende linealmente de a.   y=∑_i▒a_i * ⟦
		      funks⟧ _i (x) .  El conjunto de respuesta establecidos para
		      a[0 ... ma-1],  chisq = χ², y la matriz de covarianza covar[0
		      ... ma-1][0 ... ma-1].
		*/
		Int i,j,k;
		Doub tmp,thresh,sum;
		if (x) ma = funcs((*x)[0]).size();
		else ma = funcsmd(row(*xmd,0)).size();
		a.resize(ma);
		covar.resize(ma,ma);
		MatDoub aa(ndat,ma);
		VecDoub b(ndat),afunc(ma);
		for (i=0;i<ndat;i++) {
			if (x) afunc=funcs((*x)[i]);
			else afunc=funcsmd(row(*xmd,i));
			tmp=1.0/sig[i];
			for (j=0;j<ma;j++) aa[i][j]=afunc[j]*tmp;
			b[i]=y[i]*tmp;
		}
		SVD svd(aa);
		thresh = (tol > 0. ? tol*svd.w[0] : -1.);
		svd.solve(b,a,thresh);
		chisq=0.0;
		for (i=0;i<ndat;i++) {
			sum=0.;
			for (j=0;j<ma;j++) sum += aa[i][j]*a[j];
			chisq += SQR(sum-b[i]);
		}
		for (i=0;i<ma;i++) {
			for (j=0;j<i+1;j++) {
				sum=0.0;
				for (k=0;k<ma;k++) if (svd.w[k] > svd.tsh)
					sum += svd.v[i][k]*svd.v[j][k]/SQR(svd.w[k]);
				covar[j][i]=covar[i][j]=sum;
			}
		}
```

```
        }
        /*    A partir de aquí, el código de los encajes multidimensionales,
              que se discutirá más tarde
        */

        MatDoub_I *xmd;
        VecDoub (*funcsmd)(VecDoub_I &);

        Fitsvd(MatDoub_I &xx, VecDoub_I &yy, VecDoub_I &ssig,
        VecDoub funks(VecDoub_I &), const Doub TOL=1.e-12)
        : ndat(yy.size()), x(NULL), xmd(&xx), y(yy), sig(ssig),
        funcsmd(funks), tol(TOL) {}
        /*    Constructor para encajes multidimensionales. Exactamente el mismo
              que el constructor anterior, excepto que ahora xx es una matriz cuyas filas son
              los puntos de datos multidimensionales y funks ahora es una función
              de un punto de datos multidimensional (como un VecDoub).
        */

