ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE CIENCIAS
ESCUELA DE FÍSICA Y MATEMÁTICA
INGENIERIA ESTADÍSTICA INFORMÁTICA

"ELABORACIÓN DE UN SOFTWARE LIBRE ESTADÍSTICO COMO APOYO ACADÉMICO AL CUARTO NIVEL DE LA CARRERA DE INGENIERIA EN ESTADÍSTICA INFORMÁTICA"

TESIS DE GRADO

PREVIA LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO EN ESTADÍSTICA INFORMÁTICA

PRESENTADO POR:

LUIS FABIÁN CABEZAS ARÉVALO

RIOBAMBA – ECUADOR.
2010

AGRADECIMIENTO

Un sincero agradecimiento a la ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO, noble institución que me permitió consagrar mis estudios superiores. Al Mat. Alberto Vilañez Director de tesis, Mat Marcelo Cortez, Ing. Patricio Londo, asesores, por su orientación en el desarrollo de esta tesis.

DEDICATORIA

A DIOS, por darme la vida, salud y fuerzas para luchar cada día y enfrentar las adversidades; a mi madre María Esperanza Arévalo Rodríguez y mi abuelita Luz María Rodríguez Bonilla pilares fundamentales en mi vida y formación estudiantil pues está reflejado el sacrificio y apoyo incondicional que siempre me brindaron, mi tía María Constancia Arévalo, mis hermanos Pedro y Viviana, y mi abuelito Juan Arévalo Arévalo (+) que sin estar presente, siempre me ha dado la fuerza para seguir adelante.

NOMBRE	FIRMA	FECHA
Dra. Yolanda Díaz **Decana de la Facultad de Ciencias**	……	……
Dr. Richard Pachacama **Director Escuela de Física y Matemática**	……	……
Mat. Alberto Vilañez **Director de Tesis**	……	……
Mat. Marcelo Cortez **Miembro del Tribunal**	……	……
Ing. Patricio Londo **Miembro del Tribunal**	……	……
Sr. Carlos Rodríguez **Director Centro de Documentación**	……	……

NOTA DE TESIS ESCRITA: ……..

“Yo, Luis Fabián Cabezas Arévalo, soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis, y el patrimonio intelectual de la Tesis de Grado pertenecen a la Escuela Superior Politécnica de Chimborazo”

……

Luis Cabezas

ABREVIATURAS

API	Interfaz de programación de aplicaciones.
C++	Es un lenguaje C avanzado, basado en objetos.
Commodore	Ordenador doméstico de 8 bits fabricado.
FLOSS	Software libre y de código abierto.
GNU	General Public License.
HTML	Lenguaje de descripción de páginas habitual en Internet.
IDE	Integrated Development Environment.
JDK	Java Development Kit.
JFK	Java Foundation Classes.
KDE	K Desktop Environment.
GUI	Interfaces Gráficas de Usuario.
SUN	Empresa que facilita software gratis.

INDICE

INDICE DE GRÁFICOS

INDICE DE ECUACIONES

CAPITULO I

1 INTRODUCCIÓN

En la actualidad tanto el sector educativo como empresarial ven la necesidad de tener en su orgánico estructural un departamento de estadística, el cual debe estar equipado con un software de acuerdo a sus necesidades, y como es de conocimiento público dado en un decreto ejecutivo es primordial la utilización de software libre.

La carrera en estadística informática de la Escuela Superior Politécnica de Chimborazo, formar profesionales, capaces de manejar y aplicar el software libre en instituciones tanto del sector público y privado.

El software estadístico en la actualidad es fácil de manejar por el usuario por su interface gráfica.

Existe empresas que se han dedicado a la creación de software libre, pero la mayoría de estas lo han hecho solo con fines lucrativos, son muy pocas las empresas que lo hacen libremente, por esto es que no existe tanta diversidad de software para las diferentes especialidades.

La escuela de Física y Matemática, carrera de Estadística Informática utilizan software propietarios estadísticos como por ejemplo el SPSS, Minitab, Systab, STATA, jmp que es muy indispensable en la formación profesional.

Por tal razón hemos visto la necesidad de prescindir de los software pagados y dar énfasis a la utilización del software creados por sus propios estudiantes.

1.1 OBJETIVOS

1.1.1 OBJETIVO GENERAL:

Elaboración de un software libre estadístico con interface gráfica para apoyo académico a docentes y estudiantes del cuarto nivel de la carrera de Estadística Informática.

1.1.2 OBJETIVOS ESPECÍFICOS:

- Creación de un software estadístico utilizando un desarrollador libre como NetBeans IDE de Java.

- Crear el software para realizar Estadística Descriptiva y Estadística Inferencial.

- Conocer las bondades al programar con el desarrollador libre NetBeans IDE de Java.

- Conocer el manejo del software estadístico a docentes y estudiantes de la Carrera de Estadística Informática.

CAPITULO II

2 PARTE TEORICA

2.1 HERRAMIENTAS ESTADISTICAS

2.1.1 INTRODUCCIÓN

En esta sección hablaremos de las diferentes formas de representar los datos tanto en forma descriptiva como inferencial por ejemplo el diagrama de barras, histogramas, diagrama circular, diagrama de líneas, prueba t con una media, prueba t con dos medias, con variables independientes o dependientes.

2.2 ESTADÍSTICA

La Estadística es una disciplina que utiliza recursos matemáticos para organizar y resumir una gran cantidad de datos obtenidos de la realidad, e inferir conclusiones respecto de ellos.

La estadística puede aplicarse a cualquier ámbito de la realidad, y por ello es utilizada en física, química, biología, medicina, astronomía, psicología, sociología, lingüística, demografía, etc.

2.3 POBLACIÓN ESTADÍSTICA

Es un conjunto de personas, entidades u objetos del cual se quiere saber algo que nos interesa para tomar una determinación acertada.

Para facilitar el estudio de las poblaciones éstas se clasifican en:

Población finita.

Población infinita.

2.4 MUESTRA

Una muestra es un conjunto de medidas u observaciones tomadas a partir de una población dada; es un subconjunto de la población. Desde luego, el número de observaciones en una muestra es menor que el número de posibles observaciones en la población, de otra forma, la muestra será la población misma. Las muestras se toman debido a que no es factible desde el punto de vista económico usar a toda la población. En algunos casos es imposible recolectar todas las posibles observaciones en la población.

2.4.1 MUESTRA ALEATORIA

Es aquella que se obtiene de tal manera que cada posible observación disponible en la población, tiene la misma probabilidad de ser seleccionada.

Para poder obtener estas muestras es necesario que no intervenga la preferencia del investigador por algún elemento de la población; es decir, cada elemento de la población deberá tener igual oportunidad de ser seleccionado.

Los promedios y proporciones muestrales son características medibles de las muestras respectivas y se les llama estadísticas o estadígrafos.

2.5 CLASIFICACIÓN DE LA ESTADÍSTICA

En base a lo que se ha dicho se concluye, que la Estadística como disciplina o área de estudio comprende técnicas descriptivas como inferenciales. Incluye la observación y tratamiento de datos numéricos y el empleo de los datos estadísticos con fines inferenciales. Para su estudio se clasifica de la siguiente forma:

2.5.1 ESTADÍSTICA DESCRIPTIVA.

Es el estudio que incluye la obtención, organización, presentación y descripción de información cuantitativa y cualitativa.

2.5.2 ESTADÍSTICA INFERENCIAL

La inferencia estadística es una técnica mediante la cual se obtienen generalizaciones o se toman decisiones en base a una información parcial o completa obtenida mediante técnicas descriptivas.

2.6 TIPOS DE VARIABLES

Variables Cualitativas: es cuando solamente se busca en ella una cualidad o un atributo.

Ejemplo: Color de ojos.

Variables Cuantitativas: Si la variables que se estudia puede ser expresada numéricamente.

Ejemplo: Estado de cuenta.

2.6.1 VARIABLE CUANTITATIVA

Es aquella que se puede asociar con un número con el cual podemos realizar operaciones o comparaciones.

2.6.1.1 CLASIFICACIÓN DE LAS VARIABLES CUANTITATIVAS

Variables discretas: Solo se puede asumir ciertos valores y suelen haber intervalos entre los valores.

Variables Continuas: Pueden tomar cualquier valor en un rango específico (orden decimal).

2.7 PRESENTACIÓN DE DATOS

Una vez que se han obtenido los datos y que se ha hecho el estudio de los valores que pueden tomar las variables, la primera tarea de la Estadística es la de ordenar y presentar los datos en tablas que permitan ver la tendencia de los mismos. Ordenados los datos se facilita su representación en diagramas y gráficas de diferentes tipos.

2.7.1 DISTRIBUCIÓN DE FRECUENCIAS

Los datos agrupados en tablas, nos permiten ver con facilidad el número de observaciones iguales o comprendidos en un intervalo, a este número de repeticiones iguales de la variable se llama frecuencia y se denota por fi. Otros valores relacionados con la frecuencia son:

La frecuencia relativa que se denota por fr.

La frecuencia acumulada que se denota por Fi.

La frecuencia relativa acumulada que se denota Fr.

2.7.2 LA FRECUENCIA

Frecuencia es el número de veces que se repite la misma observación. Se simboliza con fi.

2.7.3 LA FRECUENCIA RELATIVA

Frecuencia relativa (fr) es la proporción de elementos que pertenecen a una categoría y ésta se obtiene dividiendo su frecuencia absoluta entre el número total de elementos de la muestra.

2.7.4 LA FRECUENCIA ACUMULADA

Frecuencia acumulada (Fi) de una clase es la que se obtiene sumando las frecuencias de las clases anteriores con la frecuencia de ésta.

2.8 GRÁFICAS

Al representar en una gráfica la información concentrada en la tabla de frecuencias, ésta es un recurso visual que nos permite tener una idea clara, precisa, global y rápida acerca de las observaciones de una muestra o población.

Existen muchos tipos de gráficas en las que se pueden representar la frecuencia absoluta (fi), relativa (fr) y acumulada (Fi) y con ellas podemos estimar algunos valores con la simple observación.

Los diferentes tipos de gráfica que podemos usar para representar las observaciones de un determinado problema y la selección de este tipo, dependen de la variable en estudio. Si la

variable en estudio es del tipo cualitativo, los gráficos pueden ser: a) De barras; horizontales o verticales. b) Circulares. c) Pictogramas, etcétera.

Si la variable en estudio es de tipo cuantitativo, los gráficos que podemos usar para su representación gráfica son: a) Histogramas. b) Polígonos de frecuencias que a continuación analizaremos.

2.8.1 HISTOGRAMA

Histograma es la representación gráfica en el plano coordenado de las características concentradas en la tabla de frecuencias de una variable continua.

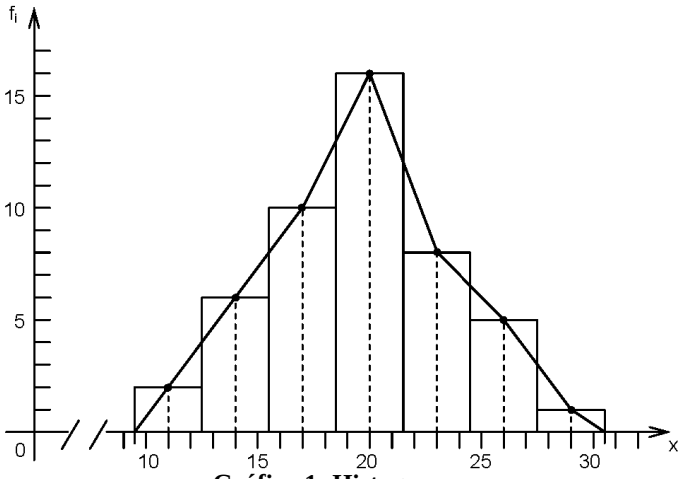


Gráfico 1: Histograma

2.8.2 POLÍGONO DE FRECUENCIA

El polígono de frecuencia se construye a partir de los datos de la tabla de frecuencias. Sobre el eje horizontal se levanta por el punto medio segmentos verticales punteados que terminan a la altura de su frecuencia de clase, se unen los puntos superiores con un segmento de recta que empieza medio punto antes del límite superior de la última clase.

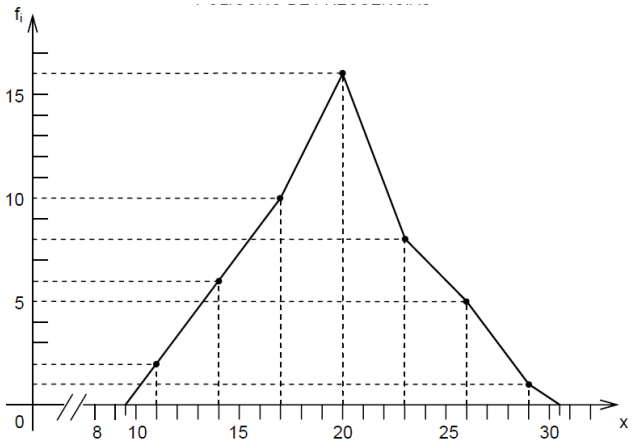


Gráfico 2: Polígono de Frecuencias

2.8.3 GRÁFICAS CIRCULARES

Una forma de representar datos u observaciones de una variable cualitativa es mediante un diagrama circular.

Para trazar la gráfica, se hace una distribución proporcional de las frecuencias del problema con respecto a la circunferencia determinando sectores circulares para cada categoría.

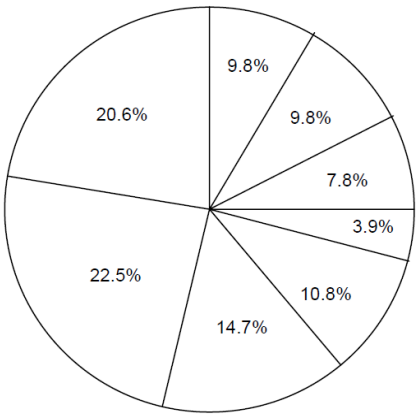


Gráfico 3: Diagrama Circular

2.8.4 GRÁFICAS DE BARRAS

Otra forma de representar gráficamente las puntuaciones de un problema, es mediante una gráfica de barras.

Para construir una gráfica de barras se trazan ejes coordenados; en el eje horizontal se representan los valores de la variable y se traza un segmento perpendicular por cada valor. Si se usan barras, éstas deberán tener el mismo ancho de la base. En el eje vertical se representa la frecuencia de cada clase usando una escala conveniente para cada puntuación. La frecuencia marca la altura de cada segmento perpendicular o barra.

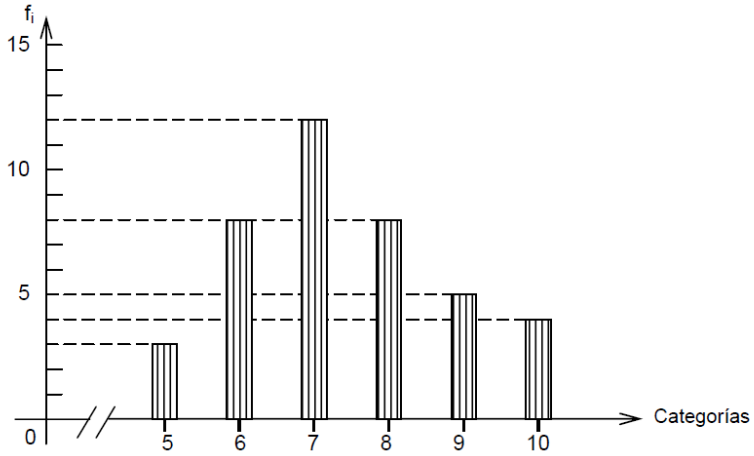


Gráfico 4: Diagrama de Barras

2.8.5 GRÁFICO DE LÍNEAS

Una forma de representar gráficamente los valores de la variable de un problema en estudio, es mediante un gráfico de líneas.

Para trazar la gráfica de líneas se usa el plano coordenado; en el eje horizontal se representa a la variable y en el eje vertical la frecuencia. Se determinan los puntos de corte del valor

de la variable con su frecuencia y se unen estos puntos obteniéndose la gráfica de línea que nos muestra con claridad los cambios que experimentó la variable.

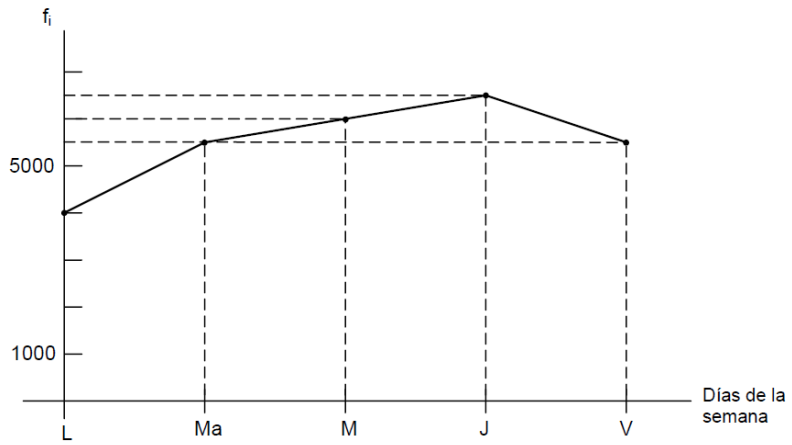


Gráfico 5: Diagrama de líneas

2.9 MEDIDAS DESCRIPTIVAS

Con estas medidas se persigue reducir en pocas cifras significativas el conjunto de observaciones de una variable y describir con ellas ciertas características de los conjuntos, logrando una comparación más precisa de los datos que la que se puede conseguir con tablas y gráficas.

2.10 MEDIDAS DE TENDENCIA CENTRAL

2.10.1 LA MEDIA ARITMÉTICA

La media es una medida de posición que dan una descripción compacta de cómo están centrados los datos y una visualización más clara del nivel que alcanza la variable, pueden servir de base para medir o evaluar valores extremos o raros y brinda mayor facilidad para efectuar comparaciones.

Es importante poner en relieve que la notación de promedio lleva implícita la idea de variación y que este número promedio debe cumplir con la condición de ser representativo de conjunto de datos.

La media como punto típico de los datos es el valor alrededor del cual se agrupan los demás valores de la variable, y su fórmula es:.

$$\overline{x} = \frac{\sum_i x_i}{n}$$

Ecuación 1: Calculo de la media Aritmética

2.10.2 LA MEDIANA

Es el valor de la observación que ocupa la posición central de un conjunto de datos ordenados según su magnitud. Es el valor medio o la media aritmética de los valores medios. La mediana es un valor de la variable que deja por debajo de él un número de casos igual al que deja por arriba.

Geométricamente la mediana es el valor de la variable que corresponde a la vertical que divide al histograma en dos áreas iguales.

Cuando determinados valores de un conjunto de observaciones son muy grandes o pequeños con respecto a los demás, entonces la media aritmética se puede distorsionar y perder su carácter representativo, en esos casos es conveniente utilizar la mediana como medida de tendencia central, y su fórmula es:

$$\tilde{\tilde{x}} = \begin{cases} x_{(n+1)/2,} & \text{si n es impar} \\ \frac{1}{2}\left(x_{n/2} + x_{n/2+1}\right), & \text{si n es par} \end{cases}$$

Ecuación 2: Calculo de la mediana.

2.10.3 LA MODA

Es el valor de un conjunto de datos que ocurre más frecuentemente, se considera como el valor más típico de una serie de datos.

Para datos agrupados se define como Clase Modal el intervalo que tiene más frecuencia.

La moda puede no existir o no ser única, las distribuciones que presentan dos o más máximos relativos se designan de modo general como bimodales o multimodales.

2.11 MEDIDAS DE DISPERSIÓN

Un rasgo principal de los datos es su dispersión o amplitud, que se refiere a su variabilidad, a la evaluación de cuán separados o extendidos están estos datos o bien cuanto difieren unos de otros.

Variación: es el grado en que los datos numéricos tienden a extenderse alrededor de un valor, generalmente el valor medio.

2.11.1 RANGO

Mide la dispersión de la totalidad de los datos. Es la más obvia de las medidas ya que es la distancia entre los valores máximo y mínimo.

El rango o recorrido da alguna idea del grado de variación que ocurre en la población, pero con frecuencia los resultados pueden ser engañosos, pues este depende de los valores extremos e ignora la variación de las demás observaciones. Está afectado por ocurrencias raras o extraordinarias.

2.11.2 VARIANZA

Otro tratamiento para evadir la suma cero de las desviaciones de las observaciones respecto a su Media Aritmética, consiste en recurrir al proceso de elevar al cuadrado estas desviaciones y sumar los cuadrados, dividiendo la suma por el número de casos, a esta cantidad se le denomina varianza, y es la más importante de las medidas de variación porque tiene la ventaja de no prescindir de los signos de las desviaciones, pero al igual que la desviación media los valores extremos pueden distorsionarla, y si formula es:

$$S^2 = \frac{\Sigma(X \quad X)^2}{n\text{-}1}$$

Ecuación 3: Calculo de la varianza muestral

.

2.11.3 DESVIACION ESTANDAR

Cuando se utiliza la varianza como medida de dispersión, para salvar el problema de trabajar con distintas dimensiones en la media y en la medida de variabilidad es necesario definir la Desviación estándar como la raíz cuadrada de la varianza.

La Desviación Estándar es útil para describir cuanto se apartan de la media de la distribución los elementos individuales. Una medida de ello se denomina puntuación estándar número de desviaciones a las que determinada observación se encuentra con respecto a la media, y su fórmula es:

$$S = \sqrt{\frac{1}{n-1}\sum_i (x_i - \bar{x})^2}$$

Ecuación 4: Calculo de la desviación estándar

2.12 MEDIDAS DE ASIMETRÍA Y CURTOSIS

2.12.1 MEDIDA DE SESGO O ASIMETRIA

En las distribuciones que no toman la forma de una curva acampanada Normal, interesa muchas veces obtener dos medias adicionales, las de asimetría y curtosis. Las medidas de asimetría muestran si en la distribución hay concentración de datos en un extremo, superior o inferior, y se denomina Sesgo positivo o a la derecha si la concentración es en el extremo inferior y Sesgo Negativo o a la izquierda si la concentración es en el superior, y u formula es:

$$g_1 = \frac{n}{(n-1)(n-2)} \sum \left(\frac{x_j - \bar{x}}{s} \right)^3$$

Ecuación 5: Calculo de la Asimetría

2.12.2 MEDIDA DE CURTOSIS

Al comparar cuán aguda es una distribución en relación con la Distribución Normal, se pueden presentar diferentes grados de apuntalamiento.