        VecDoub row(MatDoub_I &a, const Int i) {
              Int j,n=a.ncols();
              VecDoub ans(n);
              for (j=0;j<n;j++) ans[j] = a[i][j];
              return ans;
        }
};
```

Para problemas casi o degenerados, si quieres probar diferentes umbrales de valor singular, se llama al constructor Fitsvd una vez. A continuación, tantas veces como quieras, "llegar a" y aumentar tol, a continuación, llamar encaja de nuevo y examinar el valor resultante de chisq (y opcionalmente también la matriz de covarianza). Sigue adelante, siempre y cuando chisq no aumenta demasiado.

EJEMPLOS

Tenga en cuenta que algunos problemas aparentemente no lineales se pueden expresar de manera que son lineales. Por ejemplo, un modelo exponencial con dos parámetros a y b,

$$y(x) = a \cdot e^{-b \cdot x} \qquad (4.1.2.21)$$

puede ser reescrita como

$$\log[y(x)] = c - b \cdot x \qquad (4.1.2.22)$$

que es lineal en los parámetros c y b. (Por supuesto, usted debe ser consciente de que estas transformaciones no tienen exactamente los errores de Gauss en errores de Gauss.)
También esté atento a funciones "sin parámetros", como en

$$y(x) = a \cdot e^{(-b \cdot x + d)} \qquad (4.1.2.23)$$

Aquí los parámetros a y d son, de hecho son, indistinguibles. Este es un buen ejemplo de que las ecuaciones normales serán exactamente singulares, y donde SVD se encuentra un valor singular cero. SVD entonces hace una elección de mínimos

cuadrados para el establecimiento de un equilibrio entre la a y d (o, más bien, sus equivalentes en el modelo lineal calculado tomando los logaritmos). Sin embargo - y esto es cierto siempre SVD devuelve un valor singular cero - que es mejor aconseja averiguar analíticamente donde la degeneración está entre sus funciones de base, y luego hacer supresiones apropiados en el conjunto base.

ENCAJES MULTIDIMENSIONALES

Si está midiendo una sola variable *y* en función de más de una variable - por ejemplo, un vector de variables **x** - entonces sus funciones de base serán funciones de un vector, $X_0(\mathbf{x}), \ldots, X_{M-1}(\mathbf{x})$. La función de mérito χ^2 es ahora.

$$\chi^2 = \sum_{i=0}^{N-1} \left[\frac{y_i - \sum_{k=0}^{M-1} a_k X_k(\mathbf{x}_i)}{\sigma_i} \right]^2 \qquad (4.1.2.24)$$

Toda la discusión anterior pasa a través de cambios, con x reemplazados por **x**. De hecho, anticipamos esto en la codificación de Fitsvd, lo que puede hacer que un encaje multidimensional lineal general sea tan fácilmente como un unidimensional. Aquí es ahora:
El segundo constructor, sobrecargado, en Fitsvd sustituye un xx matriz para lo que antes era un vector. Las filas de la matriz son los puntos de datos ndat. El número de columnas es la dimensionalidad del espacio (es decir, de **x**). Del mismo modo, las funciones suministradas por el usuario funks ahora toma un argumento vector, un **x**. Un ejemplo sencillo (ajuste de una función cuadrática a los datos en dos dimensiones) podría ser
Asegúrese de que el argumento de la función de usuario tiene exactamente el tipo "VecDoub_I &" (y no, por ejemplo, "VecDoub &" o "VecDoub_I"), ya que los estrictos compiladores de C++ son muy exigentes con esto.

Los dos constructores en Fitsvd comunican para adaptarse si los puntos de datos son unidimensionales o multidimensionales estableciendo ya sea xmd o x a NULL. Esto explica la rareza que *x* se une a los datos del usuario como un puntero, mientras *y* y *sig* estaban consolidados como referencias. (¡Sí, sabemos que esto es un poco de un truco!)

Capítulo V. ANÁLISIS, RESULTADOS DE LA INVESTIGACIÓN Y CONTRASTACIÓN DE HIPÓTESIS.

La pregunta pertinente aquí, podría ser la siguiente: ¿Cómo realizar el análisis, verificar los resultados de la investigación y contrastar las hipótesis, para una herramienta de tipo muy general, que genera modelos matemáticos a partir de conjuntos de datos específicos?

Una respuesta es que una herramienta de tipo general sólo se puede comprobar con estudios de casos, o sea, con problemas específicos concretos, y así analizar las respuestas concretas, específicas del problema, lamentablemente los autores de libros de metodología de investigación científica no dan pautas para tesis de tipo de construcción de modelos de tipo general, por lo que esta tesis podría ser una precursora en su campo, por lo menos para nuestro medio, no veo otra salida.

Los resultados de la investigación en estos casos, simplemente son los modelos matemáticos generados, es decir, el vector de constantes a[0,…, ma-1], donde ma, es el número de las funciones bases establecidas por el investigador o ingeniero de transportes, que servirán de factores a esas funciones bases respectivamente.

El siguiente problema a resolver es la contrastación de hipótesis, cómo hacerlo, para un modelo matemático general; la respuesta no es lo que por común hacen los investigadores, graficando los valores de los datos observados versus los generados por el modelo, y si las gráficas coinciden, muy bien y si no coinciden, hay que hacer algo de trampa, como aumentar el número de datos o de iteraciones, esto no prueba nada, y es un error metodológico bastante arraigado.

La respuesta correcta es muy simple, mostrando los errores (desviación estándar) de todos los coeficientes del vector de constantes a[0, …, ma-1], por tanto podremos saber con toda precisión ajustando a un modelo adecuado, por ejemplo a una distribución normal, la probabilidad exacta de confianza o el nivel de error aceptable de nuestro modelo, para cualesquiera de los valores de las constantes, podemos saber para una grado aceptable de probabilidad el rango de su variación con respecto a la media (el valor constante hallado del vector **a**). Hay varios caminos para realizar la contrastación de hipótesis, respondiendo a la pregunta general, ¿son estas dos distribuciones de datos, muestras de una misma población, o más específicamente, el modelo matemático generado, se ajusta a los datos observados, y con qué grado de precisión? Hay muchos caminos, pero todos conducen a Roma, podemos realizar la Prueba del t de Student, para una diferencia significativa de las medias de los datos observados, versus los generados, otro es realizar una prueba de Kolmogorov – Smirnov, para la función hallada y los datos observados, la misma prueba del Chi – Cuadrado, para los datos observados y los esperados (generados por el modelo), podemos realizar pruebas no paramétricas para dos distribuciones, una generada por el modelo y la otra para los datos observados, como el Tau de Kendall o el Coeficiente de Correlación de Spearman, etc.; por estas razones se ha desarrollado clases ad hoc de gran parte de la estadística inferencial y de toda la estadística descriptiva, para poder realizar estas pruebas de contrastación de hipótesis; ya que es posible realizar en términos económicos y en vista que no tengo las licencias de esos paquetes de software estadísticos, los de uso libre, o shareware

simplemente no valen.

Por tanto, para saber cómo está el budín, hay que probarlo, en este caso, generando modelos matemáticos para varios problemas reales, de diversos campos, ya que estás funciones o modelos, no se limitan a un determinado tipo de problemas, como es la generación de viajes.

Resolveremos tres problemas reales, hallaremos los modelos para cada uno de ellos (parámetros o factores que multiplicaran a las funciones bases dadas por el usuario, usando el método de la máxima verosimilitud, que ya se indicó) y, para la contrastación de las hipótesis nos bastará hallar las matrices de varianzas y covarianzas de dichos factores; luego realizaremos las pruebas de bondad de ajuste, por varios métodos.

Entonces creemos los programas de pruebas correspondientes, para los datos que se dan, las funciones bases que suministremos y mostremos los resultados arriba indicados para estos tres problemas que se dan a continuación y realicemos las pruebas de bondad de ajuste.

Para el primer caso de uso, usaremos la prueba t de Student, verificando si los datos observados son reflejados por el modelo o ley matemática generada; entonces tenemos

Hipótesis Nula: No hay una diferencia significativa entre las medias de las dos distribuciones, o en otras palabras, las dos distribuciones son muestras sacadas de la misma población, es decir, los datos observados de la variable dependiente versus los datos generados por el modelo, para esos mismos datos de las variables independientes, respectivamente; no tienen medias significativamente diferentes. En otras palabras el modelo se ajusta significativamente a los datos.

Hipótesis Alternativa: Hay una diferencia significativa entre las medias de las dos distribuciones, o en otros términos, los valores generados para la variable regresionada del modelo matemático generado, no corresponden con los datos observados de la variable dependiente para los datos de las variables independientes. En pocas palabras el modelo es malo o no sirve.