1. Mesocúrtica, Normal.

2. Plarticúrtica, Menor apuntalamiento.

3. Leptocúrtica, Mayor apuntalamiento.

Fórmula para calcular la Curtosis es:

$$g_2 = \left\{ \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum \left(\frac{x_j - \bar{x}}{s} \right)^4 \right\} - \frac{3(n-1)^2}{(n-2)(n-3)}$$

Ecuación 6: Calculo de la Curtosis

2.13 REGRESION LINEAL

Cuando se estudian dos características simultáneamente sobre una muestra, se puede considerar que una de ellas influye sobre la otra de alguna manera. El objetivo principal de la regresión es descubrir el modo en que se relacionan, y las formulas para calcular los coeficientes son:

$$b_1 = r\frac{S_Y}{S_X} \qquad\qquad b_0 = \bar{y} - b_1\bar{x}$$

Ecuación 7: Calculo de los coeficientes de regresión lineal

2.14 CORRELACIÓN LINEAL SIMPLE

Para ver si existe relación lineal entre dos variables X e Y, emplearemos un parámetro que nos mida la fuerza de asociación lineal entre ambas variables. La medida de asociación lineal más frecuentemente utilizada entre dos variables es “r” o coeficiente de correlación lineal de Pearson; este parámetro se mide en términos de covarianza de X e Y, y su fórmula es:

$$r = \frac{S_{xy}}{S_x S_y}$$

Ecuación 8: Calculo de correlación entre dos variables.

2.15 COMPARACIÓN DE MEDIAS MUESTRALES

En comparación de medias muestrales nos encontramos con varios procedimientos para contraste de medias:

1. Prueba T para una muestra.
2. Pruebas T para dos muestras independientes.
3. Prueba T para dos muestras relacionadas.

2.15.1 Prueba T para una muestra

La prueba de T para una muestra no permite contrastar hipótesis sobre la media poblacional a partir de la media obtenida en la muestra. Es necesario que la población de la que se extrae la muestra sea normal o la muestra suficiente grande.

- En contrastar variables se pasa las variables que se desean contrastar.
- En valor de prueba se escribe el valor de la media de la población.

La significación (Sig., o p-valor) indica la probabilidad de que la muestra contrasta venga de una población cuya media es el valor de prueba.

El intervalo de confianza para la diferencia entre la media de la muestra y el valor de prueba. Si el cero no está en ese intervalo no podemos aceptar que la población tenga de media el valor de prueba.

La fórmula para calcular la prueba T para una muestra es:

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}; v = n - 1$$

Donde $\mu = \mu_0$ y σ desconocida.

2.15.2 Prueba T para dos muestras independientes.

Prueba T para dos muestras independientes nos permite contrastar el que las medias de dos poblaciones independientes son iguales utilizando para ello las medias de dos muestras aleatorias extraídas de esas poblaciones.

- En contrastar variables se pasan las variables independientes que se deseen contrastar.

Hay que considerar si se consideran la varianzas de las poblaciones iguales o no ya que el estadístico es diferente en cada caso.

Se realiza el intervalo de confianza para la diferencia de medias, si el cero no está en ese intervalo no podemos considerar iguales esas medias para el nivel de confianza elegido.

Las fórmulas para calcular la prueba T para dos muestras independientes es:

Cuando las varianzas poblacionales son iguales y desconocidas.

$$t = \frac{(\bar{x}_1 - \bar{x}_2) - d_0}{s_p\sqrt{(1/n_1) + (1/n_2)}}$$

Ecuación 9: Calculo de la T de dos medias independientes con varianza poblacionales iguales.

Donde $\mu_1 - \mu_2 = d_0$

Para $\upsilon = n_1 - n_2 - 2$, $\sigma_1 = \sigma_2$ pero desconocidas

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

Ecuación 10: Calculo de la diferencia de la medias muestrales.

Cuando las varianzas poblacionales son diferentes y desconocidas.

$$t' = \frac{(\bar{x}_1 - \bar{x}_2) - d_0}{\sqrt{(s_1^2/n_1) + (s_2^2/n_2)}}$$

Ecuación 11: Calculo de la T de dos medias independientes y varianzas poblacionales desiguales.

Donde $\mu_1 - \mu_2 = d_0$

Para $\sigma_1 \neq \sigma_2$ pero desconocidas

$$v = \frac{(s_1^2/n_1 + s_2^2/n_2\ )^2}{\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1}}$$

Ecuación 12: Calculo de los grados de libertad para varianzas desiguales.

2.15.3 Prueba T para dos muestras relacionadas.

La Prueba T para dos muestras relacionadas nos permite contrastar hipótesis sobre igualdad de medias para muestras relacionadas.

Se considera una población de diferencias con media μD, resultados de restar las puntuaciones de un mismo grupo en dos variables o en la misma variable en dos momentos diferentes. De la población de diferencias se extrae una muestra aleatoria de tamaño n y se utiliza la media de está para contrastar la hipótesis de que la media de la población de diferencias es cero.

Es necesario que la población de diferencias se distribuya normalmente.

- En variables relacionadas se trasladan las parejas de variables que se desean contrastar, solo incorpora variables con formato numérico.

Nos muestra la correlación de los pares de variables y nos da el intervalo de confianza para la diferencia.

Las fórmulas para calcular la prueba T para dos muestras dependientes es:

$$t = \frac{\bar{d} - d_0}{s_d/\sqrt{n}}$$

Ecuación 13: Calculo de la T para observaciones pareadas

Donde $\mu_D = d_0$

Para $\upsilon = n-1$

2.16 INGENIERIA DE SOFTWARE

Es una disciplina de la ingeniería basada en la aplicación práctica de las ciencias de la computación y otras disciplinas, que comprende todos los aspectos de producción de software de calidad, es decir el análisis, diseño, implementación, validación, mantenimiento y evolución del software y la obtención de la documentación asociada. Puede también ser considerada como la aplicación sistemática de método, herramientas y técnicas para conseguir los objetivos fijados en cuanto a requerimientos, en un sistema de software. Debe apoyarse en estándares probados y ajustarse a un cronograma de proyecto y a un presupuesto real.

La ingeniería de software comprende adicionalmente actividades como la administración de proyectos de software y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.

2.17 DISEÑO Y CONSTRUCCION DEL SOFTWARE

Para tener éxito al diseñar y construir un software se necesita disciplina, es decir un enfoque de ingeniería. Añade que construir software de computadoras es un proceso de aprendizaje iterativo, y el resultado, algo que podría llamar capital del software, es el conjunto del software reunido, depurado y organizado mientras se desarrolla el proyecto.

Define el proceso de software como un marco de trabajo de las tareas que se requieren para construir software de calidad.

Considera que la ingeniería de software es una tecnología multicapas o estratificada, como se muestra en la gráfico 6 y que se describe a continuación.

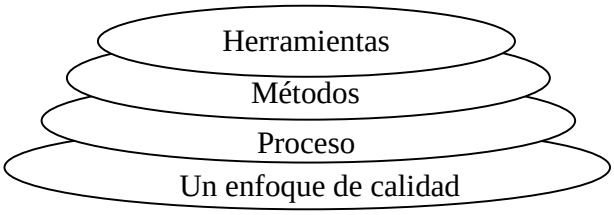


Gráfico 6: Capas de la ingeniería de software

En la base como en cualquier rama de la ingeniería, está la capa de enfoque de calidad, que debe apoyarse sobre un compromiso de organización de calidad.

Luego viene la capa de proceso, cuyas áreas claves forman la base de la gestión de proyectos y establecen el contexto en que se aplican los métodos técnicos, se obtienen productos del trabajo (modelos, documentos, datos, informes, formularios, etc.), se establecen hito, se asegura la calidad y el cambio se gestiona adecuadamente.

Luego más arriba está la capa de métodos de ingeniería de software que indican cómo construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Esto métodos dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.

Encima de las anteriores se ubica la capa de herramienta de la ingeniería del software, las cuales proporcionan un enfoque automático o semiautomático para el proceso creada por

una herramienta se la pueda utilizar en otra, se establece un sistema soporte para el desarrollo de software llamado ingeniería de software.

Para resolver los problemas reales de una industria u organización, el ingeniero o equipo de ingenieros de software debe incorporar una estrategia de desarrollo que acompañe al proceso, métodos y capa de herramientas. Esta estrategia a menudo se llama modelo de proceso, el mismo que se selecciona según la naturaleza del proyecto y la aplicación, los métodos y las herramientas a utilizarse, entre otros. Entre estos modelos se mencionan los siguientes:

El modelo lineal secuencial o modelo en cascada.

El modelo de construcción de prototipos.

El modelo de desarrollo rápido de aplicaciones DRA.

Los modelos evolutivos (incluye los modelos incrementales, espiral, concurrentes).

El modelo de desarrollo basado en componentes y tecnologías de objetos.

El modelo de métodos formales.

Como se puede apreciar, la ingeniería de software es una disciplina que integra procesos, métodos y herramientas para el desarrollo del software de computadoras.

Se han puesto varios modelos de procesos para la ingeniería de software diferente, cada uno exhibiendo ventajas e inconvenientes, pero todos tiene una serie de fases genéricas en común.

2.18 EL PROCESO DE SOFTWARE

Se determina que un proceso de software es un conjunto de actividades y resultados asociados que producen un producto de software. Aunque existen muchos procesos diferentes de software, las actividades fundamentales que son comunes para ellos son:

Especificación del software. Se debe definir las funcionalidades del software y las restricciones de sus operaciones.

Diseño e implementación del software. Corresponde al desarrollo de software, se debe producir software que cumpla especificaciones.

Validación del software. Se debe validar el software para asegurar que hace lo que el cliente desea. Incluye efectivamente la validación y verificación del software (V&V).

Evolución del software. El software debe evolucionar para cumplir los cambios en las necesidades del usuario.

2.19 PROCESOS DEL SOFTWARE

Un modelo de proceso del software es una representación abstracta de un proceso del software. Cada modelo de proceso representa un proceso desde una perspectiva particular por lo que solo provee información parcial acerca de ese proceso. Existen modelos generales que no son descripciones definidas de los procesos del software, más bien son abstracciones útiles que se pueden utilizar para explicar diferentes enfoques para desarrollar software. Entre la variedad de modelos generales o paradigmas de desarrollo de software, está:

2.19.1 Modelo en cascada

El primer modelo de desarrollo de software que se publicó, se derivó de otros procesos de ingeniería. Éste toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y las representa como fases separadas del proceso.

El modelo en cascada consta de las siguientes fases:

1. **Definición de los requisitos**: Los servicios, restricciones y objetivos son establecidos con los usuarios del sistema. Se busca hacer esta definición en detalle.
2. **Diseño de software:** Se particiona el sistema, en sistemas de software o hardware. Se establece la arquitectura total del sistema. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.
3. **Implementación y pruebas unitarias:** Construcción de los módulos y unidades de software. Se realizan pruebas de cada unidad.
4. **Integración y pruebas del sistema:** Se integran todas las unidades. Se prueban en conjunto. Se entrega el conjunto probado al cliente.
5. **Operación y mantenimiento:** Generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

La interacción entre fases puede observarse en la gráfico 7. Cada fase tiene como resultado documentos que deben ser aprobados por el usuario.

Una fase no comienza hasta que termine la fase anterior y generalmente se incluye la corrección de los problemas encontrados en fases previas.

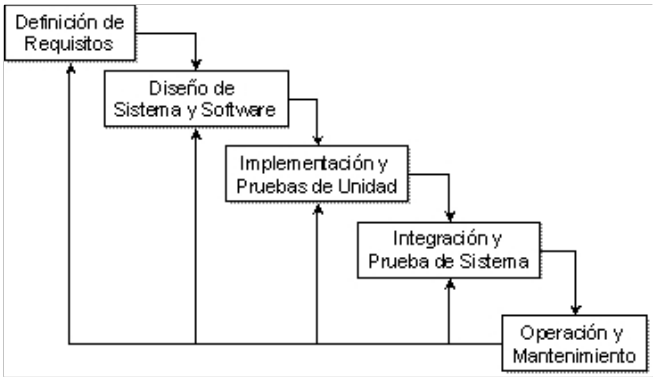


Gráfico 7: Modelo de desarrollo en cascada.

2.20 SOFTWARE

Conjunto de instrucciones y datos codificados para ser leídas e interpretadas por una computadora. Estas instrucciones y datos fueron concebidos para el procesamiento electrónico de datos.

2.21 SOFTWARE LIBRE

El software libre (en inglés free software, aunque en realidad esta denominación también puede significar gratis, y no necesariamente libre, por lo que se utiliza el hispanismo libre software también en inglés) es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente. Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software; de modo más preciso, se refiere a cuatro

libertades de los usuarios del software: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo cual se puede ayudar a otros, y de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie (para la segunda y última libertad mencionadas, el acceso al código fuente es un requisito previo).

El software libre suele estar disponible gratuitamente, o al precio de costo de la distribución a través de otros medios; sin embargo no es obligatorio que sea así, por lo tanto no hay que asociar software libre a "software gratuito" (denominado usualmente freeware), ya que, conservando su carácter de libre, puede ser distribuido comercialmente ("software comercial"). Análogamente, el "software gratis" o "gratuito" incluye en ocasiones el código fuente; no obstante, este tipo de software no es libre en el mismo sentido que el software libre, a menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

Tampoco debe confundirse software libre con "software de dominio público". Éste último es aquel software que no requiere de licencia, pues sus derechos de explotación son para toda la humanidad, porque pertenece a todos por igual. Cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original. Este software sería aquel cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado, tras un plazo contado desde la muerte de este, habitualmente 70 años. Si un autor condiciona su uso bajo una licencia, por muy débil que sea, ya no es del dominio público.

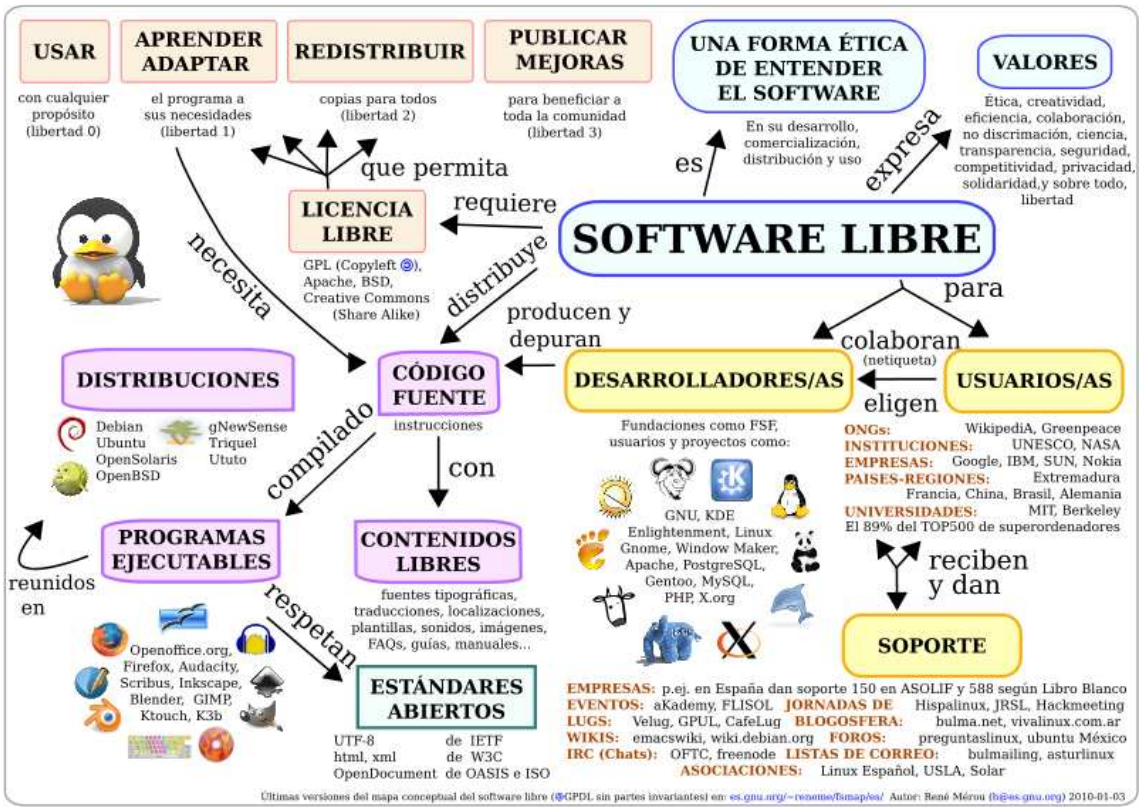


Gráfico 8: Software libre

2.22 LIBERTADES DEL SOFTWARE LIBRE

Artículo principal: Definición del Software Libre

De acuerdo con tal definición, el software es "libre" si garantiza las siguientes libertades:

Libertad	Descripción
0	La libertad de usar el programa, con cualquier propósito.
1	La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
2	La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu

	prójimo.
3	La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.
Las libertades 1 y 3 requieren acceso al código fuente porque estudiar y modificar software sin su código fuente es muy poco viable.	

Ciertos teóricos usan este cuarto punto (libertad 3) para justificar parcialmente las limitaciones impuestas por la licencia GNU GPL frente a otras licencias de software libre (ver Licencias GPL). Sin embargo el sentido original es más libre, abierto y menos restrictivo que el que le otorga la propia situación de incompatibilidad, que podría ser resuelta en la próxima versión 3.0 de la licencia GNU GPL, causa en estos momentos graves perjuicios a la comunidad de programadores de software libre, que muchas veces no pueden reutilizar o mezclar códigos de dos licencias distintas, pese a que las libertades teóricamente lo deberían permitir.

En el sitio web oficial de Open Source Initiative está la lista completa de las licencias de software libre actualmente aprobadas y tenidas como tales.

El término software no libre se emplea para referirse al software distribuido bajo una licencia de software más restrictiva que no garantiza estas cuatro libertades. Las leyes de la propiedad intelectual reservan la mayoría de los derechos de modificación, duplicación y redistribución para el dueño del copyright; el software dispuesto bajo una licencia de software libre rescinde específicamente la mayoría de estos derechos reservados.

La definición de software libre no contempla el asunto del precio; un eslogan frecuentemente usado es "libre como en libertad, no como en cerveza gratis" o en inglés "Free as in freedom, not as in free beer" (aludiendo a la ambigüedad del término inglés "free"), y es habitual ver a la venta CDs de software libre como distribuciones Linux. Sin embargo, en esta situación, el comprador del CD tiene el derecho de copiarlo y redistribuirlo. El software gratis puede incluir restricciones que no se adaptan a la definición de software libre —por ejemplo, puede no incluir el código fuente, puede prohibir explícitamente a los distribuidores recibir una compensación a cambio, etc.—.

Para evitar la confusión, algunas personas utilizan los términos "libre" (software libre) y "gratis" (software gratis) para evitar la ambigüedad de la palabra inglesa "free". Sin embargo, estos términos alternativos son usados únicamente dentro del movimiento del software libre, aunque están extendiéndose lentamente hacia el resto del mundo. Otros defienden el uso del término open source software (software de código abierto). La principal diferencia entre los términos "open source" y "free software" es que éste último tiene en cuenta los aspectos éticos y filosóficos de la libertad, mientras que el "open source" se basa únicamente en los aspectos técnicos.

En un intento por unir los mencionados términos que se refieren a conceptos semejantes, se está extendiendo el uso de la palabra "FLOSS" con el significado de free/libre and open source software e, indirectamente, también a la comunidad que lo produce y apoya.

2.23 ENTORNO DE LA PROGRAMACIÓN EN JAVA

En esta sección se presentan las características generales de Java como lenguaje de programación algorítmico. En este apartado Java es muy similar a C/C++, lenguajes en los que está inspirado. Se va a intentar ser breve, considerando que ya conoce algunos otros lenguajes de programación y está familiarizado con lo que son variables, bifurcaciones, bucles, etc.

2.24 Qué es Java

Java es un lenguaje originalmente desarrollado por un grupo de ingenieros de Sun, utilizado por Netscape posteriormente como base para Javascript. Si bien su uso se destaca en el Web, sirve para crear todo tipo de aplicaciones (locales, intranet o internet).

Java es un lenguaje:

- *de objetos*
- *independiente de la plataforma*

Algunas características notables:

- *robusto*
- *gestiona la memoria automáticamente*
- *no permite el uso de técnicas de programación inadecuadas*
- *multithreading*
- *cliente-servidor*
- *mecanismos de seguridad incorporados*
- *herramientas de documentación incorporadas*

2.25 Lenguaje de Objetos

¿Por qué puse "de" objetos y no "orientado a" objetos? Para destacar que, al contrario de otros lenguajes como C++, no es un lenguaje modificado para poder trabajar con objetos sino que es un lenguaje creado para trabajar con objetos desde cero. De hecho, TODO lo que hay en Java son objetos.

2.26 ¿Qué es un objeto?

Bueno, se puede decir que todo puede verse como un objeto. Pero seamos más claros. Un objeto, desde nuestro punto de vista, puede verse como una pieza de software que cumple con ciertas características:

- *encapsulamiento*
- *herencia*

Encapsulamiento significa que el objeto es auto-contenido, o sea que la misma definición del objeto incluye tanto los datos que éste usa (atributos) como los procedimientos (métodos) que actúan sobre los mismos.

Cuando se utiliza programación orientada a objetos, se definen clases (que definen objetos genéricos) y la forma en que los objetos interactúan entre ellos, a través de mensajes. Al crear un objeto de una clase dada, se dice que se crea una instancia de la clase, o un objeto propiamente dicho. Por ejemplo, una clase podría ser "autos", y un auto dado es una instancia de la clase.