```
#include <iostream>
#include <conio.h>

#include "C:\nr_c304\code\nr3.h"
#include "C:\nr_c304\code\svd.h"
#include "C:\nr_c304\code\fitsvd.h"
#include "C:\nr_c304\code\sort.h"
#include "C:\nr_c304\code\moment.h"
#include "C:\nr_c304\code\gamma.h"
#include "C:\nr_c304\code\erf.h"
#include "C:\nr_c304\code\incgammabeta.h"
#include "C:\nr_c304\code\stattests.h"


VecDoub quadratic2d(VecDoub_I &xx);


int main()
```

```
{
	std::cout << "En la tabla 8.8, pag. 251, del libro \"Econometría\" de los\n"
		<< "autores Damodar N. Gujarati y Dawn C. Porter, se da los datos\n"
		<< "del PIB real, empleo y capital fijo real en México, desde el\n"
		<< "año 1955 hasta el año 1974 inclusive. Realice una regresión\n"
		<< "cuadrática para el PIB real mexicano\n\n";

	const int numDatos = 1974 - 1955 + 1;

	double empleoCapital[numDatos][2] = {
		{8310, 182113},
		{8529, 193749},
		{8738, 205192},
		{8952, 215130},
		{9171, 225021},
		{9569, 237026},
		{9527, 248897},
		{9662, 260661},
		{10334, 275466},
		{10981, 295378},
		{11746, 315715},
		{11521, 337642},
		{11540, 363599},
		{12066, 391847},
		{12297, 422382},
		{12955, 455049},
		{13338, 484677},
		{13738, 520553},
		{15924, 561531},
		{14154, 609825}
	};

	double pib[numDatos] = {
		114043,
		120410,
		129187,
		134705,
		139960,
		150511,
		157897,
		165286,
		178491,
		199457,
		212323,
		226977,
		241194,
		260881,
		277498,
		296530,
		306712,
		329030,
		354057,
		374977
	};

	MatDoub_I xx(numDatos, 2, &empleoCapital[0][0]);
```

```
      VecDoub_I yy(numDatos, pib);

      std::cout << "Año\tPIB\tEmpleo\tCapital\n";
    for(int i = 0; i < numDatos; i++)
          std::cout << 1955 + i  << "\t" << yy[i] << "\t" << xx[i][0] << "\t"
<<
                      xx[i][1] <<"\n";

      double sssig[numDatos];

      for(int i = 0; i < numDatos; i++)
            sssig[i] = 1.;

      VecDoub_I ssig(numDatos, sssig);

      Fitsvd miFitSVD(xx, yy, ssig, quadratic2d);
      miFitSVD.fit();


      std::cout << "\nValor de los parámetros o factores de las funciones
base\n";
      for(int i = 0; i < miFitSVD.ma; i++)
            std::cout << miFitSVD.a[i] << std::endl;

      std::cout << "\nEl PIB para el año 1973 es = "
                  << miFitSVD.a[0] + miFitSVD.a[1]*15924 + miFitSVD.a[2]*561531
                  + miFitSVD.a[3]*15924*15924 + miFitSVD.a[4]*15924*561531
                  + miFitSVD.a[5]*561531*561531 << std::endl << std::endl ;

      std::cout << " La matriz de varianzas y covarianzas es: " << std::endl;
      std::cout << miFitSVD.covar << std::endl;

      std::cout << " El valor de chi-cuadrado es: " << std::endl;
      std::cout << miFitSVD.chisq << std::endl;

      std::cout << "Prueba de Constrastación de Hipótesis por el método de la\n"
                   << " t de Student" << std::endl;

      /*    Hallemos los valores de la variable dependiente del modelo
            matemático    generado,    para    las    varibles    dependientes
correspondientes
            al de los datos observados.
      */

      VecDoub_IO y(numDatos);

      for(int i = 0; i < numDatos; i++)
            y[i]     =     miFitSVD.a[0]     +     miFitSVD.a[1]*xx[i][0]     +
miFitSVD.a[2]*xx[i][1]
                  +              miFitSVD.a[3]*xx[i][0]*xx[i][0]              +
miFitSVD.a[4]*xx[i][0]*xx[i][1]
                  + miFitSVD.a[5]*xx[i][1]*xx[i][1];

      std::cout << "Año\tPBI Generado\tPBI observado\n";
      for(int i = 0; i < numDatos; i++)
            std::cout << 1955 + i << "\t" << y[i] << "\t\t" << yy[i] << "\n";

      std::cout << "Para los dos conjuntos de datos de PBI, el generado y el\n"
                  << "observado,  probar  si  hay  o  no  hay  una  diferencia
significativa\n"
                  << "de los PBIs, con una probabilidad de error del 1%\n\n";
```

```
	double t;
	double prob;
	ttest(y, yy, t, prob);

	std::cout << "El valor de la variable t Student's es = " << t << std::endl;
	std::cout << "La probabilidad es = " << prob << std::endl;

	if(prob < 0.01)
		std::cout << "Para un nivel de confianza del 99% hay una diferencia\n"
			<< "significativa de los PBIs medias. El modelo no es bueno\n";
	else
		std::cout << "Para un nivel de confianza del 99%  no hay una diferencia\n"
			<< "significativa de los PBIs medias. El modelo es muy bueno\n";
	getche();

	return 0;
}

/*	Funciones de base dadas por el usuario. Es de tipo cuadrático, para
	mostrar que no sólo se pueden modelar problemas de funciones lineales.
	Sino de cualquier tipo. En este problema las variables independientes son
	dos: el empleo, en miles de personas y el capital en millones de pesos
	mexicanos de 1960; la variable dependiente es el PIB en millones de pesos
	mexicanos de 1960.
*/


VecDoub quadratic2d(VecDoub_I &xx ) {
	VecDoub ans(6);
	Doub x = xx[0], y = xx[1];
	ans[0] = 1.;
	ans[1] = x; ans[2] = y;
	ans[3] = x*x;	ans[4] = x*y; ans[5] = y*y;