La ventaja de esto es que como no hay programas que actúen modificando al objeto, éste se mantiene en cierto modo independiente del resto de la aplicación. Si es necesario modificar

el objeto (por ejemplo, para darle más capacidades), esto se puede hacer sin tocar el resto de la aplicación… lo que ahorra mucho tiempo de desarrollo y debugging ¡En Java, inclusive, ni siquiera existen las variables globales! (Aunque parezca difícil de aceptar, esto es una gran ventaja desde el punto de vista del desarrollo).

En cuanto a la herencia, simplemente significa que se pueden crear nuevas clases que hereden de otras preexistentes; esto simplifica la programación, porque las clases hijas incorporan automáticamente los métodos de las madres. Por ejemplo, nuestra clase "auto" podría heredar de otra más general, "vehículo", y simplemente redefinir los métodos para el caso particular de los automóviles… lo que significa que, con una buena biblioteca de clases, se puede reutilizar mucho código inclusive sin saber lo que tiene adentro.

2.27 Un ejemplo simple

Para ir teniendo una idea, vamos a poner un ejemplo de una clase Java:

```
public class Muestra extends Frame {
    // atributos de la clase
        Button si;
        Button no;
    // métodos de la clase:
    public Muestra () {
    Label comentario = new Label("Presione un botón", Label.CENTER);
    si = new Button("Sí");
```

```
        no = new Button("No");
        add("North", comentario);
        add("East", si);
        add("West", no);
        }
}
```

Esta clase no está muy completa así, pero da una idea… Es una clase heredera de la clase Frame (un tipo de ventana) que tiene un par de botones y un texto. Contiene dos atributos ("si" y "no"), que son dos objetos del tipo Button, y un único método llamado Muestra (igual que la clase, por lo que es lo que se llama un constructor).

2.28 Independiente de la plataforma

Esto es casi del todo cierto…

En realidad, Java podría hacerse correr hasta sobre una Commodore 64. La realidad es que para utilizarlo en todo su potencial, requiere un sistema operativo multithreading (como Unix, Windows95, OS/2…).

¿Cómo es esto? Porque en realidad Java es un lenguaje interpretado… al menos en principio.

Al compilar un programa Java, lo que se genera es un seudocódigo definido por Sun, para una máquina genérica. Luego, al correr sobre una máquina dada, el software de ejecución Java simplemente interpreta las instrucciones, emulando a dicha máquina genérica. Por

supuesto esto no es muy eficiente, por lo que tanto Netscape como Hotjava o Explorer, al ejecutar el código por primera vez, lo van compilando (mediante un JIT: Just In Time compiler), de modo que al crear por ejemplo la segunda instancia de un objeto el código ya esté compilado específicamente para la máquina huésped.

Además, Sun e Intel se han puesto de acuerdo para desarrollar procesadores que trabajen directamente en Java, con lo que planean hacer máquinas muy baratas que puedan conectarse a la red y ejecutar aplicaciones Java cliente-servidor a muy bajo costo.

El lenguaje de dicha máquina genérica es público, y si uno quisiera hacer un intérprete Java para una Commodore sólo tendría que implementarlo y pedirle a Sun la aprobación (para que verifique que cumple con los requisitos de Java en cuanto a cómo interpreta cada instrucción, la seguridad, etc.)

2.29 Algunas características…

Entre las características que nombramos nos referimos a la robustez. Justamente por la forma en que está diseñado, Java no permite el manejo directo del hardware ni de la memoria (inclusive no permite modificar valores de punteros, por ejemplo); de modo que se puede decir que es virtualmente imposible colgar un programa Java. El intérprete siempre tiene el control.

Inclusive el compilador es suficientemente inteligente como para no permitir un montón de cosas que podrían traer problemas, como usar variables sin inicializarlas, modificar valores de punteros directamente, acceder a métodos o variables en forma incorrecta, utilizar herencia múltiple, etc.

Además, Java implementa mecanismos de seguridad que limitan el acceso a recursos de las máquinas donde se ejecuta, especialmente en el caso de los Applets (que son aplicaciones que se cargan desde un servidor y se ejecutan en el cliente).

También está diseñado específicamente para trabajar sobre una red, de modo que incorpora objetos que permiten acceder a archivos en forma remota (via URL por ejemplo).

Además, con el JDK (Java Development Kit) vienen incorporadas muchas herramientas, entre ellas un generador automático de documentación que, con un poco de atención al poner los comentarios en las clases crea inclusive toda la documentación de las mismas en formato HTML!

Las clases en Java

En Java hay un montón de clases ya definidas y utilizables.

Éstas vienen en las bibliotecas estándar:

- java.lang - clases esenciales, números, strings, objetos, compilador, runtime, seguridad y threads (es el único paquete que se incluye automáticamente en todo programa Java)
- java.io - clases que manejan entradas y salidas
- java.util - clases útiles, como estructuras genéricas, manejo de fecha, hora y strings, número aleatorios, etc.
- java.net - clases para soportar redes: URL, TCP, UDP, IP, etc.
- java.awt - clases para manejo de interface gráfica, ventanas, etc.
- java.awt.image - clases para manejo de imágenes

- java.awt.peer - clases que conectan la interface gráfica a implementaciones dependientes de la plataforma (motif, windows)
- java.applet - clases para la creación de applets y recursos para reproducción de audio.

Para que se den una idea, los números enteros, por ejemplo, son "instancias" de una clase no redefinible, Integer, que desciende de la clase Number e implementa los siguientes atributos y métodos:

```
public final class java.lang.Integer extends java.lang.Number {
    // Atributos
        public final static int MAX_VALUE;
        public final static int MIN_VALUE;
    // Métodos Constructores
        public Integer(int value);
        public Integer(String s);
    // Más Métodos
        public double doubleValue();
        public boolean equals(Object obj);
        public float floatValue();
        public static Integer getInteger(String nm);
        public static Integer getInteger(String nm, int val);
        public static Integer getInteger(String nm, Integer val);
```

```
    public int hashCode();
    public int intValue();
    public long longValue();
    public static int parseInt(String  s);
    public static int parseInt(String  s, int  radix);
    public static String toBinaryString(int  i);
    public static String toHexString(int  i);
    public static String toOctalString(int  i);
    public String toString();
    public static String toString(int  i);
    public static String  toString(int  i, int  radix);
    public static Integer valueOf(String  s);
    public static Integer valueOf(String  s, int  radix);
}
```

Esto también nos da algunas ideas:

- La estructura de una clase
- ¡Métodos repetidos!

De la estructura enseguida hablaremos; en cuanto a los métodos repetidos (como parseInt por ejemplo), al llamarse al método el compilador decide cuál de las implementaciones del mismo usar **basándose** en la cantidad y tipo de parámetros que le pasamos. Por ejemplo, parseInt("134") y parseInt("134",16), al compilarse, generarán llamados a dos métodos distintos.

2.30 Estructura de una clase

Una clase consiste en:

algunas_palabras class ***nombre_de_la_clase*** *[algo_más] {*

[lista_de_atributos]

[lista_de_métodos]

}

Lo que está entre [y] es opcional…

Ya vemos qué poner en "algunas_palabras" y "algo_más", por ahora sigamos un poco más.

La lista de atributos (nuestras viejas variables locales) sigue el mismo formato de C: se define primero el tipo y luego el nombre del atributo, y finalmente el ";".

public final static int MAX_VALUE;

También tenemos "algunas_palabras" adelante, pero en seguida las analizaremos.

En cuanto a los métodos, también siguen la sintaxis del C; un ejemplo:

```
public int incContador() {
        cnt++;
        return(cnt);
        }
        // declaración y apertura de {
        // instrucciones, separadas por ";"
        // cierre de }
```

Finalmente, se aceptan comentarios entre /* y */, como en C, o bien usando // al principio del comentario (el comentario termina al final de la línea).

2.31 Estructura de clases

Comenzamos analizando la clase Contador, para ver las partes que forman una clase, una por una y en detalle. Resumen completo de la sintaxis.

Armamos pequeñas aplicaciones para probar cada cosa.

Recordemos la definición de la clase Contador:

```
// Implementación de un contador sencillo
        public class Contador {
        // Atributos
        int cnt;
        // Constructor
        public Contador() {
        cnt = 0;
        }
        // Métodos
        public int incCuenta() {
        cnt++;
        return cnt;
        }
        public int getCuenta() {
        return cnt;
        }
       }
```

2.32 Declaración de la clase

La clase se declara mediante la línea public class Contador. En el caso más general, la declaración de una clase puede contener los siguientes elementos:

[public] [final | abstract] class Clase [extends ClaseMadre] [implements Interfase1 [, Interfase2]…]

o bien, para interfaces:

[public] interface Interfase [extends InterfaseMadre1 [, InterfaseMadre2]…]

Como se ve, lo único obligatorio es class y el nombre de la clase. Las interfaces son un caso de clase particular que veremos más adelante.

2.33 Public, final o abstract

Definir una clase como pública (public) significa que puede ser usada por cualquier clase en cualquier paquete. Si no lo es, solamente puede ser utilizada por clases del mismo paquete (básicamente, se trata de un grupo de clases e interfaces relacionadas, como los paquetes de biblioteca incluidos con Java).

Una clase final (final) es aquella que no puede tener clases que la hereden. Esto se utiliza básicamente por razones de seguridad (para que una clase no pueda ser reemplazada por otra que la herede), o por diseño de la aplicación.

Una clase abstracta (abstract) es una clase que puede tener herederas, pero no puede ser instanciada. Es, literalmente, abstracta (como la clase Number definida en java.lang). ¿Para qué sirve? Para modelar conceptos. Por ejemplo, la clase Number es una clase abstracta que representa cualquier tipo de números (y sus métodos no están implementados: son abstractos); las clases descendientes de ésta, como Integer o Float, sí implementan los métodos de la madre Number, y se pueden instanciar.

Por lo dicho, una clase no puede ser final y abstract a la vez (ya que la clase abstract requiere descendientes…)

2.34 Extends

La instrucción extends indica de qué clase desciende la nuestra. Si se omite, Java asume que desciende de la superclase Object.

Cuando una clase desciende de otra, esto significa que hereda sus atributos y sus métodos (es decir que, al menos que los redefinamos, sus métodos son los mismos que los de la clase madre y pueden utilizarse en forma transparente, a menos que sean privados en la clase madre o, para subclases de otros paquetes, protegidos o propios del paquete).

2.35 Implements

Una interfase (interface) es una clase que declara sus métodos pero no los implementa; cuando una clase implementa (implements) una o más interfases, debe contener la implementación de todos los métodos (con las mismas listas de parámetros) de dichas interfases.

Esto sirve para dar un ascendiente común a varias clases, obligándolas a implementar los mismos métodos y, por lo tanto, a comportarse de forma similar en cuanto a su interfase con otras clases y subclases.

2.36 Interface

Una interfase (interface), como se dijo, es una clase que no implementa sus métodos sino que deja a cargo la implementación a otras clases. Las interfases pueden, asimismo, descender de otras interfases pero no de otras clases.

Todos sus métodos son por definición abstractos y sus atributos son finales (aunque esto no se indica en el cuerpo de la interfase).

Son útiles para generar relaciones entre clases que de otro modo no están relacionadas (haciendo que implementen los mismos métodos), o para distribuir paquetes de clases indicando la estructura de la interfase pero no las clases individuales (objetos anónimos).

Si bien diferentes clases pueden implementar las mismas interfases, y a la vez descender de otras clases, esto no es en realidad herencia múltiple ya que una clase no puede heredar atributos ni métodos de una interface; y las clases que implementan una interfase pueden no estar ni siquiera relacionadas entre sí.

2.37 El cuerpo de la clase

El cuerpo de la clase, encerrado entre { y }, es la lista de atributos (variables) y métodos (funciones) que constituyen la clase.

No es obligatorio, pero en general se listan primero los atributos y luego los métodos.

2.38 Declaración de atributos

En Java no hay variables globales; todas las variables se declaran dentro del cuerpo de la clase o dentro de un método. Las variables declaradas dentro de un método son locales al método; las variables declaradas en el cuerpo de la clase se dice que son miembros de la clase y son accesibles por todos los métodos de la clase.

Por otra parte, además de los atributos de la propia clase se puede acceder a todos los atributos de la clase de la que desciende; por ejemplo, cualquier clase que descienda de la clase Polygon hereda los atributos npoints, xpoints e ypoints.

Finalmente, los atributos miembros de la clase pueden ser atributos de clase o atributos de instancia; se dice que son atributos de clase si se usa la palabra clave static: en ese caso la variable es única para todas las instancias (objetos) de la clase (ocupa un único lugar en memoria). Si no se usa static, el sistema crea un lugar nuevo para esa variable con cada instancia (o sea que es independiente para cada objeto).

La declaración sigue siempre el mismo esquema:

[private | protected | public] [static] [final] [transient] [volatile] Tipo NombreVariable [= Valor];

2.39 Private, protected o public

Java tiene 4 tipos de acceso diferente a las variables o métodos de una clase: privado, protegido, público o por paquete (si no se especifica nada).

De acuerdo a la forma en que se especifica un atributo, objetos de otras clases tienen distintas posibilidades de accederlos:

Acceso desde:	private	protected	public	(package)
la propia clase	S	S	S	S
subclase en el mismo paquete	N	S	S	S
otras clases en el mismo paquete	N	S	S	S
subclases en otros paquetes	N	X	S	N
otras clases en otros paquetes	N	N	S	N

S: puede acceder

N: no puede acceder

X: puede acceder al atributo en objetos que pertenezcan a la subclase, pero no en los que pertenecen a la clase madre. Es un caso especial ; más adelante veremos ejemplos de todo esto.

2.40 Static y final

Como ya se vio, **static** sirve para definir un atributo como de clase, o sea único para todos los objetos de la clase.

En cuanto a **final**, como en las clases, determina que un atributo no pueda ser sobreescrito o redefinido. Es decir: no se trata de una variable, sino de una constante.

2.41 Transient y volatile

Son casos bastante particulares y que no habían sido implementados en Java 1.0.

Transient denomina atributos que no se graban cuando se archiva un objeto, o sea que no forman parte del estado permanente del mismo.

Volatile se utiliza con variables modificadas asincrónicamente por objetos en diferentes **threads** (literalmente "hilos", tareas que se ejecutan en paralelo); básicamente esto implica que distintas tareas pueden intentar modificar la variable simultáneamente, y volatile asegura que se vuelva a leer la variable (por si fue modificada) cada vez que se la va a usar (esto es, en lugar de usar registros de almacenamiento como buffer).

2.42 MÉTODOS

Los métodos son funciones definidas dentro de una clase. Salvo los métodos static o de clase, se aplican siempre a un objeto de la clase por medio del operador punto (.). Dicho objeto es su argumento implícito. Los métodos pueden además tener otros argumentos explícitos que van entre paréntesis.

2.43 VARIABLES

Una variable es un nombre que contiene un valor que puede cambiar a lo largo del programa. De acuerdo con el tipo de información que contienen, en Java hay dos tipos principales de variables:

Variables de tipos primitivos. Están definidas mediante un valor único que puede ser entero, de punto flotante, carácter o booleano. Java permite distinta precisión y distintos rangos de valores para estos tipos de variables (char, byte, short, int, long, float, double, boolean). Ejemplos de variables de tipos primitivos podrían ser: 123, 3456754, 3.1415, 12e-09, 'A', True, etc.

Variables referencia. Las variables referencia son referencias o nombres de una información más compleja: arrays u objetos de una determinada clase.

2.44 CONSTANTE

Las constantes son datos cuyo valor no puede variar durante la ejecución de un programa.

En un programa pueden aparecer constantes de dos tipos: literales y simbólicas.

2.45 OPERADORES

Java es un lenguaje rico en operadores. Estos operadores se describen brevemente en los apartados siguientes.

2.45.1 Operadores aritméticos

Son operadores binarios (requieren siempre dos operandos) que realizan las operaciones aritméticas habituales: suma (+), resta (-), multiplicación (*), división (/) y resto de la división (%).

2.45.2 Operadores de asignación

Los operadores de asignación permiten asignar un valor a una variable. El operador de asignación por excelencia es el operador igual (=). La forma general de las sentencias de asignación con este operador es:

Operador	Utilización	Expresión equivalente
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2

variable = expresión;

Java dispone de otros operadores de Tabla precedente Otros operadores de asignación. Se trata de versiones abreviadas del operador (=) que realizan operaciones “acumulativas” sobre una variable. La tabla precedente muestra estos operadores y su equivalencia con el uso del operador igual (=).

2.45.3 Operadores unarios

Los operadores más (+) y menos (-) unarios sirven para mantener o cambiar el signo de una variable, constante o expresión numérica. Su uso en Java es el estándar de estos operadores.

2.45.4 Operador condicional?:

Este operador, permite realizar bifurcaciones condicionales sencillas. Su forma general es la siguiente:

booleanExpression ? res1 : res2

donde se evalúa booleanExpression y se devuelve res1 si el resultado es true y res2 si el resultado es false. Es el único operador ternario (tres argumentos) de Java. Como todo operador que devuelve un valor puede ser utilizado en una expresión. Por ejemplo las sentencias:

x=1; y=10; z = (x<y)?x+3:y+8;

asignarían a z el valor 4, es decir x+3.

Operadores incrementales

Java dispone del operador incremento (++) y decremento (--). El operador (++) incrementa en una unidad la variable a la que se aplica, mientras que (--) la reduce en una unidad. Estos operadores se pueden utilizar de dos formas:

- Precediendo a la variable (por ejemplo: ++i). En este caso primero se incrementa la variable y luego se utiliza (ya incrementada) en la expresión en la que aparece.
- Siguiendo a la variable (por ejemplo: i++). En este caso primero se utiliza la variable en la expresión (con el valor anterior) y luego se incrementa.

En muchas ocasiones estos operadores se utilizan para incrementar una variable fuera de una expresión. En este caso ambos operadores son equivalentes. Si se utilizan en una expresión más complicada, el resultado de utilizar estos operadores en una u otra de sus formas será diferente. La actualización de contadores en bucles for es una de las aplicaciones más frecuentes de estos operadores.

2.45.5 Operadores relacionales

Los operadores relacionales sirven para realizar comparaciones de igualdad, desigualdad y relación de menor o mayor. El resultado de estos operadores es siempre un valor boolean (true o false) según se cumpla o no la relación considerada.

Operador	Utilización	El resultado es true
>	op1 > op2	si op1 es mayor que op2
>=	op1 >= op2	si op1 es mayor o igual que op2
<	op1 < op2	si op1 es menor que op2
<=	op1 <= op2	si op1 es menor o igual que op2
==	op1 == op2	si op1 y op2 son iguales
!=	op1 != op2	si op1 y op2 son diferentes

2.45.6 Operadores lógicos

Los operadores lógicos se utilizan para construir expresiones lógicas, combinando valores lógicos (true y/o false) o los resultados de los operadores relacionales. La siguiente tabla muestra los operadores lógicos de Java. Debe notarse que en ciertos casos el segundo operando no se evalúa porque ya no es necesario (si ambos tienen que ser true y el primero es false, ya se sabe que la condición de que ambos sean true no se va a cumplir). Esto puede traer resultados no deseados y por eso se han añadido los operadores (&) y (|) que garantizan que los dos operandos se evalúan siempre.

Operador	Nombre	Utilización	Resultado
&&	AND	op1 && op2	si op1 y op2 son true. Si op1 es false ya no se evalúa op2
||	OR	op1 || op2	true si op1 u op2 son true. Si op1 es true ya no se evalúa op2
!	negación	! op	true si op es false y false si op es true
&	AND	op1 & op2	true si op1 y op2 son true. Siempre se evalúa op2
|	OR	op1 | op2	true si op1 u op2 son true. Siempre se evalúa op2

2.46 ESTRUCTURA DE PROGRAMACIÓN

Las estructuras de programación o estructuras de control permiten tomar decisiones y realizar un proceso repetidas veces. Son los denominados bifurcaciones y bucles. En la mayoría de los lenguajes de programación, este tipo de estructuras son comunes en cuanto a concepto, aunque su sintaxis varía de un lenguaje a otro.

2.46.1 Sentencias o expresiones

Una expresión es un conjunto variables unidos por operadores. Son órdenes que se le dan al computador para que realice una tarea determinada.

Una sentencia es una expresión que acaba en punto y coma (;). Se permite incluir varias sentencias en una línea, aunque lo habitual es utilizar una línea para cada sentencia. Por ejemplo:

i=0; j=5; x=i+ j; // Línea compuesta de tres sentencias

2.46.2 Comentarios

Java interpreta que todo lo que aparece a la derecha de dos barras "//" en una línea cualquiera del código es un comentario del programador y no lo tiene en cuenta. El comentario puede empezar al comienzo de la línea o a continuación de una instrucción que debe ser ejecutada. La segunda forma de incluir comentarios consiste en escribir el texto entre los símbolos /*…*/. Este segundo método es válido para comentar más de una línea de código. Por ejemplo:

// Esta línea es un comentario int a=1; // Comentario a la derecha de una sentencia // Esta es la forma de comentar más de una línea utilizando // las dos barras. Requiere incluir dos barras al comienzo de cada línea / Esta segunda forma es mucho más cómoda para comentar un número elevado de líneas.*

2.47 ESTRUCTURAS DE CONTROL

2.47.1 Bifurcaciones

Las bifurcaciones permiten ejecutar una de entre varias acciones en función del valor de una expresión lógica o relacional. Se tratan de estructuras muy importantes ya que son las encargadas de controlar el flujo de ejecución de un programa. Existen dos bifurcaciones diferentes: if y switch.

2.47.2 Bifurcación if

Esta estructura permite ejecutar un conjunto de sentencias en función del valor que tenga la expresión de comparación (se ejecuta si la expresión de comparación tiene valor true). Tiene la forma siguiente:

if (booleanExpression) {

statements; }

Las llaves {} sirven para agrupar en un bloque las sentencias que se han de ejecutar, y no son necesarias si sólo hay una sentencia dentro del if.