	return ans;
}
```

Año	PIB*	Empleo†	Capital fijo‡
1955	114 043	8 310	182 113
1956	120 410	8 529	193 749
1957	129 187	8 738	205 192
1958	134 705	8 952	215 130
1959	139 960	9 171	225 021
1960	150 511	9 569	237 026
1961	157 897	9 527	248 897
1962	165 286	9 662	260 661
1963	178 491	10 334	275 466
1964	199 457	10 981	295 378
1965	212 323	11 746	315 715
1966	226 977	11 521	337 642
1967	241 194	11 540	363 599
1968	260 881	12 066	391 847
1969	277 498	12 297	422 382
1970	296 530	12 955	455 049
1971	306 712	13 338	484 677
1972	329 030	13 738	520 553
1973	354 057	15 924	561 531
1974	374 977	14 154	609 825

* Millones de pesos de 1960.
† Miles de personas.
‡ Millones de pesos de 1960.

Tabla 8.8. [10]

Aquí se muestran los resultados, es decir, el modelo matemático, y la contrastación de hipótesis:

En la tabla 8.8, pag. 251, del libro "Econometría" de los autores Damodar N. Gujarati y Dawn C. Porter, se da los datos del PIB real, empleo y capital fijo real en México, desde el año 1955 hasta el año 1974 inclusive. Realice una regresión cuadrática para el PIB real mexicano

Año PIB Empleo Capital

1955	114043	8310	182113
1956	120410	8529	193749
1957	129187	8738	205192
1958	134705	8952	215130
1959	139960	9171	225021
1960	150511	9569	237026
1961	157897	9527	248897
1962	165286	9662	260661
1963	178491	10334	275466
1964	199457	10981	295378
1965	212323	11746	315715
1966	226977	11521	337642
1967	241194	11540	363599
1968	260881	12066	391847
1969	277498	12297	422382
1970	296530	12955	455049
1971	306712	13338	484677
1972	329030	13738	520553
1973	354057	15924	561531
1974	374977	14154	609825

Valor de los parámetros o factores de las funciones base
-0.00703089
-21.531
1.32688
0.00231611
-8.1058e-05
2.84642e-07

El PIB para el año 1973 es = 354474

La matriz de varianzas y covarianzas es:

1.99546e-13	6.1139e-10	-1.80639e-11	-9.45318e-14	4.02334e-15	-3.63723e-17
6.1139e-10	1.87325e-06	-5.53451e-08	-2.89643e-10	1.23274e-11	-1.11445e-13
-1.80639e-11	-5.53451e-08	1.70267e-09	8.17715e-12	-3.50343e-13	3.06553e-15
-9.45318e-14	-2.89643e-10	8.17715e-12	4.93775e-14	-2.13422e-15	2.07468e-17
4.02334e-15	1.23274e-11	-3.50343e-13	-2.13422e-15	9.32638e-17	-9.19883e-19
-3.63723e-17	-1.11445e-13	3.06553e-15	2.07468e-17	-9.19883e-19	9.51822e-21

El valor de chi-cuadrado es:
1.22868e+08

Prueba de Constrastación de Hipótesis por el método de la t de Student

Año	PBI Generado	PBI observado
1955	109431	114043
1956	118664	120410
1957	127618	129187
1958	135383	134705
1959	143052	139960
1960	152694	150511
1961	160774	157897
1962	169246	165286
1963	181203	178491
1964	196699	199457
1965	213339	212323
1966	224512	226977
1967	239940	241194
1968	257799	260881
1969	275679	277498
1970	294668	296530
1971	310823	306712
1972	329498	329030
1973	354474	354057
1974	374617	374977

```
Para los dos conjuntos de datos de PBI, el generado y el
observado, probar si hay o no hay una diferencia significativa
de los PBIs, con una probabilidad de error del 1%

El valor de la variable t Student's es = -2.40198e-05
La probabilidad es = 0.999981
Para un nivel de confianza del 99%  no hay una diferencia
significativa de los PBIs medias. El modelo es muy bueno
```

Para el segundo caso de uso, usaremos la prueba del χ^2, verificando si los datos observados son reflejados por el modelo o ley matemática generada; entonces tenemos

Hipótesis Nula: Son los dos conjuntos de datos, los coches observados, contra los coches esperados por el modelo matemático generado, procedentes de la misma función de distribución, es decir, de la misma población. En otras palabras el modelo matemático generado, para un nivel de confianza es bueno o excelente. O

Hipótesis Alternativa: Podemos refutar, a un cierto nivel requerido de importancia, la hipótesis nula de que dos conjuntos de datos se han extraído de la misma función de distribución de la población. Refutar la hipótesis nula de hecho demuestra que los conjuntos de datos son de diferentes distribuciones. En pocas palabras el modelo es malo o no sirve.

```
#include <iostream>
#include <conio.h>

#include "C:\nr_c304\code\nr3.h"
#include "C:\nr_c304\code\svd.h"
#include "C:\nr_c304\code\fitsvd.h"
#include "C:\nr_c304\code\sort.h"
#include "C:\nr_c304\code\moment.h"
#include "C:\nr_c304\code\gamma.h"
#include "C:\nr_c304\code\erf.h"
#include "C:\nr_c304\code\incgammabeta.h"
#include "C:\nr_c304\code\stattests.h"


VecDoub multipleLineal6d(VecDoub_I &xx);


int main()
{
  // */



   std::cout << "En el fichero Problema-5-9 se presentan datos relativos a \n"
             << "veinticuatro países. El fichero consta de las siguientes
variables\n"
             << "referidas a cada país: \n"
             << "- Coches: Número de coches por persona.\n"
             << "- Pob: Población en millones de personas.\n"
             << "- Den: Densidad de población.\n"
             << "- Ingresos: Ingresos per capita en dólares U.S.A.\n"
             << "- Gasol: Precio de la gasolina en centavos U.S.A. por litro.\n"
```

```
            << "- Consumo: Toneladas de gasolina consumida por coche al año.\n"
            << "- Pasaj: Miles de pasajeros-kilómetros por persona que usan bús
o tren.\n"
            << "- País: País al que se refieren los datos de la fila.\n\n"

            << "Se quiere ajustar un modelo de regresión múltiple que explique
la\n"
            << "variable Coches en función de las variables explicativas: Pob,
Den,\n"
            << "Ingresos, Gasol, Consumo y Pasaj.\n";

      double coches[] = {
            0.27,
            0.3,
            0.42,
            0.28,
            0.24,
            0.33,
            0.35,
            0.08,
            0.34,
            0.2,
            0.3,
            0.18,
            0.43,
            0.3,
            0.4,
            0.28,
            0.1,
            0.18,
            0.34,
            0.32,
            0.014,
            0.27,
            0.53,
            0.09
      };

      const int numDatos = sizeof(coches) / sizeof(coches[0]);

      double regresoras[numDatos*6] = {

            7.5,      89,             7.7,        49,         1.1,        2.6,
            9.8,      323,            9.8,        59,         1,          1.6,
            23.5,     2,              8.7,        17,         2.8,        0.1,
            5.1,      119,            11,         56,         1.2,        1.9,
            4.8,      16,             7.1,        49,         1.2,        2.2,
            53.3,     97,             8.8,        61,         1,          1.5,
            61.3,     247,            10.4,       49,         1.1,        1.7,
            9.4,      71,             3.4,        56,         1.7,        0.7,
            0.2,      2,              9.8,        57,         1.2,        2,
            3.2,      46,             3.8,        40,         1.5,        0.3,
            56.7,     188,            4.6,        61,         0.6,        1.8,
            114.9,    309,            8.5,        49,         1.2,        3.5,
            0.4,      138,            9.8,        44,         1.6,        0.8,
            13.9,     412,            9.4,        56,         1,          1.5,
            3.1,      12,             5.9,        34,         1.3,        0.2,
            4.1,      13,             9.8,        61,         1,          1.7,
            9.8,      107,            1.8,        68,         0.7,        0.9,
            36.8,     73,             4,          44,         0.8,        1.3,
            8.3,      18,             10.6,       42,         1.3,        1.7,
            6.3,      153,            13.3,       56,         1.3,        2,
            42.7,     55,             1.2,        36,         3.3,        0.1,
            55.8,     229,            5.5,        35,         1.2,        1.6,
            218.2,    23,             9.7,        17,         2.7,        0.3,
```

```
            22,      86,             2.1,        40,         1.1,       2.1
      };

      MatDoub_I xx(numDatos, 6, regresoras);

      VecDoub_I yy(numDatos, coches);

      std::cout << "\nCoches\tPob\tDen\tIngres\tGasol\tConsumo\tPasaj\n";
      for(int i = 0; i < numDatos; i++)
            std::cout << yy[i] << "\t" << xx[i][0] << "\t" << xx[i][1] << "\t"
                        << xx[i][2] << "\t" << xx[i][3] << "\t" << xx[i][4]
                        << "\t" << xx[i][5] << "\n";

      double sssig[numDatos];

      for(int i = 0; i < numDatos; i++)
            sssig[i] = 1.;

      VecDoub_I ssig(numDatos, sssig);

      Fitsvd myFitSVD(xx, yy, ssig, multipleLineal6d);
      myFitSVD.fit();

      for(int i = 0; i < myFitSVD.ma; i++)
            std::cout << myFitSVD.a[i] << std::endl;

      std::cout << "coches = " << myFitSVD.a[0] << " + " << myFitSVD.a[1] << " * Pob + "
                  << myFitSVD.a[2] << " * Den + " << myFitSVD.a[3] << " * Ingreso + "
                  << myFitSVD.a[4] << " * Gasol + " << myFitSVD.a[5] << " * Consumo + "
                  << myFitSVD.a[6] << " * Pasaj\n" ;


      std::cout << " La matriz de varianzas y covarianzas es: " << std::endl;
      std::cout << myFitSVD.covar << std::endl;

      std::cout << " \nEl valor de chi-cuadrado es: " << std::endl;
      std::cout << myFitSVD.chisq << std::endl;

      std::cout << "\nPrueba de Constrastación de Hipótesis por el método del \n"
                   << " χ²" << std::endl;

      /*    Hallemos los valores de la variable dependiente (coches) del modelo
            matemático    generado,    para    las    varibles    dependientes correspondientes
            al de los datos observados.
      */



      VecDoub_IO y(numDatos);

      for(int i = 0; i < numDatos; i++)
            y[i]     =     myFitSVD.a[0]     +     myFitSVD.a[1]*xx[i][0]     + myFitSVD.a[2]*xx[i][1]
                  + myFitSVD.a[3]*xx[i][2] + myFitSVD.a[4]*xx[i][3]
                  + myFitSVD.a[5]*xx[i][4] + myFitSVD.a[6]*xx[i][5];
```

```
	std::cout << "Pais\tCoches Observados\tCoches Generados\n";
	for(int i = 0; i < numDatos; i++)
		std::cout << 1 + i << "\t" << yy[i] << "\t\t\t" << y[i] << "\n";

	std::cout << "\nPara los dos conjuntos de datos de Coches, el observado
y el\n"
		<< "generado, probar si hay o no hay una diferencia
significativa\n"
		<< "de las dos distribuciones, con una probabilidad de error
del 1%\n\n";


	double df;
	double chsq;
	double prob;
	chsone(yy, y, df, chsq, prob);

	std::cout << "El número de grados de libertad es = " << df << std::endl;
	std::cout << "La probabilidad es = " << prob << std::endl;
	std::cout << "El valor de Chi - Cuadrado es = " << chsq << std::endl;

	if(prob < 0.01)
		std::cout << "Para un nivel de confianza del 99% hay una
diferencia\n"
			<< "significativa de las dos distribuciones. El modelo
no es bueno\n";
	else
		std::cout << "Para un nivel de confianza del 99%  no hay una
diferencia\n"
			<< "significativa de las dos distribuciones. El modelo
es muy bueno\n";



	getche();

	return 0;
}

/*	Funciones de base dadas por el usuario. Es de tipo lineal.
	En este problema las variables independientes son dos seis:
	la variable dependiente es el número de coches por persona, en esos
	paises.
*/