2.47.3 Bifurcación if else

Análoga a la anterior, de la cual es una ampliación. Las sentencias incluidas en el else se ejecutan en el caso de no cumplirse la expresión de comparación (false),

if (booleanExpression) {

statements1; } else {

statements2; }

2.47.4 Bifurcación if elseif else

Permite introducir más de una expresión de comparación. Si la primera condición no se cumple, se compara la segunda y así sucesivamente. En el caso de que no se cumpla ninguna de las comparaciones se ejecutan las sentencias correspondientes al else.

```
if (booleanExpression1) {
        statements1;
} else if
(booleanExpression2) {
        statements2;
 } else if
(booleanExpression3) {
        statements3;
} else { statements4;
}
```

2.47.5 Sentencia switch

Se trata de una alternativa a la bifurcación if elseif else cuando se compara la misma expresión con distintos valores. Su forma general es la siguiente:

```
switch (expression) {
case value1:  statements1; break;
case value2: statements2; break;
case value3: statements3; break;
case value4: statements4; break;
case value5: statements5; break;
case value6: statements6; break;
[default: statements7;]
}
```

2.47.6 Bucles

Un bucle se utiliza para realizar un proceso repetidas veces. Se denomina también lazo o loop. El código incluido entre las llaves {} (opcionales si el proceso repetitivo consta de una sola línea), se ejecutará mientras se cumpla unas determinadas condiciones. Hay que prestar especial atención a los bucles infinitos, hecho que ocurre cuando la condición de finalizar el bucle (booleanExpression) no se llega a cumplir nunca. Se trata de un fallo muy típico, habitual sobre todo entre programadores poco experimentados.

2.47.7 Bucle while

Las sentencias statements se ejecutan mientras booleanExpression sea true.

while (booleanExpression) {
statements; }

2.47.8 Bucle for

La forma general del bucle for es la siguiente:

for (initialization; booleanExpression; increment) {
statements; }

La sentencia o sentencias initialization se ejecuta al comienzo del for, e increment después de statements. La booleanExpression se evalúa al comienzo de cada iteración; el bucle termina cuando la expresión de comparación toma el valor false. Cualquiera de las tres partes puede estar vacía. La

initialization y el increment pueden tener varias expresiones separadas por comas.

2.47.9 Bucle do while

Es similar al bucle while pero con la particularidad de que el control está al final del bucle (lo que hace que el bucle se ejecute al menos una vez, independientemente de que la condición se cumpla o no). Una vez ejecutados los statements, se evalúa la condición: si resulta true se vuelven a ejecutar las sentencias incluidas en el bucle, mientras que si la condición se evalúa a false finaliza el bucle. Este tipo de bucles se utiliza con frecuencia para controlar la satisfacción de una determinada condición de error o de convergencia.

do {
statements }
while (booleanExpression);

2.47.10 Sentencias break y continue

La sentencia break es válida tanto para las bifurcaciones como para los bucles. Hace que se salga inmediatamente del bucle o bloque que se está ejecutando, sin realizar la ejecución del resto de las sentencias.

La sentencia continue se utiliza en los bucles (no en bifurcaciones). Finaliza la iteración “i” que en ese momento se está ejecutando (no ejecuta el resto de sentencias que hubiera hasta el final del bucle). Vuelve al comienzo del bucle y comienza la siguiente iteración (i+1).

2.47.11 Sentencia return

Otra forma de salir de un bucle (y de un método) es utilizar la sentencia return. A diferencia de continue o break, la sentencia return sale también del método o función. En el caso de que la función devuelva alguna variable, este valor se deberá poner a continuación del *return (return value;).*

2.47.12 Bloque try {...} catch {...} finally {...}

Java incorpora en el propio lenguaje la gestión de errores. El mejor momento para detectar los errores es durante la compilación. Sin embargo prácticamente sólo los errores de sintaxis son detectados en esta operación. El resto de problemas surgen durante la ejecución de los programas.

En el lenguaje Java, una Exception es un cierto tipo de error o una condición anormal que se ha producido durante la ejecución de un programa. Algunas excepciones son fatales y provocan que se deba finalizar la ejecución del programa. En este caso conviene terminar ordenadamente y dar un mensaje explicando el tipo de error que se ha producido. Otras excepciones, como por ejemplo no encontrar un fichero en el que hay que leer o escribir algo, pueden ser recuperables.

2.47.13 Concepto de Interface

Una interface es un conjunto de declaraciones de funciones. Si una clase implementa (implements) una interface, debe definir todas las funciones especificadas por la interface. Las interfaces pueden definir también variables finales (constantes). Una clase puede implementar más de una interface, representando una alternativa a la herencia múltiple.

En algunos aspectos los nombres de las interfaces pueden utilizarse en lugar de las clases. Por ejemplo, las interfaces sirven para definir referencias a cualquier objeto de cualquiera de las clases que implementan esa interface. Con ese nombre o referencia, sin embargo, sólo se pueden utilizarlos métodos de la interface. Éste es un aspecto importante del polimorfismo.

Una interface puede derivar de otra o incluso de varias interfaces, en cuyo caso incorpora las declaraciones de todos los métodos de las interfaces de las que deriva (a diferencia de las clases, las interfaces de Java sí tienen herencia múltiple).

2.48 CLASES DE UTILIDAD

Programando en Java nunca se parte de cero: siempre se parte de la infraestructura definida.

2.48.1 ARRAYS

Los arrays de Java (vectores, matrices, hiper-matrices de más de dos dimensiones) se tratan como objetos de una clase predefinida. Los arrays son objetos, pero con algunas características propias. Los arrays pueden ser asignados a objetos de la clase Object y los métodos de Object pueden ser utilizados con arrays.

Un array se puede crear de la siguiente forma

double[] x = new double[100];

2.48.2 ARRAYS BIDIMENSIONALES

Los arrays bidimensionales de Java se crean con reserva dinámica de memoria. En Java una matriz es un vector de vectores fila, o más en concreto un vector de referencias a los vectores fila. Con este esquema, cada fila podría tener un número de elementos diferente.

Una matriz se puede crear directamente en la forma,

int [][] mat = new int[3][4];

2.49 CLASE Math

La clase java.lang.Math proporciona una serie de constantes y funciones de uso muy común en expresiones aritméticas.

Math es una clase en el paquete fundamental java.lang. Los métodos estáticos de la clase Math realizan cálculos matemáticos básicos tales como máximo, mínimo, valor absoluto y operaciones numéricas que incluyen funciones exponenciales, logarítmicas, raíz cuadrada y trigonométrica.

Los operadores aritméticos permiten realizar operaciones aritméticas básicas, actúan sobre operandos numéricos y devuelven un resultado de tipo numérico.

2.50 CLASES EN NetBeans IDE 6.5.

2.50.1 JTable

Un JTable es un componente visual de Java que nos permite dibujar una tabla, de forma que en cada fila/columna de la tabla podamos poner el dato que queramos; un nombre, un apellido, una edad, un número, etc.

2.50.2 JButton

Esta clase implementa la forma más habitual de botón gráfico de interacción que sirve para ejecutar una acción haciendo clic sobre él. También se puede activar mediante el teclado si se asociado una combinación de teclas. Puede tener un texto y/o icono.

2.50.3 JToolBar

Esta clase implementa una barra de herramientas, formada normalmente por botones o controles que incluyen iconos, y que aparecen organizados como una fila o una columna dependiendo de la zona de la pantalla donde se coloque.

2.50.4 JPanel

JPanel es un contenedor simple de propósito general que sirve para agrupar a otros componentes. Habitualmente se utiliza para agrupar componentes a los que se aplica un gestor de disposición adecuado.

2.50.5 JScrollPane

La clase JScrollPane proporciona un panel con la capacidad de presentar barras desplazamiento para mostrar su contenido. Es adecuado para presentar información que no cabe completamente en la zona de visualización asignada.

2.50.6 JFileChooser

Se trata de un selector de archivos que permite la elección interactiva de un archivo o un directorio.

2.50.7 JFrame

JFrame se emplea para crear la ventana principal de una aplicación. Es una ventana con marco que incluye los controles habituales de cambio de tamaño y cierre (por ejemplo, cerrar, iconizar, maximizar).

2.50.8 JDialog

La clase JDialog es la clase raíz de las ventanas secundarias que implementa cuadros de dialogo en Swing. Estas ventanas dependen de una ventana principal (o con marco, normalmente de clase JFrame) y si la ventana principal se cierra, se iconiza o se desiconiza, las ventanas secundarias realizan la misma operación de forma automática.

2.50.9 JTextField

Componente que permite mostrar una única línea de texto.

2.50.10JList

La clase JList implementa una lista de elementos que se presenta habitualmente en forma de columna. En esta lista el usuario puede realizar la selección de uno (comportamiento por defecto) o varios elementos.

2.50.11 JCheckBox

Es una casilla de verificación de dos estados posibles: seleccionada o no seleccionada, que determina y modifica su apariencia gráfica (normalmente mediante una cruz o marca de selección). Generalmente se utiliza para permitir que el usuario decida si desea elegir una opción o no. Si hay varias casillas de verificación, éstas no son mutuamente excluyentes, de modo que varias de ellas pueden estar seleccionadas de forma simultánea.

2.50.12 JLabel

Esta clase implementa una etiqueta que puede contener una cadena de texto, un icono o ambos. En una etiqueta se puede especificar donde aparece su contenido indicando el alineamiento vertical y horizontal.

CAPITULO III

3 INTERFAZ GRÁFICA EN JAVA

3.1 INTRODUCCION

En este capítulo hablaremos del desarrollo del software libre estadístico denominado SoftEstad versión 1.0, utilizando el modelo en cascada, que se adapta a las condiciones en el desarrollado para realizar estadística descriptiva y su forma de programación en un software desarrollador libre como Java.

Para realizar el software en Java se utilizo componentes gráficos:

3.1.1 Ventana principal.

Esta es la vista de la pantalla principal donde se comenzó a elaborar el software.

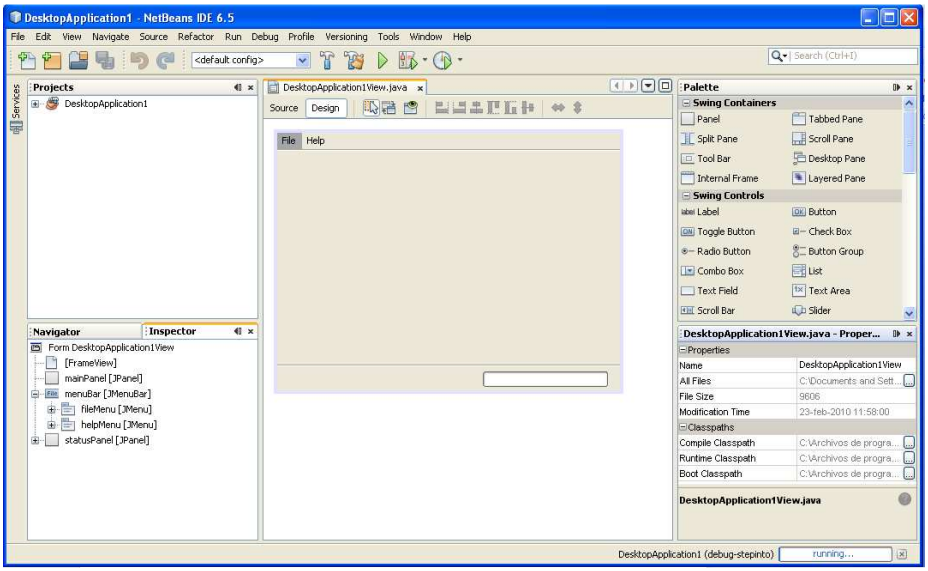


Gráfico 9: Pantalla Principal de NetBeans IDE 6.5

3.1.2 JTable

Se utilizo el componente JTable para visualizar los datos que vamos a analizar.

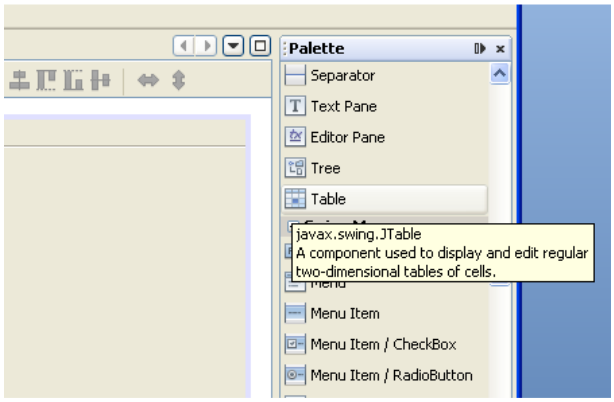


Gráfico 10: Componente JTable

3.1.3 JButton

El componente JButton se utilizo en varias ventanas el cual se utiliza para realizar acciones como acceder a otra ventana o realizar los cálculos necesarios.

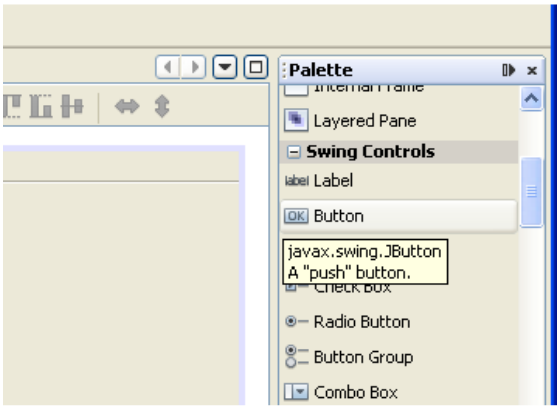


Gráfico 11: Componente JButton

3.1.4 JToolBar

Esta clase implemento para añadir una barra de menús que no permitió poner en una forma ordenada, las diferentes acciones que realiza el software.

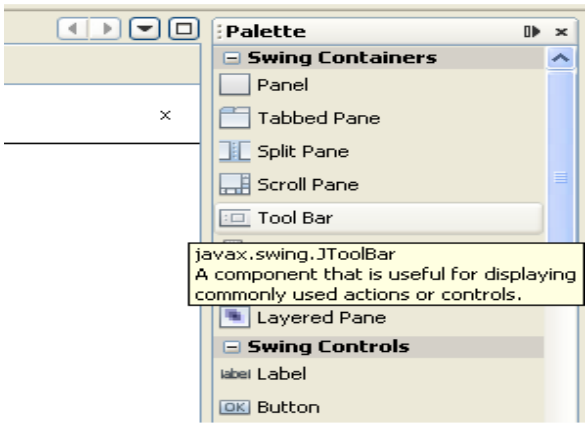


Gráfico 12: Componente JTooBar

3.1.5 JPanel

JPanel nos sirvió para la agrupación varios componentes en forma ordenada.

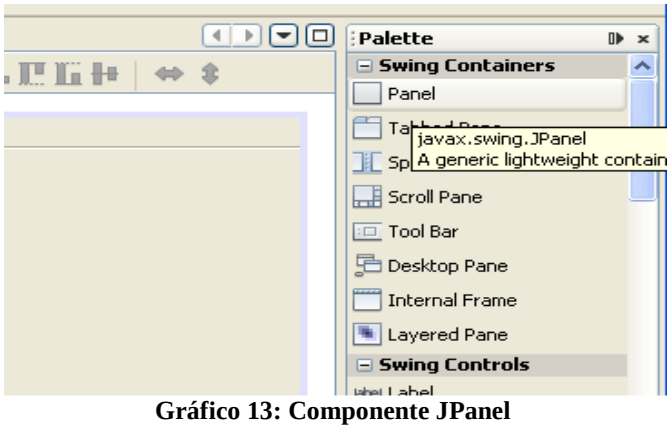


Gráfico 13: Componente JPanel

3.1.6 JScrollPane

La clase JScrollPane se utiliza cuando se desea ver la información que esta fuera del alcance visual de la pantalla.

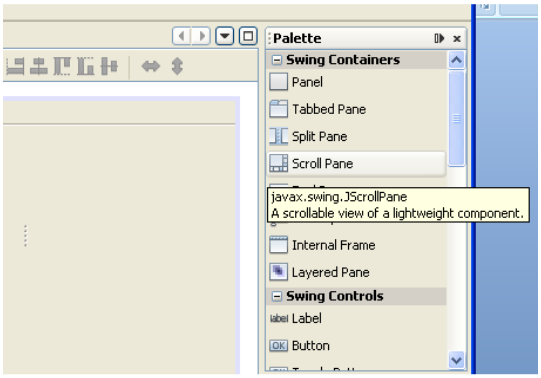


Gráfico 14: Componente JScrollPane

3.1.7 JFileChooser

El componente JFileChooser utilizamos para abrir archivo de datos de tipo texto delimitado y de Excel 97-2003.

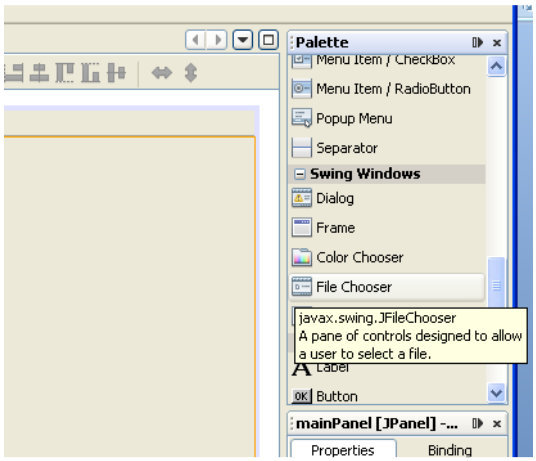


Gráfico 15: Componente JFileChooser

3.1.8 JFrame

Son ventanas donde se añade todo los componentes visuales para realizar el análisis de las variables.

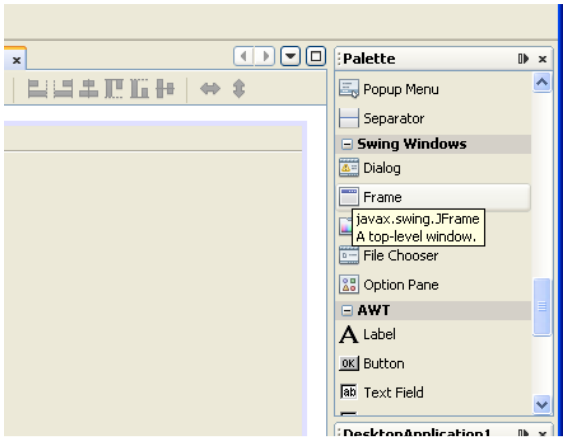


Gráfico 16: Componente JFrame

3.1.9 JDialog

La clase JDialog se utilizo para los visualizar los resultados de las variable que se mando a analizar.

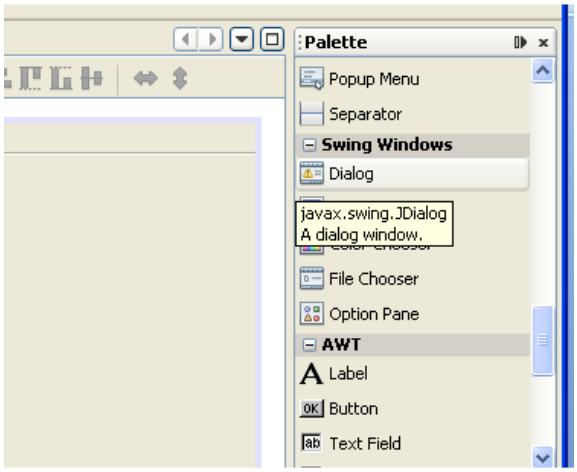


Gráfico 17: Componente JDialog

3.1.10 JTextField

Componente que permitió visualizar la variable seleccionada para el análisis de datos que se necesita al momento de realizar un análisis en particular.

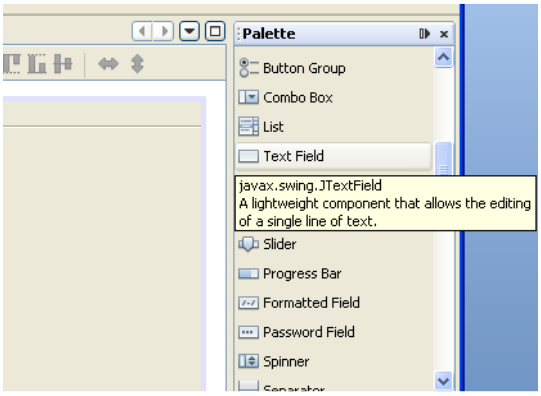


Gráfico 18: Componente JTextField

3.1.11 JList

La clase JList nos ayudo a visualizar lo nombres de las variables para su posterior análisis.

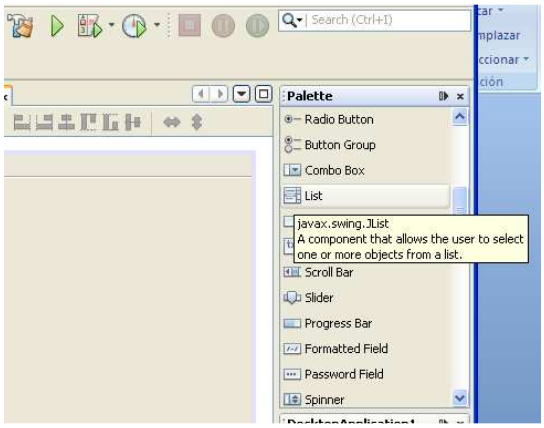


Gráfico 19: Componente JList

3.1.12 JCheckBox

El JCheckBox se utilizo en la ventana de estadística descriptiva permitiéndonos seleccionar las diferentes opciones que había para el análisis de la variable.

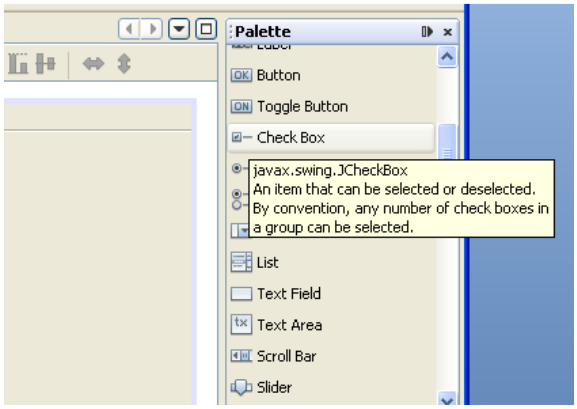


Gráfico 20: Componente JCheckBox

3.1.13 JLabel

JLabel nos permitió etiquetar las diferentes actividades que realizaba en la ventana.