VecDoub multipleLineal6d(VecDoub_I &xx) {
	VecDoub ans(7);
	Doub u = xx[0], v = xx[1], w = xx[2], x = xx[3], y = xx[4], z = xx[5];
	ans[0] = 1.;
	ans[1] = u;
	ans[2] = v;
	ans[3] = w;
	ans[4] = x;
	ans[5] = y;
	ans[6] = z;

	return ans;
}
```

Estos son los resultados del programa para este problema, se muestra el modelo y el análisis de contrastación de hipótesis, además se hará una interpretación suscinta y

cualitativa del modelo matemático generado:

En el fichero Problema-5-9 se presentan datos relativos a veinticuatro países. El fichero consta de las siguientes variables referidas a cada país:
- Coches: Número de coches por persona.
- Pob: Población en millones de personas.
- Den: Densidad de población.
- Ingresos: Ingresos per capita en dólares U.S.A.
- Gasol: Precio de la gasolina en centavos U.S.A. por litro.
- Consumo: Toneladas de gasolina consumida por coche al año.
- Pasaj: Miles de pasajeros-kilómetros por persona que usan bús o tren.
- País: País al que se refieren los datos de la fila.

Se quiere ajustar un modelo de regresión múltiple que explique la variable Coches en función de las variables explicativas: Pob, Den, Ingresos, Gasol, Consumo y Pasaj.

Coches	Pob	Den	Ingres	Gasol	Consumo	Pasaj
0.27	7.5	89	7.7	49	1.1	2.6
0.3	9.8	323	9.8	59	1	1.6
0.42	23.5	2	8.7	17	2.8	0.1
0.28	5.1	119	11	56	1.2	1.9
0.24	4.8	16	7.1	49	1.2	2.2
0.33	53.3	97	8.8	61	1	1.5
0.35	61.3	247	10.4	49	1.1	1.7
0.08	9.4	71	3.4	56	1.7	0.7
0.34	0.2	2	9.8	57	1.2	2
0.2	3.2	46	3.8	40	1.5	0.3
0.3	56.7	188	4.6	61	0.6	1.8
0.18	114.9	309	8.5	49	1.2	3.5
0.43	0.4	138	9.8	44	1.6	0.8
0.3	13.9	412	9.4	56	1	1.5
0.4	3.1	12	5.9	34	1.3	0.2
0.28	4.1	13	9.8	61	1	1.7
0.1	9.8	107	1.8	68	0.7	0.9
0.18	36.8	73	4	44	0.8	1.3
0.34	8.3	18	10.6	42	1.3	1.7
0.32	6.3	153	13.3	56	1.3	2
0.014	42.7	55	1.2	36	3.3	0.1
0.27	55.8	229	5.5	35	1.2	1.6
0.53	218.2	23	9.7	17	2.7	0.3
0.09	22	86	2.1	40	1.1	2.1

0.469733
0.000590876
-0.000107003
0.0322364
-0.00405042
-0.107598
-0.0714819

coches = 0.469733 + 0.000590876 * Pob + -0.000107003 * Den + 0.0322364 * Ingreso + -0.00405042 * Gasol + -0.107598 * Consumo + -0.0714819 * Pasaj

La matriz de varianzas y covarianzas es:
3.14639 -0.00152354 9.21511e-05 -0.0178116 -0.0365635 -0.735728 -0.138127
-0.00152354 2.68546e-05 -2.96841e-06 7.52873e-06 4.59324e-05 -0.000490829 -0.000384439
9.21511e-05 -2.96841e-06 4.55242e-06 -1.47776e-05 -1.01686e-05 0.000132381 -8.76707e-05

```
-0.0178116 7.52873e-06      -1.47776e-05     0.0045157  4.44007e-05
      -0.00417643      -0.00725787
-0.0365635 4.59324e-05      -1.01686e-05     4.44007e-05
      0.000614279      0.0062081  -0.00114516
-0.735728  -0.000490829     0.000132381      -0.00417643      0.0062081
      0.266833   0.075318
-0.138127  -0.000384439     -8.76707e-05     -0.00725787      -
0.00114516 0.075318   0.11627


El valor de chi-cuadrado es:
0.0446865

Prueba de Constrastación de Hipótesis por el método del
 X²
Pais  Coches Observados     Coches Generados
1     0.27              0.21018
2     0.3               0.295934
3     0.42              0.386582
4     0.28              0.322857
5     0.24              0.214887
6     0.33              0.312631
7     0.35              0.376435
8     0.08              0.117517
9     0.34              0.282598
10    0.2               0.244342
11    0.3               0.191105
12    0.18              0.200795
13    0.43              0.363559
14    0.3               0.295239
15    0.4               0.368587
16    0.28              0.310488
17    0.1               0.107019
18    0.18              0.255388
19    0.34              0.382903
20    0.32              0.376163
21    0.014             0.0197258
22    0.27              0.270247
23    0.53              0.528078
24    0.09              0.11074

Para los dos conjuntos de datos de Coches, el observado y el
generado, probar si hay o no hay una diferencia significativa
de las dos distribuciones, con una probabilidad de error del 1%

El número de grados de libertad es = 23
La probabilidad es = 1
El valor de Chi - Cuadrado es = 0.186653
Para un nivel de confianza del 99%  no hay una diferencia
significativa de las dos distribuciones. El modelo es muy bueno
```

Se observa que prácticamente el error es muy pequeñísimo y que el modelo es muy bueno, en términos estadísticos.

Un análisis cualitativo del modelo matemático generado es que la posición de los coches para esos 24 países desarrollados se explica así:

Los habitantes tienen casi medio coche, que no depende de variable alguna; y que esta cantidad de coches aumenta con el tamaño de la población del país, disminuye con su densidad, es decir, disminuye si el número de habitantes está concentrada en áreas pequeñas; aumenta con el ingreso de las personas; disminuye con el costo de los

combustibles; también disminuye con el consumo de combustible por los coches y por último disminuye cuando los pasajes por bus o tren aumenta.

Esta ley cualitativa explica muy bien el funcionamiento de la posesión de coches.

Para el tercer caso de uso, se predice la cantidad de viajes para un cambio en las variables dependientes. Usaremos, nuevamente la prueba del χ^2, para la contrastación de las hipótesis, verificando si los datos observados son reflejados por el modelo o ley matemática generada; entonces tenemos

Hipótesis Nula: Son los dos conjuntos de datos, los coches observados, contra los coches esperados por el modelo matemático generado, procedentes de la misma función de distribución, es decir, de la misma población. En otras palabras el modelo matemático generado, para un nivel de confianza es bueno o excelente. O

Hipótesis Alternativa: Podemos refutar, a un cierto nivel requerido de importancia, la hipótesis nula de que dos conjuntos de datos se han extraído de la misma función de distribución de la población. Refutar la hipótesis nula de hecho demuestra que los conjuntos de datos son de diferentes distribuciones. En pocas palabras el modelo es malo o no sirve.

```
#include <iostream>
#include <conio.h>

#include "C:\nr_c304\code\nr3.h"
#include "C:\nr_c304\code\svd.h"
#include "C:\nr_c304\code\fitsvd.h"
#include "C:\nr_c304\code\sort.h"
#include "C:\nr_c304\code\moment.h"
#include "C:\nr_c304\code\gamma.h"
#include "C:\nr_c304\code\erf.h"
#include "C:\nr_c304\code\incgammabeta.h"
#include "C:\nr_c304\code\stattests.h"

VecDoub viajes4d(VecDoub_I &xx);

int main()
{


      std::cout << "En el libro \"Modelos de Transporte\" de Juan de Dios Ortuzar\n"
                  << " y Luis G. Willumsen, se propone el siguiente problema, pag
248\n\n"
                  << "4.1 Se considera una zona con las siguientes
características:\n"
                  << " Tipo de hogar\tN°\tIngresos\tHabitantes\tViajes/dia\n"
                  << " 0 coches\t180\t4000\t\t4\t\t6\n"
                  << " 1 coche\t80\t18000\t\t4\t\t8\n"
                  << " 2 o + coches\t40\t50000\t\t6\t\t11\n\n"
                  << "Como consecuencia de una disminución en los impuestos de\n"
                  << "importación y un incremento de los ingresos reales del 30%,
se\n"
                  << "espera que en los proximos cinco años el 50% de las
familias\n"
                  << "actualmente sin coche adquiera uno. Se desea estimar
cuántos\n"
                  << "viajes generaría la zona en este caso; verifique que el
método\n"
```

```
                << "utilizado sea realmente el mejor disponible\n";

        double regresores[9] = {
                0,      4000,  4,
                1,      18000, 4,
                2,      50000, 6
        };

        double viajes [3] = {
                6,
                8,
                11
        };

        std::cout << "Se modela utilizando regresión lineal múltiple\n";

        MatDoub_I xx(3, 3, regresores);


        VecDoub_I yy(3, viajes);

        std::cout << "\nViajes\tCoches\tIngreso\tPersonas\n";
        for(int i = 0; i < 3; i++)
                std::cout << yy[i] << "\t" << xx[i][0] << "\t" << xx[i][1] << "\t"
                                        << xx[i][2] << "\n";

        std::cout << "\n";

        double sssig[3];

        for(int i = 0; i < 3; i++)
                sssig[i] = 1.;

        VecDoub_I ssig(3, sssig);

        Fitsvd xFitSVD(xx, yy, ssig, viajes4d);
        xFitSVD.fit();

        for(int i = 0; i < xFitSVD.ma; i++)
                std::cout << xFitSVD.a[i] << std::endl;

        std::cout << "\nviajes = " << xFitSVD.a[0] << " + " << xFitSVD.a[1] << " *
Coches + "
                << xFitSVD.a[2] << " * Ingreso + " << xFitSVD.a[3] << " *
Personas\n";

        std::cout << "Viajes totales en la zona, en la nueva situación = :\n";
        std::cout << 90*(xFitSVD.a[0] + xFitSVD.a[1]*(0) + xFitSVD.a[2]*(1.3*4000)+
xFitSVD.a[3]*4) +
                                170*(xFitSVD.a[0] + xFitSVD.a[1]*(1) +
xFitSVD.a[2]*(1.3*18000)+ xFitSVD.a[3]*4) +
                                40*(xFitSVD.a[0] + xFitSVD.a[1]*(2) +
xFitSVD.a[2]*(1.3*50000)+ xFitSVD.a[3]*6)
                                << "\n";




        VecDoub_IO y(3);

        for(int i = 0; i < 3; i++)
                y[i] = xFitSVD.a[0] + xFitSVD.a[1]*xx[i][0] + xFitSVD.a[2]*xx[i][1]
                        + xFitSVD.a[3]*xx[i][2];

        std::cout << "Familia\tViajes Observados\tViajes Generados\n";
        for(int i = 0; i < 3; i++)
                std::cout << 1 + i << "\t" << yy[i] << "\t\t\t" << y[i] << "\n";

        std::cout << "\nPara los dos conjuntos de datos de Viajes, el observado y el\n"
```

```
                    << "generado, probar si hay o no hay una diferencia
significativa\n"
                    << "de las dos distribuciones, con una probabilidad de error del
1%\n\n";


      double df;
      double chsq;
      double prob;
      chsone(yy, y, df, chsq, prob);

      std::cout << "El número de grados de libertad es = " << df << std::endl;
      std::cout << "La probabilidad es = " << prob << std::endl;
      std::cout << "El valor de Chi - Cuadrado es = " << chsq << std::endl;

      if(prob < 0.01)
            std::cout << "Para un nivel de confianza del 99% hay una diferencia\n"
                        << "significativa de las dos distribuciones. El modelo no
es bueno\n";
      else
            std::cout << "Para un nivel de confianza del 99%  no hay una
diferencia\n"
                        << "significativa de las dos distribuciones. El modelo es
muy bueno\n";



      getche();

      return 0;
}

/*    Funciones de base dadas por el usuario. Es de tipo lineal.
       En este problema las variables independientes son tres:
      Posesión de Coches, Ingresos, y Cantidad de Personas; todas por familia;
      la variable dependiente es el número de viajes por familia.
*/