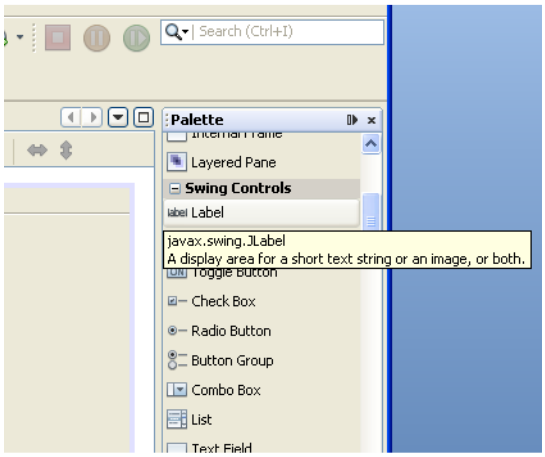


Gráfico 21: Componente JLabel

IMPLEMENTACIÓN DEL SOFWARE

A continuación se procede a implementar o construir el sistema de software SoftEstad versión 1.0, para el efecto se consideran los siguientes pasos:

- Preparación del entorno de generación y construcción
- Generación de formularios y código de los componentes del software.
- Ejecución de las pruebas del sistema.

3.2 Preparación del entorno de generación y construcción

La implementación el software SoftEstad versión 1.0, se basa en la utilización del IDE de NetBeans IDE 6.5.

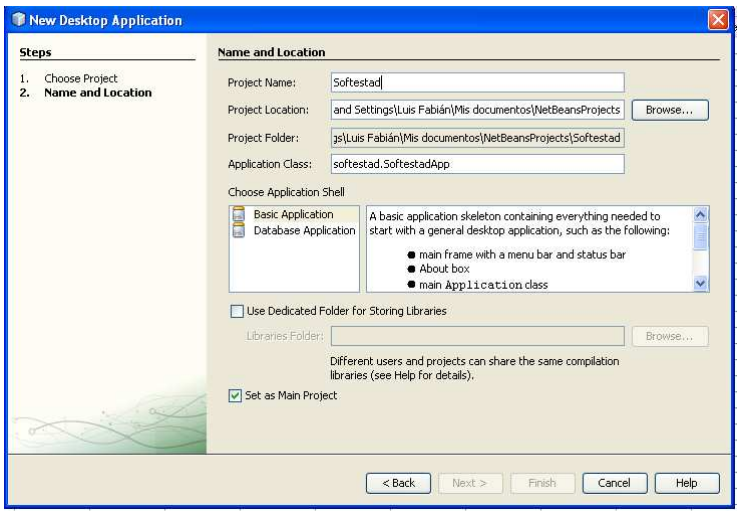


Gráfico 22: Vista para iniciar un proyecto en NetBeans IDE 6.5

Para la implementación de la interface se utiliza un conjunto de formularios diseñados mediante la utilización de la paleta de componentes, que permite establecer la pantalla de

presentación, la pantalla de menú principal que incluye opciones para seleccionar el tipo de estadística desea realizar.

Otros formularios que se definen son los de las pantallas de presentación para realizar la selección de la variable que necesitamos analizar.

En las respectivas unidades de cada formulario se tiene el código necesario para el procedimiento de la información pertinente en cada situación, lo cual hace posible la presentación de tablas, gráficos.

A continuación se presenta una muestra en secuencia, de lo arriba señalado.

3.2.1 Vista parcial de la pantalla de presentación del software

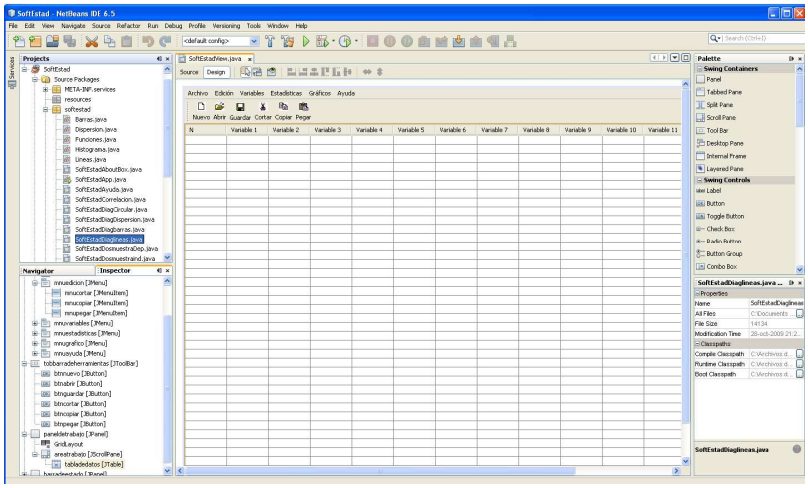


Gráfico 23: Vista parcial de la pantalla de presentación del software.

Más vistas de los formularios y códigos que son parte de SoftEstad versión 1.0 se presenta a continuación.

3.2.2 Vista del diseño del formulario para Estadística Descriptiva.

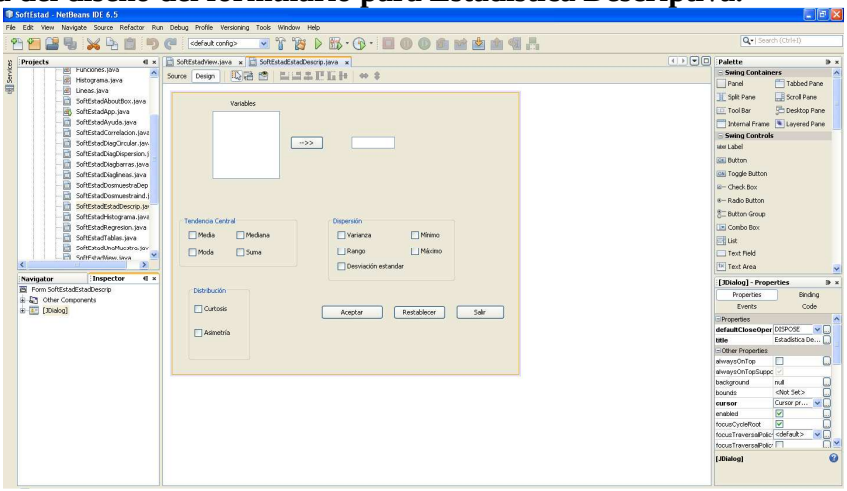


Gráfico 24: Vista del diseño del formulario para Estadística Descriptiva.

3.2.3 Vista parcial de SoftEstadEstadDescrip.java en el editor de código

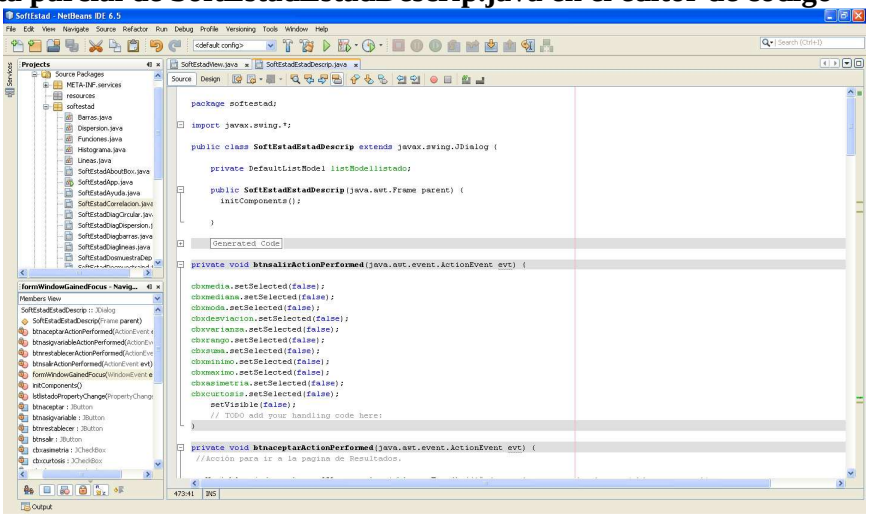


Gráfico 25: Vista parcial de SoftEstadEstadDescrip.java en el editor de código.

3.2.4 Vista parcial del formulario de comparación con una media

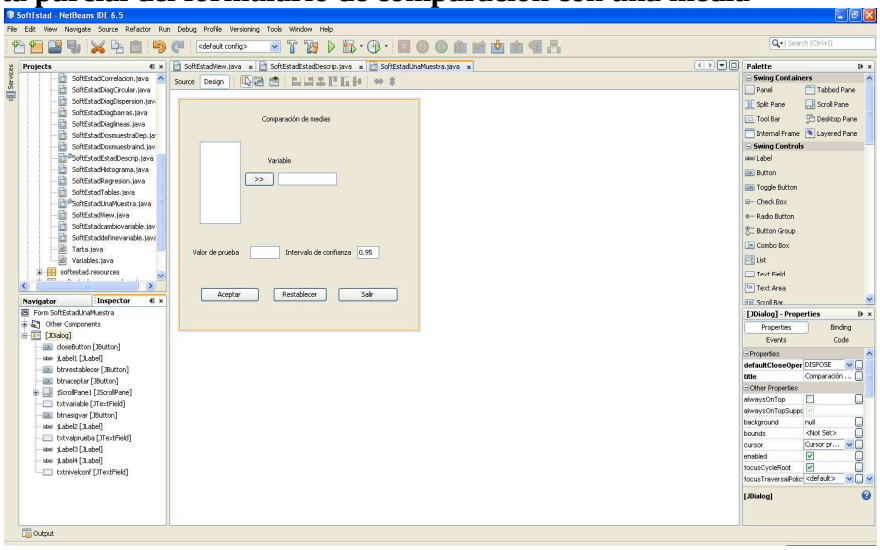


Gráfico 26: Vista parcial del formulario de comparación con una media.

3.2.5 Vista parcial del código de barra.Java en el diagrama de barras.

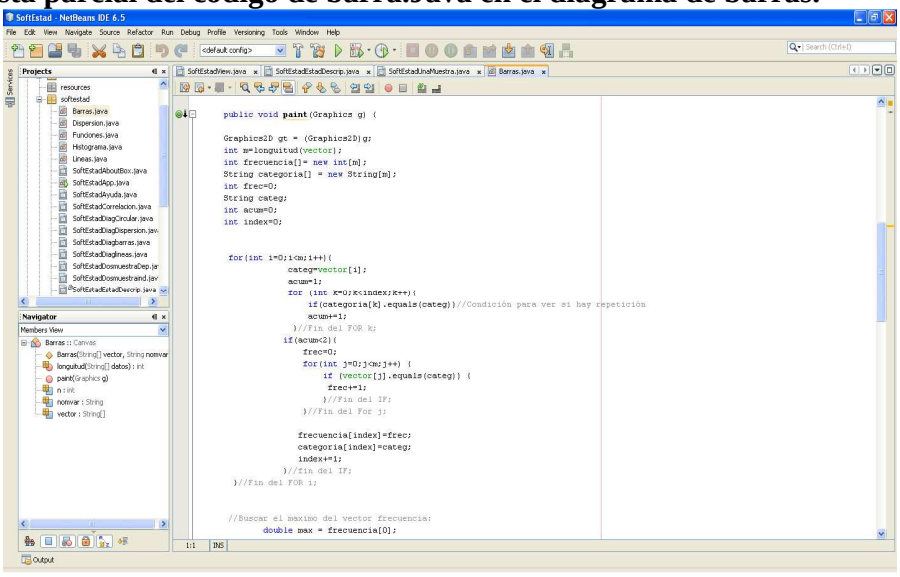


Gráfico 27: Vista parcial del código de barra.Java en el diagrama de barras.

3.2.6 Vista de la Clase Funciones.java para calcular estadísticas.

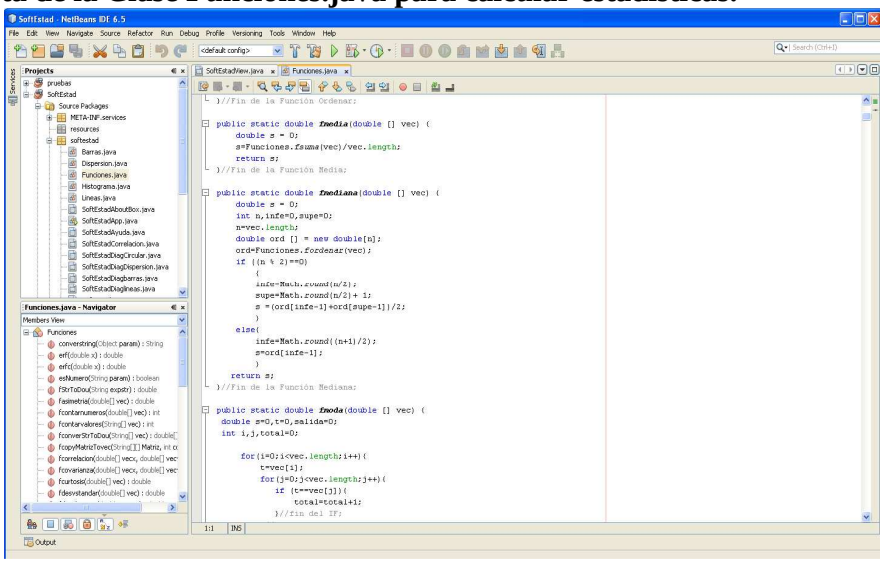


Gráfico 28: Vista de la Clase Funciones.java para calcular estadísticas.

3.3 Ejecución de la pruebas del software.

Las pruebas del software se efectuaron de manera progresiva, iterativa y paralela a la construcción de los formularios y código que componen SoftEstad versión 1.0, hasta obtener el funcionamiento correcto y esperado del mismo, tanto por unidades como del sistema en conjunto.

3.4 Acerca de la elaboración de manual de usuario

Debe elaborarse el respectivo manual de usuario y adjuntarse en la documentación complementaria del proyecto desarrollado, Igualmente puede definirse un plan para la formación de los usuarios, en este caso profesores y estudiantes

3.5 Acerca de la evolución del software.

A los largo del tiempo ha existido entre el proceso de desarrollo y el proceso de evolución (o mantenimiento) del software tiende en la actualidad a ser mas irrelevante. Continúa explicando que hoy en día pocos software son completamente nuevos, lo que implica que tiene más sentido ver el desarrollo y el mantenimiento como actividades continúas. Más que dos procesos separados, es más realista considerar a la ingeniería de software como un proceso evolutivo en la cual el software se cambia continuamente durante su periodo de vida como respuesta a los requerimientos cambiantes y necesidades del usuario.

3.6 Ejecución del software obtenido

3.6.1 Venta principal del software SofEstad V1.0.

La aplicación del software SoftEstad versión 1.0, obtenida, presenta el siguiente entorno durante su ejecución.

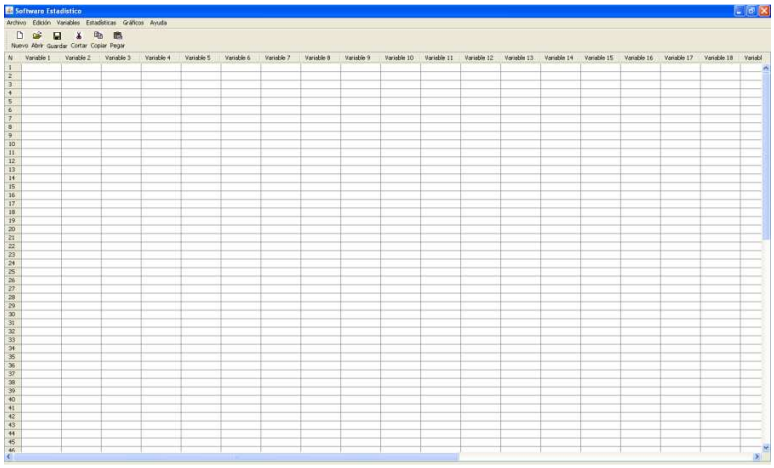


Gráfico 29: Ventana principal del software SoftEstad.

3.6.2 Ventana para abrir un archivo de texto delimitado.

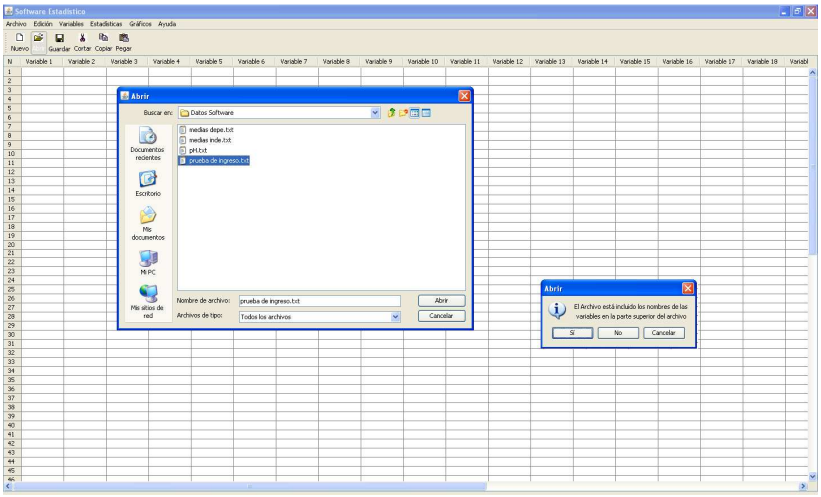


Gráfico 30: Ventana para abrir un archivo texto delimitado.

3.6.3 Vista de la ventana para realizar estadística descriptiva.

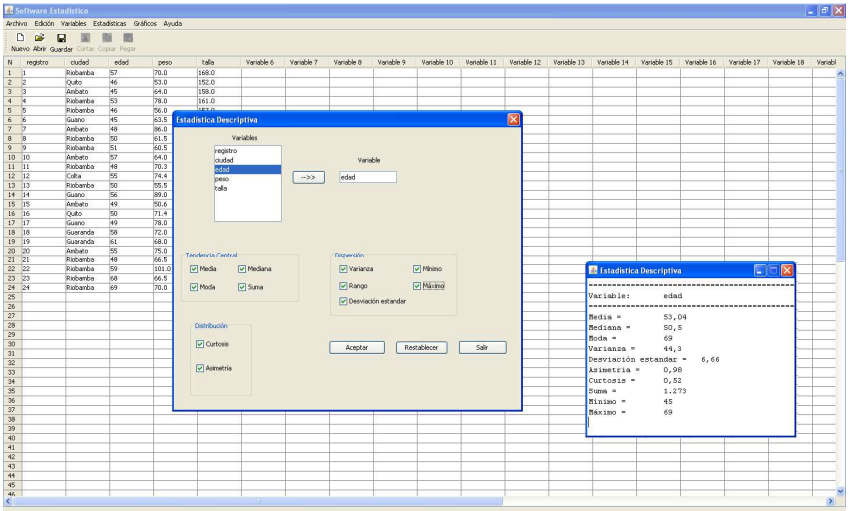


Gráfico 31: Vista de la ventana para realizar estadística descriptiva.

3.6.4 Vista de la ventana para realizar regresión lineal simple.

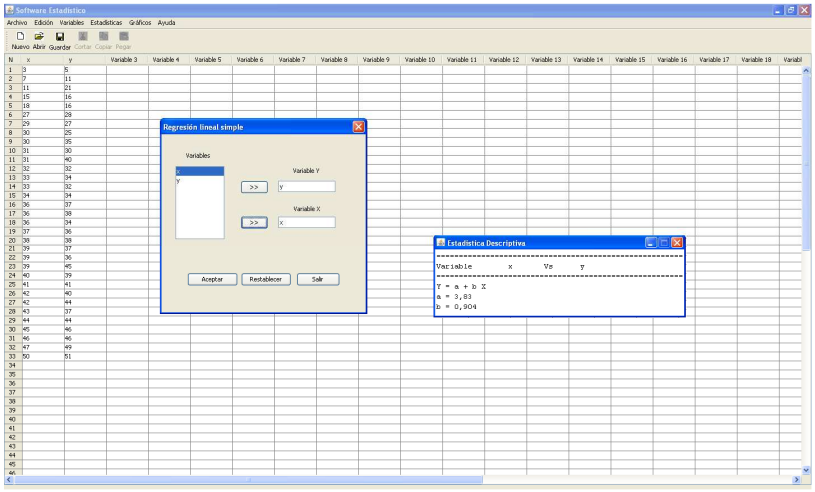


Gráfico 32: Vista de la ventana para realizar regresión lineal.

3.6.5 Vista de la ventana para graficar el diagrama de barras.

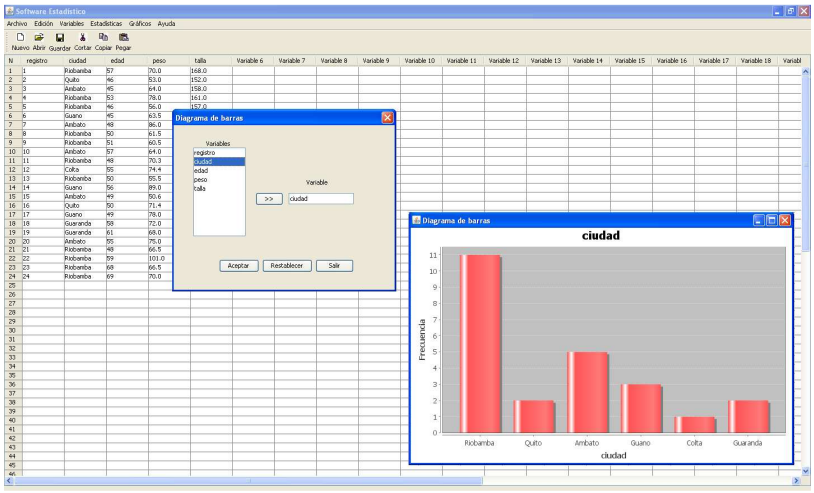


Gráfico 33: Vista de la ventana para graficar el diagrama de barras.

3.6.6 Vista de la ventana para graficar el diagrama circular.

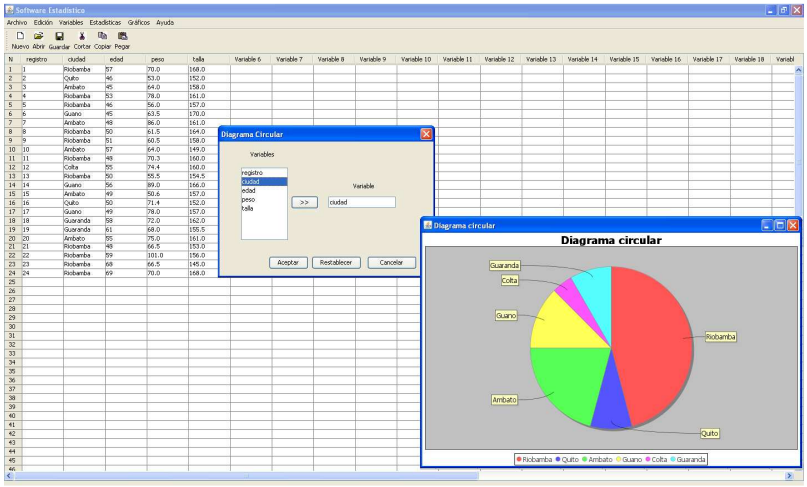


Gráfico 34: Vista de la ventana para realizar el diagrama circular.

3.6.7 Vista de la ventana para realizar la correlación.

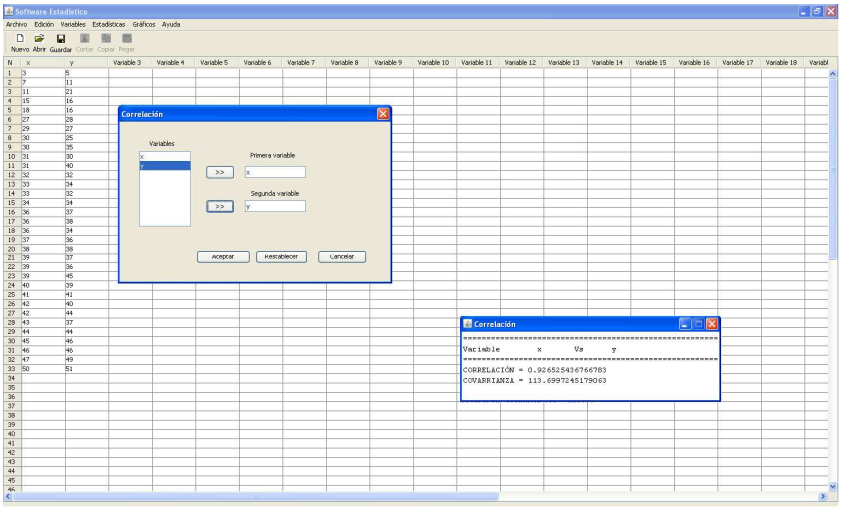


Gráfico 35: Vista de la ventana para realizar la correlación.

3.6.8 Vista de la ventana para realizar la tabla de frecuencias.

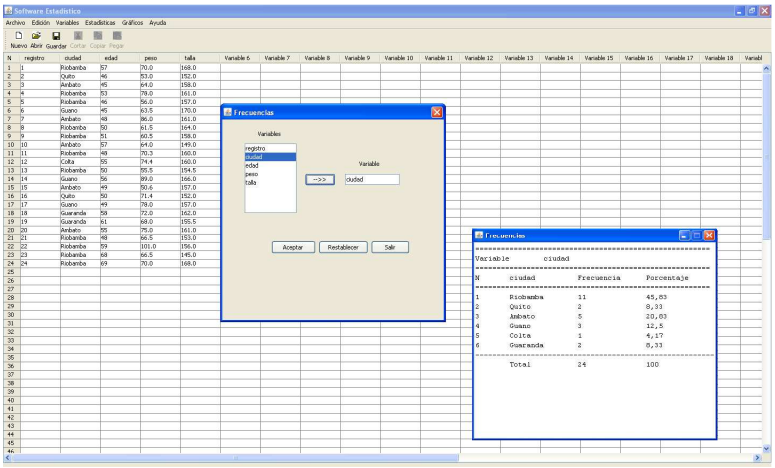


Gráfico 36: Vista de la ventana para realizar la tabla de frecuencias.

3.6.9 Vista de la ventana para realizar el diagrama de líneas.

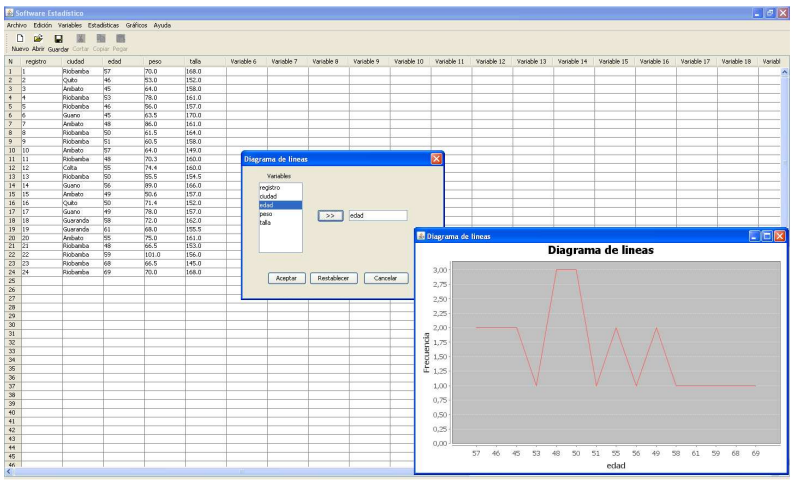


Gráfico 37: Vista de la ventana para realizar el diagrama de líneas.

3.6.10 Vista de la ventana de Acerca de…

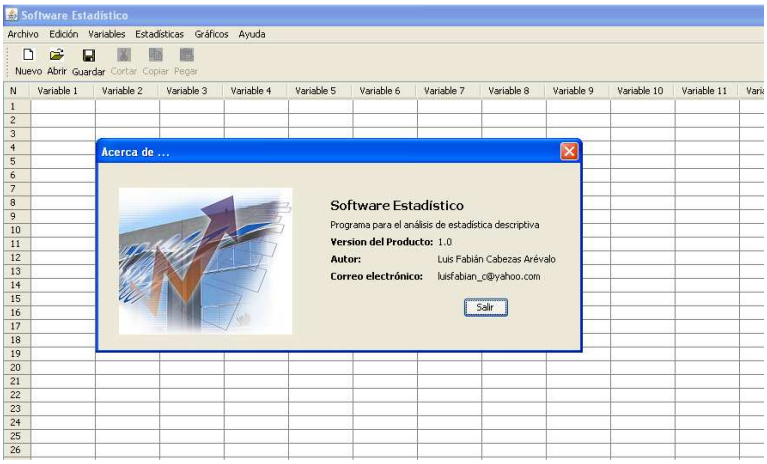


Gráfico 38: Vista de la ventana de Acerca de…

CAPITULO IV

4 CONCLUSIONES Y RECOMEDACIONES

El desarrollo del presente Proyecto de Titulación ha permitido obtener las siguientes conclusiones y recomendaciones:

4.1 CONCLUSIONES

Se ha obtenido un software libre estadístico, aplicable al cuarto nivel de la carrera de Estadística Informática sobre programación y estadística, que ayudara a que los estudiantes en este nivel mejoren notablemente sus conocimientos.

Se ha integrado procesos, métodos y herramientas de ingeniería de software en el desarrollo del software estadístico, que puede servir de base para el desarrollo de otro software estadístico.

El sistema informático desarrollado SoftEstad versión 1.0, permite tener un manejo y análisis de datos para tomar decisiones.

El sistema informático desarrollado SoftEstad versión 1.0 permite desarrollar estadística descriptiva e inferencial clara y concisa.

Se ha integrado armónicamente la tecno ciencia, en este caso la ingeniería de software y la informática, una combinación entre tecnología y educación.

4.2 RECOMENDACIONES

Se recomienda continuar con la línea de investigación y desarrollo de software libre en el análisis de datos para toma de decisiones.

Recomiendo que el presente software sirva de base para la realización de otros software con alcance de análisis de ANOVA, diseño experimental, regresión lineal múltiple, control de calidad, etc.

Se recomienda el trabajo multidisciplinario para la consecución de este tipo de proyectos, para que el software de igual forma sea multidisciplinario

Recomiendo el Java para realización de software libre por su versatilidad y entorno amigable.

RESUMEN

Se diseño y se implemento un software libre estadístico para apoyo académico a docentes y estudiantes del cuarto nivel de la carrera de Estadística Informática de la Facultad de Ciencias de la Escuela Superior Politécnica de Chimborazo (ESPOCH), el software fue implementado con el programa NetBeans IDE 6.5 de java, Se lo realizó mediante técnicas estadísticas como formulas para análisis descriptivo e inferencial, ejemplo: las medidas de tendencia central, dispersión, de distribución, comparación de medias, regresión lineal simple, correlación. Para realización del software se aplico la metodología de ingeniería de software, el modelo en cascada, como software libre estará disponible a los requerimientos del usuario. Se obtuvo un software denominado SoftEstad v1.0 (Software Estadístico). Obteniendo que un 90% de los estudiantes tengan facilidad de adquirir el software y su código fuente, con respecto a los demás software estadísticos.

El software tiene diferentes técnicas para analizar los datos y optimizar la aplicación de técnicas, metodologías y más desarrollo de software libre.

Con el software SoftEstad versión 1.0 se ha logrado un manejo confiable de los datos, disponer de ellos cuando se los requieran y permite generar varios tipos de análisis de datos; en forma rápida y eficaz, para la toma de decisiones. Se recomienda implementar el software desarrollado, en todos los niveles de la carrera en estadística informática.

SUMMARY

A free statistical software was designed and implemented for the academic support of teachers and students of the fourth level of the Informatics Statistics Career of the Science Faculty of the Chimborazo Higher Education Polytechnic School (ESPOCH). The software was implemented with the NetBeans IDE 6.5 of java program. It was carried out through statistical techniques such as formulae for descriptive and inferential analysis. For example: the measurements of central tendency, dispersion, distribution, mean, comparison, simple linear regression and correlation. For the software construction the software engineering methodology, the cascade model were applied. It will be available for the user as free software. A software called SoftEstad v1.0 (Statistical Software) was obtained. A 90% students have the possibility of acquiring the software and its source as compared to the other statistical software. With the SoftEstad version 1.0 software a reliable handling of data and their disposal have been possible when required; it also permits to generate various types of data analyses rapidly and efficiently to make decisions. It is recommended to implement the developed software at all levels of the informatics Statistics Career.

4.3 BIBLIOGRAFÍA

BOUDREAU, Tim. *NetBeans: The Definite Guide.* U.S.A, O'Reilly, 2002. pp. 323-336.

CANAVOS, Goerge. *Probabilidad y Estadística - Aplicaciones y Métodos.* Mexico, McGraw-Hill, 1988. pp. 303-350

HOLMES, James. *The Art of Java.* U.S.A. McGraw Hill/Osborne, 2003. pp. 235-276

HOLZNER, Steven. *La biblia de Java 2 2.ed. Mexico,* CORIOLIS, 2002. pp. 321-518

JOYANES, Luis. *Manual de Programación.* España, McGraw Hill, 2002. pp. 91-276

MONTGOMERY, Douglas y RUNGER, George. *Applied Statistics and Probalility.* U.S.A., Jhon Wiley & Sons , 2003. pp. 277-320

PATCHINE, Alexandre. *1000 Java Tips.* U.S.A., Javaa!, 2003. pp. 42-65

SÁNCHEZ, Jesús. *Programación en java 2.* España, McGraw Hill, 2005. pp.321-333

WALPOLE, Ronald y MYRES, Raymond. *Probabilidad y Estadística para ingenieros. 8.ed.* México, Prentice-Hall, 2006. pp. 321-443

4.4 INTERNET

PROGRAMACIÓN EN NETBEANS IDE

http://wiki.netbeans.org/Java_EditorUsersGuide

2007-10-05

http://inforux.wordpress.com/2008/07/21/aprediendo-con-jcheckbox-y-jradiobutton/

2008-07-21

http://www.forosdelweb.com/f45/actualizar-automaticamente-jlist-563085/

2007-10-02

http://html.rincondelvago.com/falsedades-en-estadistica.html

2002-05-02

http://www.todoexpertos.com/categorias/tecnologia-e-internet/programacion/java/respuestas/891055/jlist-con-jpaneles-dentro

2005-03-26

http://www.magusoft.net/trials/list.html

2002-05-30

http://www.programacion.com/buscar.php?texto=jlist&que=0&idzona=&num=3

2002-10-15

http://www.newsgrupos.com/es-comp-lenguajes-java/19328-jtable-poner-color-una-celda-por-defecto-ii.html

2005-10-05

http://www.chuidiang.com/java/tablas/tablarender/tablarender.php

2006-10-16

http://www.chuidiang.com/java/tablas/tablaeditor/tablaeditor.php

2006-10-14

http://www.geocities.com/chuidiang2/tablas/tablarender/tablarender.html

2006-10-08

http://www.webtutoriales.com/tutoriales/programacion/java/agregar-elementos-jlist.41.html

2007-10-08

http://foro.noticias3d.com/vbulletin/showthread.php?t=95736

2005-12-07

http://www.javahispano.org/forum/j2se/es/anadir_borrar_elementos_de_un_jlist/

2002-15-12

http://www.magusoft.net/trials/list.html

2002-10-09

http://www.lawebdelprogramador.com/news/mostrar_new.php?id=44&texto=Java&n1=12

7351&n2=1&n3=0&n4=0&n5=0&n6=0&n7=0&n8=0&n9=0&n0=0

2004-05-20

http://casidiablo.net/codigo-java-gui-radio-menu-jlist/

2002-10-02

http://javalangnullpointer.wordpress.com/2007/02/22/graficas-con-java/

2007-20-22

http://ayuda-java.blogspot.com/2007/07/cmo-hacer-cuadros-de-dilogo-simples.html

2007-07-22

http://www.javahispano.org/forum/j2se/es/como_manejar_un_jlist_desde_netbeans/

2002-05-12

http://www.programacion.com/java/tutorial/jdcbook/7/

2006-05-15

http://maxus.fis.usal.es/HOTHOUSE/sisinfo/listas.php

2002-12-03

http://afrodita.unicauca.edu.co/~dparedes/java/jdcbook/swing2.html

2008-05-30

http://d.scribd.com/docs/wtoh7xyrcd5ydz80ane.pdf

2009-02-06

http://es.answers.yahoo.com/question/index?qid=20081114085837AAItJm

2007-06-10

http://www.chuidiang.com/chuwiki/index.php?title=JList

2006-02-08

http://www.webtutoriales.com/tutoriales/programacion/java/colecciones.59.html

2009-11-15

http://www.forosdelweb.com/f45/obtener-elementos-jlist-539078/

2008-05-29

http://www.psicofxp.com/forums/programacion.313/342578-consulta-jlist.html

2006-09-12

http://www.lawebdelprogramador.com/preguntas/vercontestada.php?pagina=41&id=44&texto=Java

2009-11-22

http://casidiablo.net/category/programacion/java/ejercicios-en-java/page/4/

2009-11-23

http://java.sun.com/docs/books/tutorial/uiswing/examples/components/index.html#ListDialog

2009-10-01

http://cannes.itam.mx/Alfredo/Espaniol/Cursos/Java/Java.htm

2008-12-02

http://www.foromsn.com/index.php?Ver=Mensaje&Id=198363&VerEtiqueta=33

2008-11-23

http://www.chuidiang.com/java/graficos/libreria/ejemploslibreria.php

2009-05-20

http://www.mygnet.net/codigos/java/modo_grafico/cuadros_de_dialogo_e_inputbox_usando_javax_dot_swing_dot_joptionpane_graphic_user_iterface.1560

2009-12-06

ANEXOS

ANEXO I: DECRETO EJECUTIVO

DECRETO EJECUTIVO No. 1014
RAFAEL CORREA DELGADO
EL PRESIDENTE DE LA REPÚBLICA
CONSIDERANDO:

Que en el apartado g) del numeral 6 de la Carta Iberoamericana de Gobierno Electrónico, aprobada por el IX Conferencia Iberoamericana de Ministros de Administración Pública y Reforma del Estado, realizada en Chile el 1 de Junio de 2007, se recomienda el uso de estándares abiertos y software libre, como herramientas informáticas;

Que es el interés del Gobierno alcanzar soberanía y autonomía tecnológica, así como un significativo ahorro de recursos públicos y que el Software Libre es en muchas instancias un instrumento para alcanzar estos objetivos;

Que el 18 de Julio del 2007 se creó e incorporó a la estructura orgánica de la Presidencia de la República la Subsecretaría de Informática, dependiente de la Secretaría General de la Administración, mediante Acuerdo No. 119 publicado en el Registro Oficial No. 139 de 1 de Agosto del 2007;

Que el numeral 1 del artículo 6 del Acuerdo No. 119, faculta a la Subsecretaría de Informática a elaborar y ejecutar planes, programas, proyectos, estrategias, políticas, proyectos de leyes y reglamentos para el uso de Software Libre en las dependencias del gobierno central; y,

En ejercicio de la atribución que le confiere el numeral 9 del artículo 171 de la Constitución Política de la República;

DECRETA:

Artículo 1.- Establecer como política pública para las Entidades de la Administración Pública Central la utilización de Software Libre en sus sistemas y equipamientos informáticos.

Articulo 2.- Se entiende por Software Libre, a los programas de computación que se pueden utilizar y distribuir sin restricción alguna, que permitan su acceso a los códigos fuentes y que sus aplicaciones. Puedan ser mejoradas. Estos programas de computación tienen las siguientes libertades:

a) Utilización del programa con cualquier propósito de uso común.

b) Distribución de copias sin restricción alguna.

c) Estudio y modificación del programa (Requisito: código fuente disponible).

d) Publicación del programa mejorado (Requisito: código fuente disponible).

Articulo 3.- Las entidades de la Administración Pública Central previa a la instalación del software libre en sus equipos, deberán verificar la existencia de capacidad técnica que brinde el soporte necesario para el uso de éste tipo de software.

Articulo 4.- Se faculta la utilización de software propietario (no libre) únicamente cuando no exista una solución de Software Libre que supla las necesidades requeridas, o cuando esté en riesgo la seguridad nacional, o cuando el proyecto informático se encuentre en un punto de no retorno.

En este caso, se concibe como seguridad nacional, las garantías para la supervivencia de la colectividad y la defensa del patrimonio nacional.

Para efectos de este decreto se entiende por un punto de no retorno, cuando el sistema o proyecto informático se encuentre en cualquiera de estas condiciones:

a) Sistema en producción funcionando satisfactoriamente y que un análisis de costo – beneficio muestre que no es razonable ni conveniente una migración a Software Libre.

b) Proyecto en estado de desarrollo y que un análisis de costo – beneficio muestre que no es conveniente modificar el proyecto y utilizar Software Libre.

Periódicamente se evaluarán los sistemas informáticos que utilizan software propietario con la finalidad de migrarlos a Software Libre.

Articulo 5.- Tanto para software libre como software propietario, siempre y cuando se satisfagan los requerimientos, se debe preferir las soluciones en este orden:

a) Nacionales que permitan autonomía y soberanía tecnológica.
b) Regionales con componente nacional.
c) Regionales con proveedores nacionales.
d) Internacionales con componente nacional.
e) Internacionales con proveedores nacionales.
f) Internacionales.

Articulo 6.- La Subsecretaría de Informática como órgano regulador y ejecutor de las políticas y proyectos informáticos en las entidades del Gobierno Central deberá realizar el control y seguimiento de éste Decreto.

Para todas las evaluaciones constantes en este decreto la Subsecretaría de Informática establecerá los parámetros y metodologías obligatorias.

Articulo 7.- Encárguese de la ejecución de este decreto a los señores Ministros Coordinadores y el señor Secretario General de la Administración Pública y Comunicación.

Dado en el Palacio Nacional en la ciudad de San Francisco de Quito, Distrito Metropolitano, el día de hoy 10 de abril del 2008.

RAFAEL CORREA DELGADO
PRESIDENTE CONSTITUCIONAL DE LA REPÚBLICA

<u>ANEXO II:</u> MANUAL DE USUARIO

SoftEstad

Software Estadístico

Carrera de estadística Informática

Análisis estadístico con SoftEstad versión 1.0 para Windows. Volumen I Estadística Básica Primera Edición.

Riobamba – Ecuador

IMPRESO EN ECUADOR – PRINTED IN ECUADOR

Índice

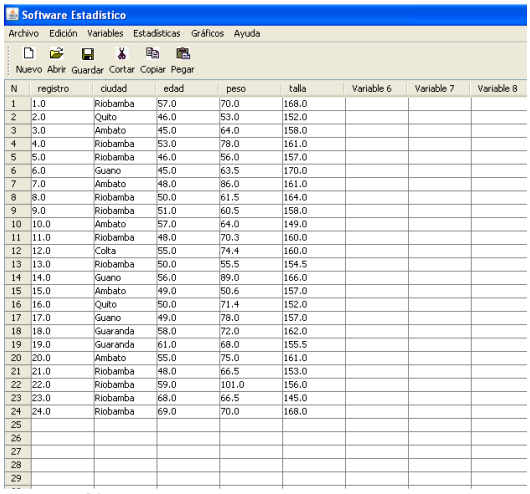
Índice de Gráficos

REQUISITOS

Antes de comenzar, asegúrese de que cuenta con todo el software, necesariamente, debe instalar la máquina virtual, JDK 5.0. o superior para su ejecución.