VecDoub viajes4d(VecDoub_I &xx) {
      VecDoub ans(4);
      Doub x = xx[0], y = xx[1], z = xx[2];
    ans[0] = 1.;
      ans[1] = x;
      ans[2] = y;
      ans[3] = z;

      return ans;
}
```

En el libro "Modelos de Transporte" de Juan de Dios Ortuzar
 y Luis G. Willumsen, se propone el siguiente problema, pag 248

4.1 Se considera una zona con las siguientes características:

Tipo de hogar	N°	Ingresos	Habitantes	Viajes/dia
0 coches	180	4000	4	6
1 coche	80	18000	4	8
2 o + coches	40	50000	6	11

Como consecuencia de una disminución en los impuestos de importación y un incremento de los ingresos reales del 30%, se espera que en los proximos cinco años el 50% de las familias actualmente sin coche adquiera uno. Se desea estimar cuántos viajes generaría la zona en este caso; verifique que el método utilizado sea realmente el mejor disponible
Se modela utilizando regresión lineal múltiple

Viajes	Coches	Ingreso	Personas
6	0	4000	4
8	1	18000	4
11	2	50000	6

1.65104
3.02767
-7.34051e-05
1.16065

viajes = 1.65104 + 3.02767 * Coches + -7.34051e-05 * Ingreso + 1.16065 * Personas

Viajes totales en la zona, en la nueva situación = :
2220.64

Familia	Viajes Observados	Viajes Generados
1	6	6
2	8	8
3	11	11

Para los dos conjuntos de datos de Viajes, el observado y el generado, probar si hay o no hay una diferencia significativa de las dos distribuciones, con una probabilidad de error del 1%

El número de grados de libertad es = 2
La probabilidad es = 1
El valor de Chi - Cuadrado es = 2.22048e-22
Para un nivel de confianza del 99% no hay una diferencia significativa de las dos distribuciones. El modelo es muy bueno

Conclusiones y recomendaciones

Conclusiones

- Los modelos matemáticos generados tienen un nivel de confianza de más del 99%, por lo que se pueden aplicar dichos modelos con seguridad suficiente.

- Los modelos matemáticos generados no tienen restricciones de linealidad de los datos.

- Se ha mejorado la metodología para el caso de tesis que tengan como objetivo crear modelos u herramientas.

- En la contrastación de hipótesis se ha sido riguroso, no se ha abusado de las gráficas, para poder “demostrar” la semejanza de los datos observados con los generados.

- Las aplicaciones de la tesis no sólo se encuadran en la generación de viajes, sino en variados problemas prácticos y técnicos. Prácticamente toda la Econometría está resuelta.

- Se comprueba el desarrollo en espiral (volver a lo mismo, pero en un nivel superior), ya que se parte de unas funciones bases y datos, para generar una función matemática con un alto grado de precisión que modelan esos datos.

Recomendaciones

- El Ingeniero, el científico o todo investigador debe estar familiarizado con algún lenguaje de programación. El tesista sugiere el C++.

- El Ingeniero de Transportes o el usuario, debe conocer muy bien el campo de aplicación para el que quiera modelar sus datos, ya que se requiere las funciones bases, que se debe proporcionar al programa, por el usuario.

Bibliografía

[1] Abramowitz, M., y Stegun, I. A. 1964, *Handbook of Mathematical* (Washington: National Bureau of Standards)

[2] Bevington, P. R., y Robinson, D.K. 2002, *Data Reduccion and Error Analysis for the Physical Sciences*, 3ra ed. (New York: McGraw - Hill)

[3] Caballero R., A. E., 2011, *Metodología integral innovadora para planes y tesis* (Lima: Instituto Metodológico Alen Caro E.I.R.L.)

[4] Cal y Mayor, R. y Cárdenas, J. 2007*, Ingeniería de Tránsito*, 8va ed. (México D.F.: Alfaomega Grupo Editor)

[5] Chan, T. F.,Golub, G.H., y LeVeque, R.J 1983, "Algorithms for Computing the Sample Variance: Analysis and Recommendations," American Statistician

[6] Fernández A., R., 2011, *Elementos de la Teoría del Tráfico Vehicular* (Lima: Fondo Editorial de la Pontificia Universidad Católica del Perú)

[7] Garber, N. J. y Hoel, L. A., 2007, *Ingeniería de Tránsito y Carreteras*, 3ra ed. (México D. F.: Cengage Learning Editores, S. A.)

[8] Gelman, A., Carlin, J.B., Stern, H. S., y Rubin, D. B. 2004, *Bayesian Data Analysis*, 2da ed. (Boca Raton, FL: Chapman & Hall/CRC)

[9] Grima, P. 2012, *La certeza absoluta y otras ficciones* (Rodesa, Navarra, España: RBA Contenidos Editoriales y Audiovisuales, S.A.)

[10] Gujarati, D.N., y Porter, D. C., 2008, *Econometría* (México D.F.: McGraw-Hill)

[11] Koffman, E. B. y Wolfgang, P. A. T., 2008, *Estructuras de Datos con C++. Objetos, Abstracciones y Diseño* (México D. F.: McGraw - Hill)

[12] Lee, R. C. T., Tseng, S. S. y Tsai, Y. T. 2007, *Introducción al diseño y análisis de algoritmos. Un enfoque estratégico* (México D. F.: McGraw - Hill)

[13] Linares F., A. 2003, *Programación Computacional del Método de los Elementos Finitos en la Ingeniería Estructural* (Ayacucho, Perú: Tesis UNSCH)

[14] Mackay, D. J. C. 2003, *Information Theory, Inference, and Learning Algorithms* (Cambridge, UK: Cambridge University Press)

[15] Monahan, J. F. *Numerical Methods of Statistics* (Cambridge, UK: Cambridge University Press)

[16] Norusis, M. J., 2006, *SPSS 14.0 Guide to Data Analysis* (Englewood Cliffs, NJ: Prentice - Hall)

[17] Ortuzar, J. D. y Willumsen L.G. 2008, *Modelos de Transporte* (Madrid: Ediciones de la Universidad de Cantabria)

[18] Pearson, E., y Johnson, N. 1968, *Tables of the Incomplete Beta Function* (Cambridge, UK: Cambridge University Press).

[19] Seber, G. A. F., y Wild, C. J. 2003, *Nonlinear Regression* (Hoboken, NJ: Wiley)

[20] Spiegel, M. R., Schiller, J. y Srinivasan, R. A. 2000, Schaum's Outline of *Theory and Problem of Probability and Statistics*, 2da. Ed. (New York: McGraw-Hill)

[21] Stroustrup, B. 2002, *El Lenguaje de Programación C++* (Madrid: Pearson Educación S. A.)

[22] Wilkinson, J. H. 1965, *The algebraic Eigen Value Problem* (New York: Oxford University Press)