VENTANAS

Ventana de principal



N	registro	ciudad	edad	peso	talla	Variable 6	Variable 7	Variable 8
1	1.0	Riobamba	57.0	70.0	168.0			
2	2.0	Quito	46.0	53.0	152.0			
3	3.0	Ambato	45.0	64.0	158.0			
4	4.0	Riobamba	53.0	78.0	161.0			
5	5.0	Riobamba	46.0	56.0	157.0			
6	6.0	Guano	45.0	63.5	170.0			
7	7.0	Ambato	48.0	86.0	161.0			
8	8.0	Riobamba	50.0	61.5	164.0			
9	9.0	Riobamba	51.0	60.5	158.0			
10	10.0	Ambato	57.0	64.0	149.0			
11	11.0	Riobamba	48.0	70.3	160.0			
12	12.0	Colta	55.0	74.4	160.0			
13	13.0	Riobamba	50.0	55.5	154.5			
14	14.0	Guano	56.0	89.0	166.0			
15	15.0	Ambato	49.0	50.6	157.0			
16	16.0	Quito	50.0	71.4	152.0			
17	17.0	Guano	49.0	78.0	157.0			
18	18.0	Guaranda	58.0	72.0	162.0			
19	19.0	Guaranda	61.0	68.0	155.5			
20	20.0	Ambato	55.0	75.0	161.0			
21	21.0	Riobamba	48.0	66.5	153.0			
22	22.0	Riobamba	59.0	101.0	156.0			
23	23.0	Riobamba	68.0	66.5	145.0			
24	24.0	Riobamba	69.0	70.0	168.0			
25								
26								
27								
28								
29								

Gráfico 39: Ventana de editor de datos

Al ingresar al Software se visualizará una pantalla de Presentación donde se pondrá los datos con los que se van trabajar, esta ventana se abre automáticamente al iniciar el software SoftEstad versión 1.0.

Menú principal

Archivo Crea un nuevo archivo de texto, Microsoft Excel 2003, abrir uno existente, grabar, leer datos creados en otra aplicaciones, etc.

Edición Contiene las habituales opciones de Windows para copiar, cortar y pegar.

Variables Contiene la opción de poder cambiar el nombre de la variable para su identificación.

Estadísticas Desde esta opción se ejecutan los procedimientos estadísticos.

Gráficos Gráfico de barras, Circular, histograma, de dispersión, etc.

Ayuda Tutorial, asistente estadístico.

Barras de herramientas

Situada debajo de la barra del menús el acceso rápido a una serie de funciones.

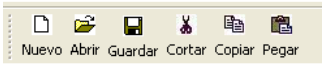


Gráfico 40: Barra de herramientas

Podemos mostrar la barra de herramientas mover simplemente pulsamos con el puntero en cualquier superficie de la misma que no sea un icono y arrastramos o hasta un espacio concreto de la pantalla o hacia los extremos izquierdo o derecho de la misma o hasta los extremos superior o inferior.

Limpia las celdas de datos para ingresar otros datos para analizar.

Abre el cuadro de dialogo para el tipo de documentos en pantalla: datos.

Guarda una base de datos en archivo de datos (txt), Microsoft Excel 2003 (xls) y archivo delimitado (dat) .

Corta el dato seleccionado.

Copia el dato seleccionado.

Pegar el dato que ha sido cortado o copiado.

Cuadros de dialogo

La mayoría de opciones en SoftEstad abren cuadros de dialogo que nos orientan para seleccionar variables y opciones de análisis. Se componen de varios elementos:

- Lista de variables origen disponibles en el archivo.
- Lista la variable seleccionada para el análisis.
- Botones de comandos para ejecutar una acción o para seleccionar especificaciones adicionales.

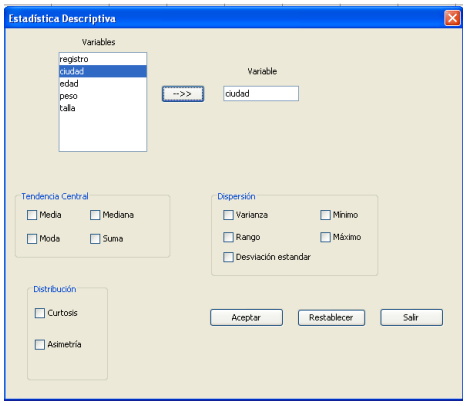


Gráfico 41: Cuadro de dialogo

Procedimientos básicos para un análisis estadístico.

Para llegar a cabo cualquier tipo de análisis con SoftEstad tenemos que realizar estas operaciones básicas:

1. Seleccionar una base de datos.
2. Seleccionar el procedimiento estadístico deseado (menú principal).
3. Seleccionar las variables a incluir en el análisis y otros parámetros adicionales (cuadro de dialogo).

A continuación se explica con un ejemplo dicho proceso: elaboración de una tabla de frecuencias.

Seleccionar un archivo de datos.

- Seleccionar la opción del menú **Archivo/Abrir**

Por defecto, SoftEstad selecciona los archivos con extensión (. txt) pero hay otras posibilidades.

- Seleccionar el archivo Descriptiva.txt y pulsamos Abrir.

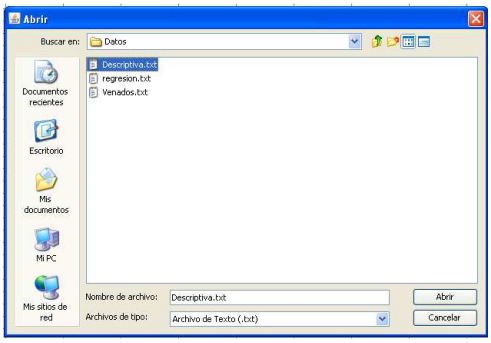


Gráfico 42: Cuadro de dialogo de Abrir.

Seleccionar un procedimiento estadístico.

- Seleccionar la opción del menú principal Análisis/Tablas.

Seleccionar las variables y opciones para el análisis.

En esta ventana podemos observar:

Las variables que contiene el archivo. Pueden ser numéricas o alfanuméricas. Seleccionando una variable de la lista.

El cuadro de la variable seleccionada para el análisis. Será la que figure en el cuadro variable.

Seleccionar la variable Ciudad y pasarla al cuadro de variable destino

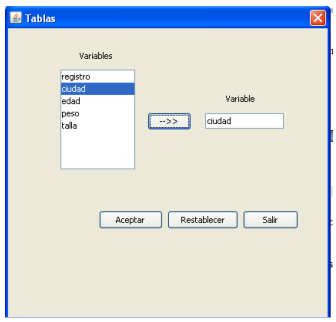


Gráfico 43: Cuadro de dialogo Frecuencias

Podemos seleccionar la variable seleccionando otra resaltando previamente en la lista de variables.

Existen tres botones de comando de ejecución inmediata:

Aceptar: Ejecuta el procedimiento y cierre el cuadro de dialogo.

Restablecer Deselecciona cualquier variable previamente seleccionada y restablece la pantalla de dialogo a las opciones por defecto que tiene al entrar al procedimiento.

Cancelar Cancela cualquier cambio en la pantalla de dialogo desde la última vez que se abrió y cierra la misma.

Ventana de salida

Esta ventana sale al realizar el análisis en cada sesión son ventanas individuales.

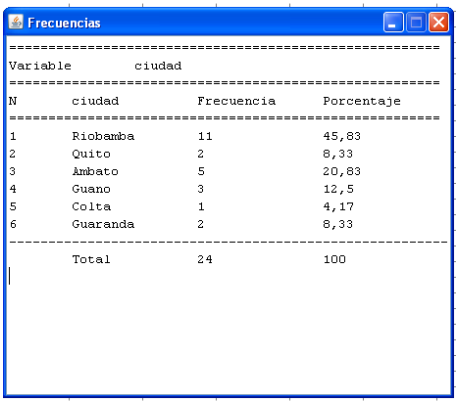


Gráfico 44: Ventana de resultados de frecuencias

Esta ventana se abre al realizar el primer análisis en cada sección y los análisis posteriores se abrirán otras ventanas en cada análisis.

Ventana de Gráficos

Este elemento que no podemos encontrar en la salida de resultados es el de gráficos de todo tipo. Para ello vamos a recuperar el ejemplo ya incorporado en el apartado anterior.

- Volvemos a seleccionar la opción del menú **Gráficos/diagrama de barras**.
- Seleccionamos la variable que deseamos graficar.
- Seleccionamos la tecla Aceptar.

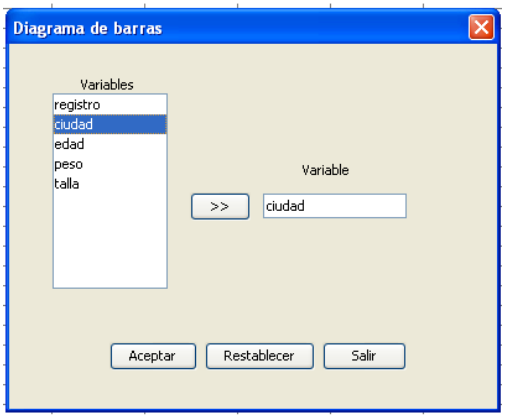


Gráfico 45: Cuadro de dialogo para Diagrama de barras

En la ventana de resultados nos aparecerá el Diagrama de frecuencias de la variable cualitativa que podemos editar haciendo clic derecho sobre el gráfico y que nos dará un submenú la ventana de edición de gráfico.

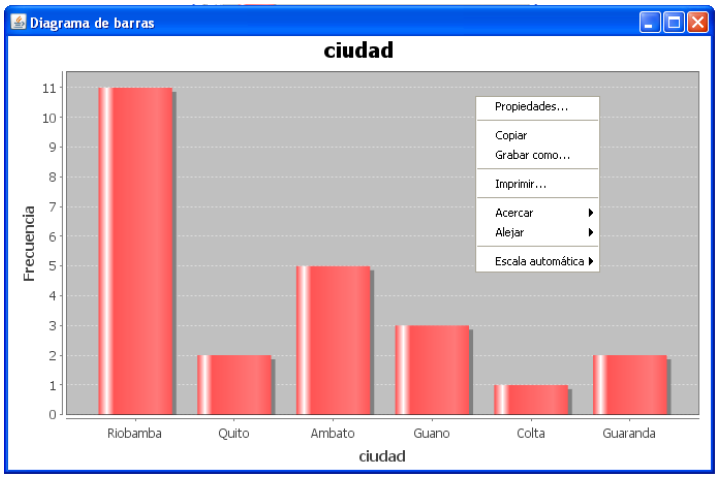


Gráfico 46: Ventana de edición de gráficos.

Propiedades Nos permite cambiar el título del grafico y el color del mismo.

Copiar Permite copiar el gráfico en el clipboard para su posterior pegado.

Grabar como Abre un cuadro de dialogo para guardar el gráfico.

Imprimir Abre un cuadro de dialogo para configurar la impresión.

Acercar Permite acercar el gráfico en forma vertical y horizontal.

Alejar Permite alejar el gráfico tanto vertical como horizontal.

Escala Automática Permite poner el grafico por defecto.

ARCHIVO DE DATOS

Crear un archivo

Podemos utilizar el editor del software para introducir los datos y crear un archivo de datos.

- Seleccionamos **Archivo/Nuevo** del menú principal. Aparecerá una nueva hoja de datos en blanco.

Abrir un archivo

- Seleccionar **Archivo/Abrir** del menú principal. Aparecerá el siguiente cuadro de dialogo.

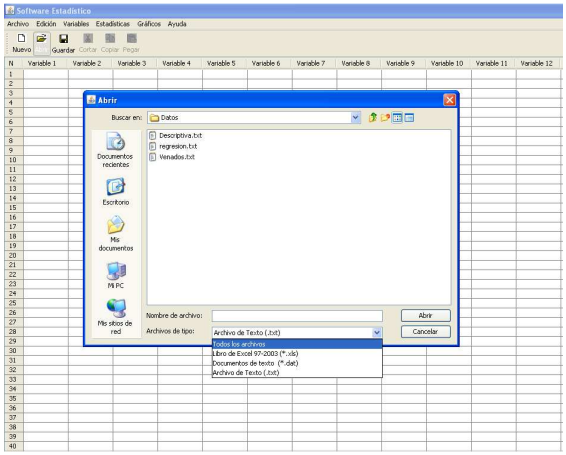


Gráfico 47: Cuadro de dialogo de Archivo/Abrir.

Buscar en: accedemos al directorio donde se encuentra el archivo con el que queremos trabajar.

Nombre del archivo: seleccionamos un archivo de la lista. Por defecto el software nos dará una relación de los archivos de su directorio con extensión .txt.

Tipo de archivos: nos permite seleccionar entre distintos tipos de archivos de datos: Excel, archivo delimitado, archivos de testo ASCII.

- Pulsamos Abrir y cargamos el archivo seleccionado.

Tipo de archivos que reconoce el software.

- Hoja de cálculo:
    - Excel (*.xls) Archivo de Microsoft Excel 97-2003.
- Tab-delimited (*.dat)
    - Archivos de texto ASCII delimitado por tabuladores.
- Texto (*.txt)
    - Archivos de texto ASCII delimitado por tabuladores.

Guardar un Archivo

Cualquier modificación que hagamos en el archivo de datos se mantendrá solo durante el tiempo que dure la sesión.

Podemos guardar los datos en distintos formatos si seleccionamos la opción **Archivo/Guardar** del menú.

Aparecerá un cuadro de dialogo en el que debemos especificar el nombre y el formato en que queremos guardar el archivo. Los formatos disponibles son:

- Tab-delimited (*.dat)
- Excel 97 -2003 (*.xls)
- Texto delimitado (*.txt)

ANALISIS DESCRIPTIVO

Frecuencias

El procedimiento Análisis/Frecuencias permite obtener una descripción de la distribución de la variable a partir de la siguiente información.

- Tabla de frecuencias.

Tabla de frecuencias

- Abrir el archivo a analizar.
- Seleccionar **Análisis/Frecuencias** del menú. En el cuadro de dialogo que aparece seleccionar la variable **ciudad** y pasamos al cuadro de destino tal como se aparece en el cuadro del Gráfico 48: Cuadro de dialogo frecuencias de análisisPulsamos Aceptar para ejecutar el procedimiento. En la ventana de resultados y en contenidos, obtenemos los resultados del Gráfico 49.

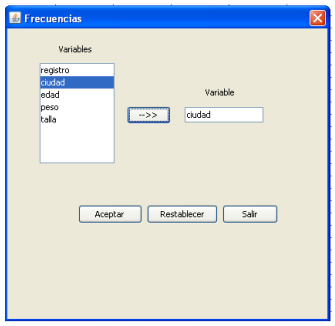


Gráfico 48: Cuadro de dialogo frecuencias de análisis

En la tabla de frecuencias que sigue a continuación tenemos cuatro columnas:

En la primera aparecen el número de casos que ha tenido la variable.

En la segunda aparecen los distintos valores de la variable.

Frecuencia Frecuencia absoluta para cada valor de la variable.

Porcentaje Frecuencia relativa.

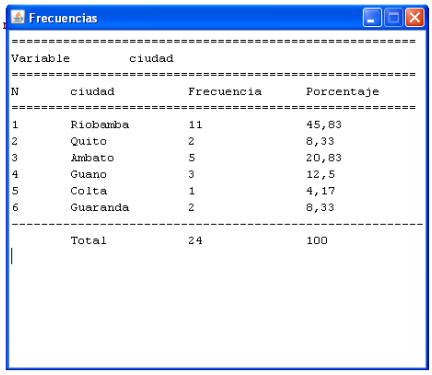


Gráfico 49: Tabla de frecuencias de la variable ciudad

Estadística descriptiva

La opción **Estadística Descriptiva** nos permite calcular algunos estadísticos de tendencia central, de dispersión o de forma.

- Seleccionar de nuevo la opción Análisis/Estadística Descriptiva y previamente cargada la variable. Seleccionamos en esta cuadro de dialogo todos los estadísticos del mismo como tal y como aparecen en el Gráfico 50. Pulsamos Aceptar el procedimiento. Obtenemos la ventana de resultados del Gráfico 51.

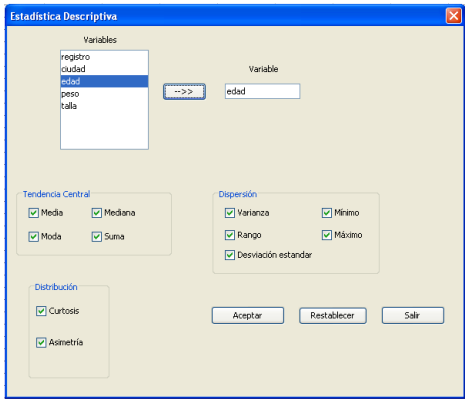


Gráfico 50: Cuadro de dialogo para estadística descriptiva.

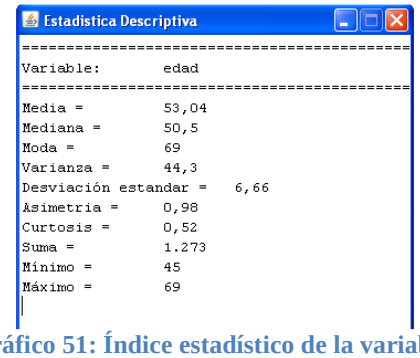


Gráfico 51: Índice estadístico de la variable.

Índice de tendencia central

Este cuadro permite seleccionar cuatro estadísticos de tendencia central:

Media: Media aritmética.

Mediana: Valor por debajo del cual se encuentra el 50 por ciento de los casos.

Moda: Valor que más se repite.

Suma: suma de todos los valores.

Índice de dispersión.

El cuadro dispersión permite seleccionar distintos índices de dispersión.

Desviación tipo o estándar: estimación de la variabilidad de las puntuaciones respecto a la media, expresada en las mismas unidades que los datos.

Varianza: Estimación de la variabilidad de las puntuaciones respecto a la media, expresada en unidades de desviación al cuadrado.

Rango: diferencia entre el valor mínimo y el máximo.

Mínimo: Valor más pequeño.

Máximo: Valor más Grande.

Índice de distribución

El cuadro Distribución permite seleccionar dos índices:

Asimetría: Coeficiente de asimetría.

Curtosis: Coeficiente de curtosis.

Gráficos

Diagrama de barras.

- Seleccionamos la opción Gráficos/Diagrama de barras y previamente cargada las variables. Seleccionamos la variable que deseamos graficar el Gráfico 52. Pulsamos Aceptar el procedimiento. Obtenemos la ventana de resultados del Gráfico 53.

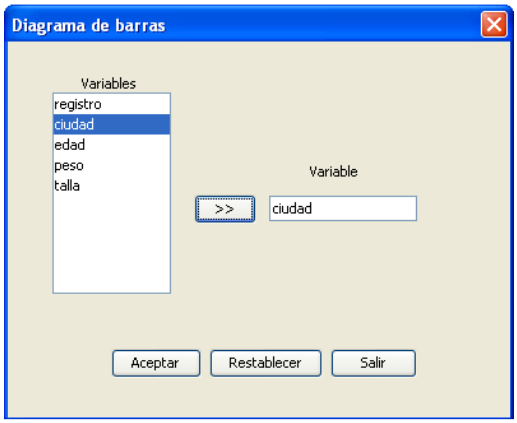


Gráfico 52: Cuadro de dialogo para graficar el diagrama de barras.

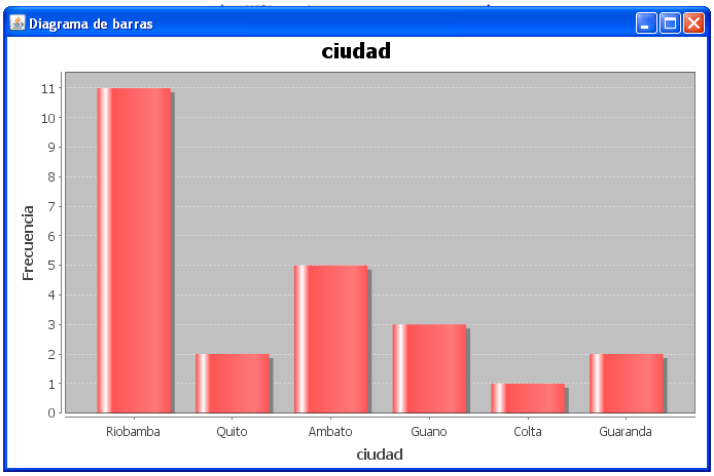


Gráfico 53: Diagrama de barras de la variable ciudad.

Podemos guardar, haciendo clic derecho sobre el gráfico, y nos visualiza un menú para dicha acción.

Histograma.

- Seleccionamos la opción Gráficos/Histograma y previamente cargada las variables. Seleccionamos la variable que deseamos graficar el Gráfico 54. Pulsamos Aceptar el procedimiento. Obtenemos la ventana de resultados del Gráfico 55.

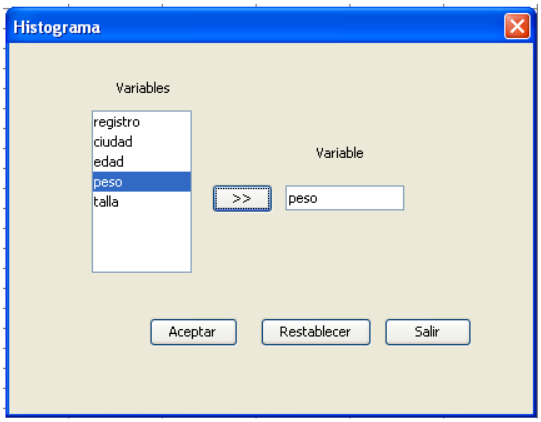


Gráfico 54: Cuadro de dialogo del procedimiento del histograma.

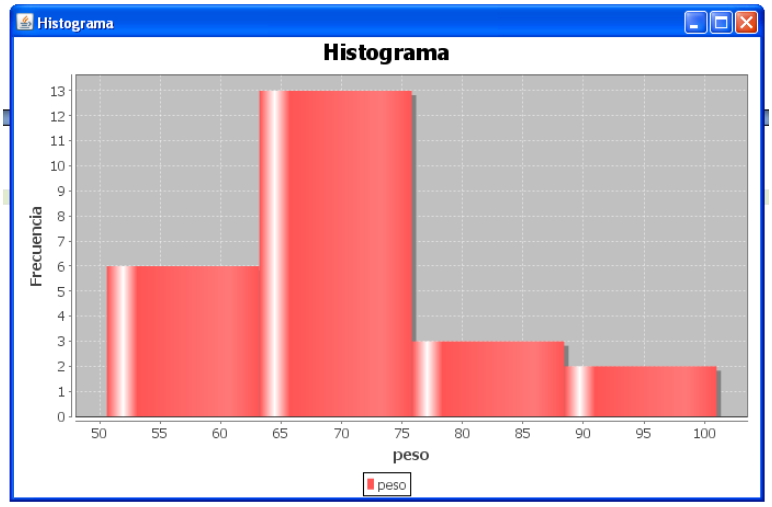


Gráfico 55: Histograma de la variable peso.

El resultado tenemos el histograma de frecuencias.

Cada barra representa el número de casos que toman valores dentro del intervalo. Los intervalos no tiene ninguna observación también están incluidos en el histograma (a diferencia del diagrama de barras, que no deja espacio para categorías vacías).

El histograma de barras es útil para variables en que tiene sentido agrupar valores adyacentes (que estén en una escala ordinal como mínimo). No tendrán sentido agrupar valores de una variable en la que las categorías se hayan asignado arbitrariamente podemos guardar el grafico haciendo clic derecho y poner guardar como… o copiar.

Diagrama circular

- Seleccionamos la opción Gráficos/Diagrama Circular y previamente cargada las variables. Seleccionamos la variable que deseamos graficar el Gráfico 56. Pulsamos Aceptar el procedimiento. Obtenemos la ventana de resultados del Gráfico 57.

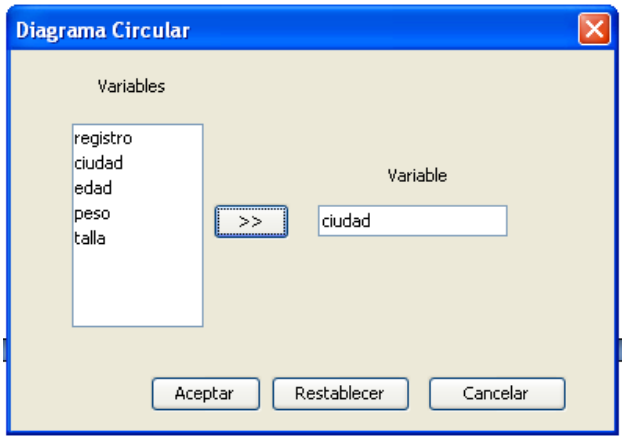


Gráfico 56: Cuadro de dialogo diagrama circular.

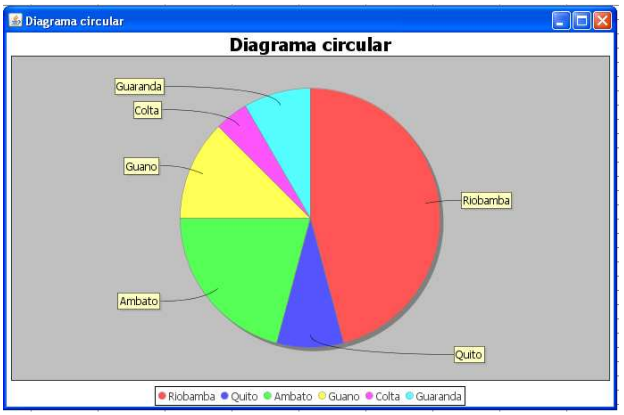


Gráfico 57: Diagrama circular de la variable ciudad.

Se utiliza un diagrama circular para representar datos ordinales, se pierde mucha información.

Diagrama de dispersión.

- Seleccionamos la opción Gráficos/Diagrama de dispersión y previamente cargada las variables. Seleccionamos la variable que deseamos graficar el Gráfico 58: Cuadro de dialogo Diagrama de dispersión.. Pulsamos Aceptar el procedimiento. Obtenemos la ventana de resultados del Gráfico 59: Diagrama de dispersión de la variable peso vs talla..

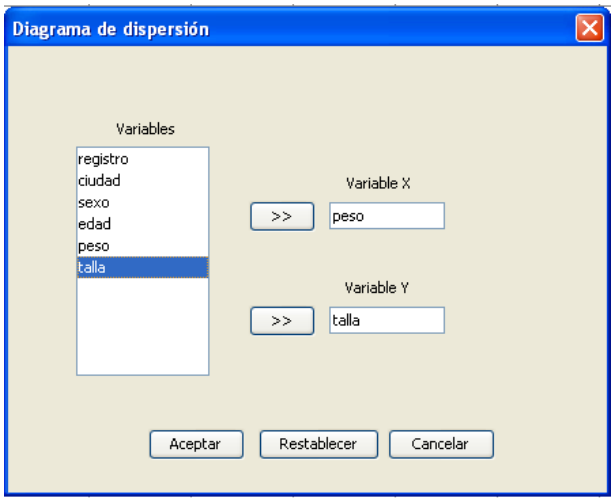


Gráfico 58: Cuadro de dialogo Diagrama de dispersión.

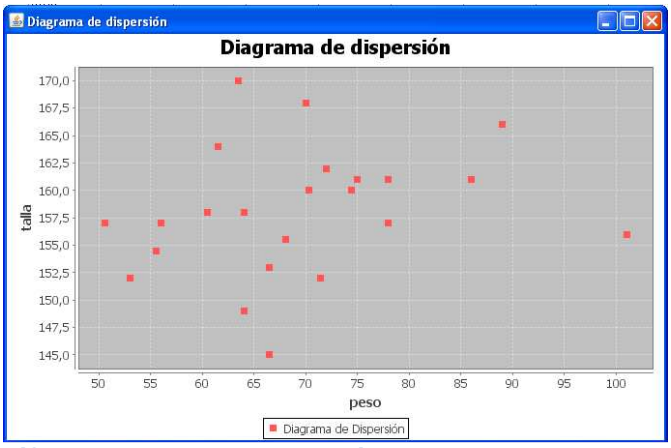


Gráfico 59: Diagrama de dispersión de la variable peso vs talla.

El diagrama de dispersión muestra una correlación entre la variable peso y la variable talla.

Diagrama de línea

- Seleccionamos la opción Gráficos/Diagrama de Dispersión y previamente cargada las variables. Seleccionamos la variable que deseamos graficar el Gráfico 60. Pulsamos Aceptar el procedimiento. Obtenemos la ventana de resultados del Gráfico 61.

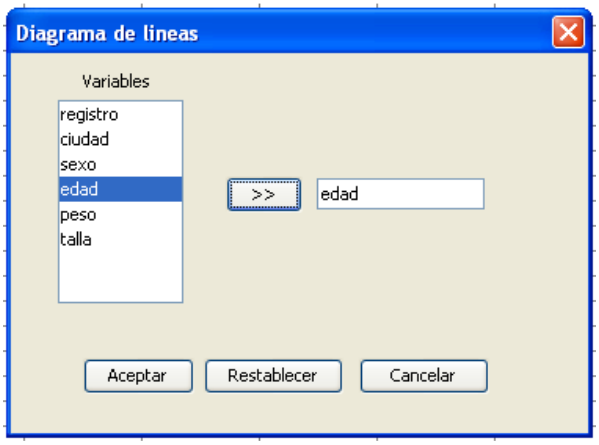


Gráfico 60: Cuadro de dialogo Diagrama de líneas.

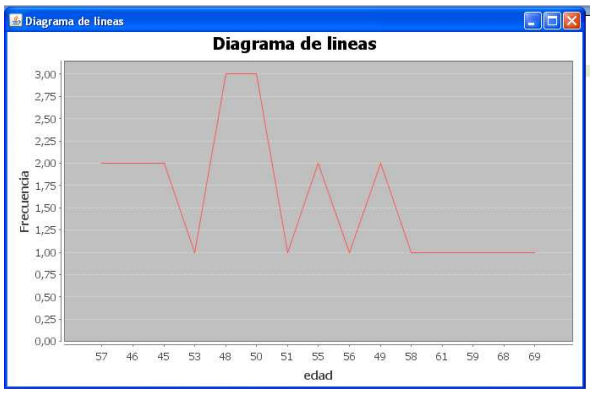


Gráfico 61: Diagrama de líneas de la variable edad

El diagrama de líneas nos visualiza la forma como está distribuido los datos.

COMPARACIÓN DE MEDIAS

Una muestra

En esta prueba se trata de comprobar la hipótesis nula (Ho) de la no existencia de diferencias significativas entre la media de una muestra por ejemplo:

Supongamos que un ingeniero se interesa en probar el sesgamiento en un medidor de pH. Se reúnen datos de una sustancia neutra (pH=7.0). Se toma una muestra de las mediciones y los datos son los siguientes.

7.07	7.00	7.10	6.97	7.00
7.03	7.01	7.01	6.98	7.08

Es entonces, de interés probar

Ho: $\mu = 7.0$
H_1: $\mu \neq 7.0$
$\alpha = 0.05$

- Entramos en Estadísticas/Comparación de medias/Prueba sobre una muestra y obtendremos el cuadro de dialogo del Gráfico 62 En variable introducimos la variable pH, y como valor de prueba 7.0 que es el sustancia neutra y en intervalo de confianza dejamos por defecto y ejecutamos el procedimiento, obtendremos los resultados del Gráfico 63.

Gráfico 62: comparación de medias: una muestra.

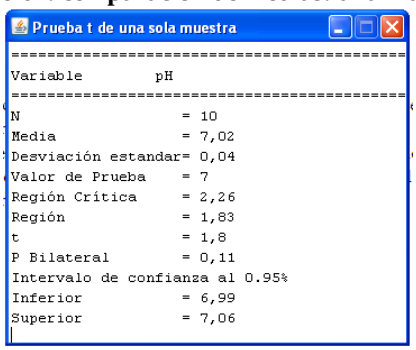


Gráfico 63: Resultado de la prueba t con una muestra.

El valor P de 0.11 sugiere resultados no concluyentes. No hay un fuerte rechazo de Ho (con una base en una α de 0.05 0 de 0.10).

Dos muestras con datos independientes.

En esta prueba, de contrastar la hipótesis nula (Ho) de no existencia de diferencias significativas entre las medias de dos muestras distintas de individuos.

Para indagar si un nuevo suero frena el desarrollo de la leucemia, se selecciona 9 ratones, todos con una etapa avanzada de la enfermedad. Cinco ratones reciben el tratamiento y cuatro no. Los tiempos de supervivencia, años a partir del momento en que comienza el experimento son los siguientes:

Con tratamiento	2.1	5.3	1.4	4.6	0.9
Sin tratamiento	1.9	0.5	2.8	3.1	

Entonces la prueba de hipótesis queda de la siguiente manera:

$$Ho: \quad \mu_1 - \mu_2 = 0$$

$$H_1: \quad \mu_1 - \mu_2 \neq 0$$

$$\alpha = 0.05$$

- Seleccionamos, estando cargado el archivo. Vamos estadísticas/Comparación de medias/ Prueba sobre dos media independientes. Obtendremos el cuadro de dialogo del Gráfico 64. Entramos a como primera variable y segunda variable las variables a ser contrastarlas. Pulsamos Aceptar para ejecutar el procedimiento y obtenemos los resultados del Gráfico 65

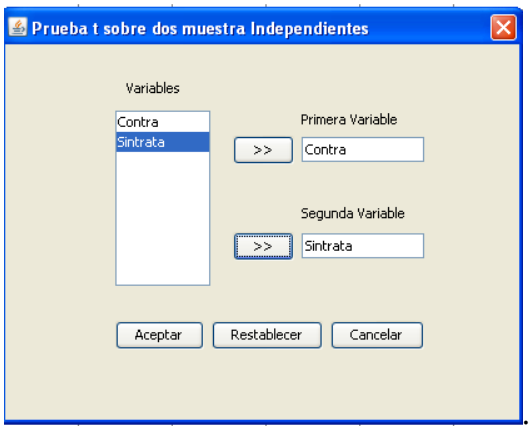


Gráfico 64: Cuadro de dialogo Prueba t de variables independientes.

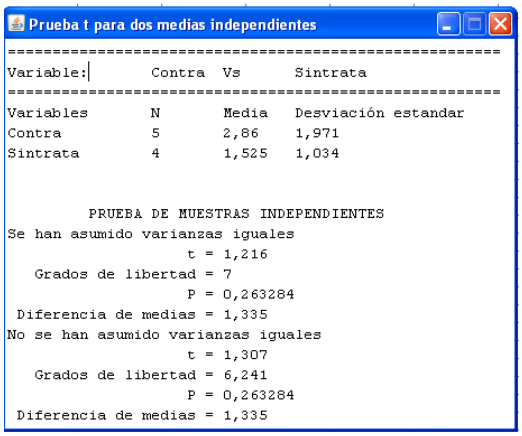


Gráfico 65: Resultados prueba t con variables independientes.

Como vemos en el grafico precedente p es mayor que 0.05 quiere de decir que no se rechaza la hipótesis nula.

Dos muestras con datos dependientes.

Se trata de contrastar la hipótesis nula de la no existencia de diferencias significativas entre las medias de dos muestras pero en caso con datos apareados.

Sena sujetos distintos en ambos grupos pero que sean comparables para a par respecto a una serie de características o circunstancias de investigación o experimentación. Es en este último caso, el de la experimentación.

En un estudio examino la influencia del fármaco succinylcholine sobre los niveles de circulación de andrógenos en la sangre. Se obtuvieron muestras de sangre de venados salvajes vía vena yugular inmediatamente después de una investigación intramuscular succinylcholine con un rifle de caza con dardos. Los venados se sangraron nuevamente aproximadamente 30 minutos después de la inyección y después se liberaron Los niveles de andrógenos al momento de la captura y 30 minutos después, medidos en nano gramos por mililitros (ng/ml), para 15 venados son los siguientes:

Venado	Al momento de la inyección	Andrógenos (ng/ml) 30 minutos después de la inyección	di
1	2.76	7.02	4.26
2	5.18	3.10	-2.08
3	2.68	5.44	2.76
4	3.05	3.99	0.94
5	4.10	5.21	1.11
6	7.05	10.26	3.21
7	6.60	13.91	7.31
8	4.79	18.53	13.74
9	7.39	7.91	0.52
10	7.30	4.85	-2.45
11	11.78	11.10	-0.68

12	3.90	3.74	-0.16
13	26.00	94.03	68.03
14	67.48	94.03	26.55
15	17.04	41.70	24.66

Entonces la prueba de hipótesis queda de la siguiente manera:

$$Ho: \mu_1 = \mu_2 \quad o \quad \mu_D = \mu_1 - \mu_2 = 0$$
$$H_1: \mu_1 \neq \mu_2 \quad o \quad \mu_D = \mu_1 - \mu_2 \neq 0$$
$$\alpha = 0.05$$

- Entramos en Estadísticas/comparación de medias/prueba sobre dos muestras dependientes y obtenemos el cuadro de dialogo del Gráfico 66. Hacemos clic con el botón asignación las variables y ejecutamos el procedimiento obtendremos los resultados en el Gráfico 67.

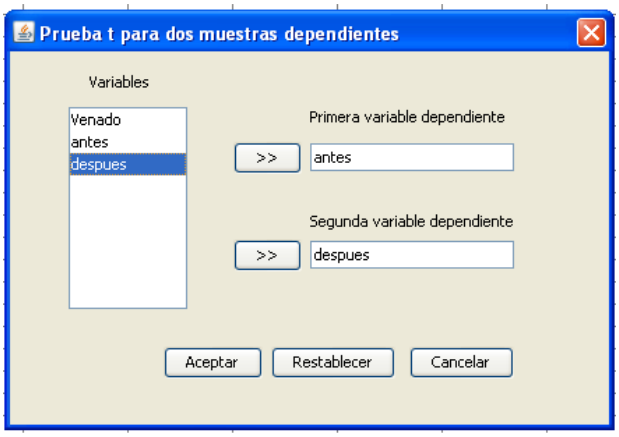


Gráfico 66: Cuadro de dialogo prueba t para muestras independientes.

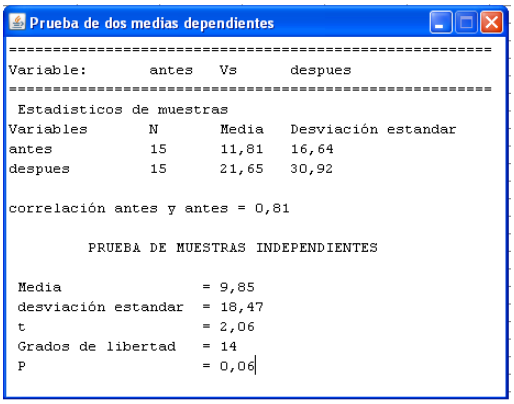


Gráfico 67: Resultados prueba t variable dependientes.

Como vemos en el Gráfico 67. Vemos que el estadística t no es significativa al nivel 0.05.

REGRESIÓN LINEAL SIMPLE

El análisis de regresión lineal es una técnica estadística utilizada para estudiar la relación entre variables.
Tanto en el caso de dos variables (regresión lineal), el análisis de regresión lineal puede utilizarse para explorara y cuantificar la relación entre una variable llamada dependiente o criterio (Y) y una variable llamada independiente o predictora (X1), así como para desarrollar una ecuación lineal con fines predictivos.

Para llevar a cabo un análisis de regresión con las especificaciones tiene establecidas por defecto.

- Seleccionar la opción Estadísticas/Regresión para acceder al cuadro de dialogo regresión lineal que muestra el Gráfico 68.

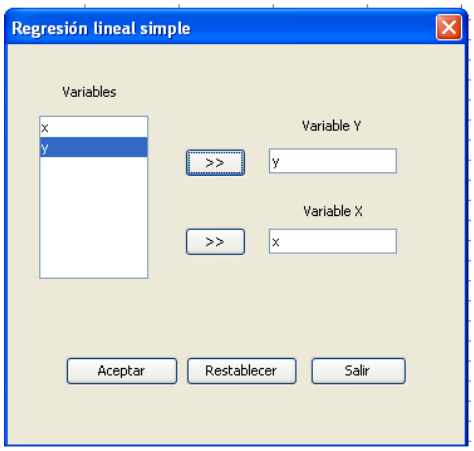


Gráfico 68: Cuadro de dialogo Regresión lineal simple.

- Seleccionamos la variable Y en la lista de variables y trasladamos al cuadro de Variable Y.
- Seleccionar la variable X al cuadro de Variable X.

Con sólo estas especificaciones, al pulsar el botón Aceptar el visor los resultados que muestra el

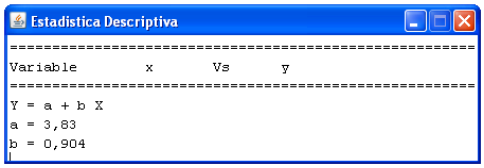


Gráfico 69: Coeficientes de regresión lineal.

En el Gráfico 69 precedente muestra los coeficientes de la recta de regresión.

El coeficiente corresponde a la constante es el origen de la recta de regresión (lo que hemos llamado ***a***), y el coeficiente correspondientes a Y es la pendiente de la recta de regresión (lo que hemos llamado ***b***):
b indica el cambio medio que corresponde a la variable dependiente por cada unidad de cambio de la variable independiente. Según, la ecuación de regresión queda de la siguiente manera.
Pronostico en Y = 3.083 + 0.904X.

A cada valor de X le corresponde un pronóstico de Y.

CORRELACIÓN

El coeficiente de correlación lineal simple, r, mide el grado de asociación lineal entre dos variables medidas en escala de intervalo o de razón, tomando valores entre -1 y 1. Valores de r próximos a indicarán fuerte asociación lineal positiva o directa, los valores cercanos a -1 indicarán fuerte asociación lineal negativa o inversa y valores próximos a 0 indicarán no asociación lineal (lo que no significa que no pueda haber otro tipo de asociación) El estimador muestral para r es el coeficiente de correlación muestral r.

Para obtener el coeficiente de correlación:

- Seleccionamos la opción Estadísticas/Correlación para acceder al cuadro de dialogo de correlación que muestra el Gráfico 70.

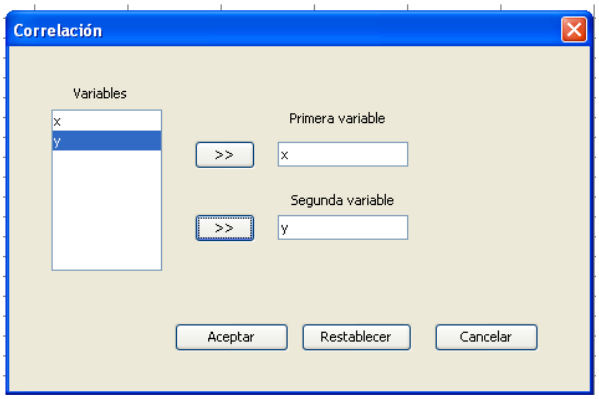


Gráfico 70: Cuadro de diálogo correlación

La lista de variables muestra las variables del archivo. Desde este cuadro de dialogo obtenemos el coeficientes de correlación.

El coeficiente de correlación que utilizamos en el de Pearson, se suele representar por y se obtiene tipificando el promedio de los productos de las puntuaciones diferenciales de cada caso (desviaciones de la media) en las dos variables correlacionadas.
El coeficiente de correlación de Pearson toma valores entre -1 y 1: un valor de 1 indica relación lineal perfecta positiva; un valor de -1 indica relación lineal negativa (en ambos casos los punto se encuentran dispuestos en un alinea recta); un valor 0 indica relación lineal nula. El coeficiente r es una medida simétrica: la correlación entre X y Y es la misma que Y y X.

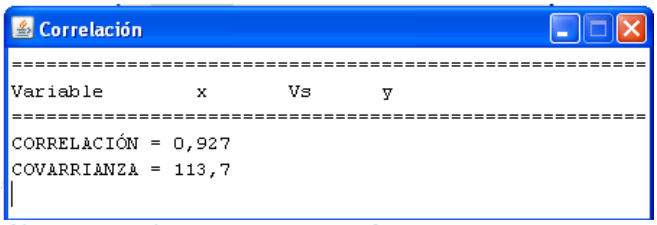


Gráfico 71: Coeficiente de correlación de Pearson y covarianza.

Como vemos en el Gráfico 71, el coeficiente de correlación entre la variable X y Y es de 0.927es quiere decir que tiene una relación fuerte y positiva.