Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de herramienta informática de apoyo a la docencia de Estadística

Grado en Ingeniería Mecánica

ALUMNO: AA. De La Torre Diaz, Juan Antonio

DIRECTORES: Castro Cao, Sandra

CURSO ACADÉMICO: 2015-2016

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Desarrollo de herramienta informática de apoyo a la docencia de Estadística

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Infantería de Marina

Universidade Vigo

RESUMEN

El presente trabajo nace de la idea de desarrollar una interfaz de usuario en MATLAB ® que permita al alumno alcanzar una mayor comprensión de los conceptos relacionados con la Estadística mediante el uso práctico de los mismos en un entorno gráfico que facilite su asimilación.

Con él, se pretende:

- Que sirva como herramienta para complementar la formación del alumno tanto en cuestiones de teoría como de práctica de la asignatura de estadística.
- Mostrar de una manera gráfica los distintos tipos de problemas que pudiesen surgir a la hora de resolver ejercicios, así como sus soluciones o distintos gráficos que ayuden mejor a su comprensión.
- Que el alumno pueda interactuar con el programa informatico de manera que afiance sus conocimientos en estadística, así como que el profesor tenga más apoyo a la hora de impartir su asignatura.

Para ello, se ha tenido en cuenta que el usuario que se enfrenta a esta aplicación tiene un conocimiento muy básico acerca del programa informático MATLAB y únicamente tiene conocimientos de estadística adquiridos durante su bachillerato. La interfaz empleada trata, de una manera sencilla, de facilitar el aprendizaje por parte de los alumnos de la asignatura.

PALABRAS CLAVE

Estadística, Docencia, Matlab ®, Herramienta informática, Interfaz Gráfica de Usuario

AGRADECIMIENTOS

A mi familia por su inestimable apoyo desde que entré en la Escuela Naval.
A mi tutora la profesora Sandra Castro Cao por su gran dedicación y seguimiento durante todo el duro proceso de trabajo de este TFG.
Y por último agradecer a mis compañeros por su apoyo, en especial a mi camareta, sin el ambiente de estudio que hemos tenido no hubiera sido posible sacar este proyecto adelante.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Introducción

La Estadística es la rama de las matemáticas que se ocupa del estudio y la aplicación del conjunto de métodos necesarios para recoger, clasificar, representar y resumir los datos de un experimento aleatorio, así como para la realización de inferencias a partir del análisis de éstos. Trata de la recopilación, organización, análisis e interpretación de datos numéricos con el propósito de obtener inferencias a partir del cálculo de probabilidades. El estudio de la estadística es realmente necesario debido a que hoy en día se ha convertido en un método efectivo para describir con gran margen de fiabilidad las tendencias y valores de datos económicos, políticos, sociales, psicológicos, biológicos y físicos, y sirve como herramienta para relacionar y analizar dichos datos [1].

El término alemán Statistik, introducido originalmente por Gottfried Achenwall en 1749, se refería al análisis de datos del Estado, es decir, la «ciencia del Estado». Aun así, en la antigüedad, esta disciplina matemática ya se utilizaba mediante representaciones gráficas y otras medidas en pieles, rocas, palos de madera y paredes de cuevas para controlar el número de personas, animales o ciertas mercancías.

Civilizaciones como la de los babilonios, hacia el año 3000 a. C. usaban ya pequeños envases moldeados de arcilla para recopilar datos sobre la producción agrícola y de los géneros vendidos o cambiados. Los egipcios analizaban los datos de la población y la renta del país mucho antes de construir las pirámides en el siglo XI a. C. Los griegos realizaban censos cuya información se utilizaba hacia el 594 a. C. para cobrar impuestos. Los libros bíblicos de Números y Crónicas incluyen en algunas partes trabajos de estadística, e incluso en China existían registros numéricos similares con anterioridad al año 2000 a. C. En resumen, la Estadística aunque no definida como en la actualidad, ha sido un método utilizado desde el comienzo de la civilización [2].

Entre los principales científicos cuyas aportaciones a la ciencia han posibilitado el desarrollo de esta podemos destacar a Kolmogórov el cual fue un pilar en la formulación del modelo fundamental de la Teoría de Probabilidades, Thomas Simpson en 1755 por la Teoría de errores, Laplace (1774) quien representó la Ley de probabilidades de errores mediante una curva y dedujo una fórmula para la media de tres observaciones y Bernoulli (1778) quien introduce el principio del máximo producto de las probabilidades de un sistema de errores concurrentes. A su vez destacan otros matemáticos en este ámbito como Carl Friedrich Gauss (1824), Friedrich Bessel (1838), Silvestre Lacroix (1816), Francis Galton (1901) o Karl Pearson (1901) entre otros tantos quienes con sus teorías ayudaron a la evolución de la Estadística y supusieron un gran cambio en su entendimiento [2].

La Estadística dentro del ámbito de la Armada cobra vital importancia. Desde la incorporación del nuevo plan de estudios, los alumnos de la Escuela Naval Militar necesitan de sus créditos

correspondientes para egresar como oficiales. Esta asignatura es estudiada durante el primer curso de ingeniería impartida por el Centro Universitario de la Defensa. Los alumnos provenientes del bachillerato suelen tener dificultad al afrontarla debido a que los conocimientos previos adquiridos suelen ser escasos, en especial aquellos que hicieron un bachillerato de letras.

Dentro de la Armada, una vez superados los años de escuela, no deja de tener importancia. Es de gran relevancia sobre todo en temas de logística, donde hace falta realizar estimaciones acerca de material, ya sea para aprovisionamiento, mantenimiento, y funciones logísticas de personal.

Hoy en día la sociedad se encuentra rodeada por las nuevas tecnologías por lo que es importante incorporarlas en la labor escolar cotidiana. Estas pueden facilitar el trabajo más rutinario de la matemática para dedicar el tiempo a tareas más complejas e interesantes en las que se busca que el uso de la tecnología favorezca la construcción de conocimiento por parte de los alumnos. Según una serie de estudios el uso de imágenes, gráficos o programas informáticos, permiten una mejor comprensión de los conceptos [1].

1.2 Objetivos

Este trabajo tiene como finalidad principal la enseñanza de la asignatura de Estadística de la Universidad acercando el mundo de las nuevas tecnologías al estudiante resultando de ayuda tanto al docente como al alumno en las tareas de enseñanza-aprendizaje. El trabajo a priori será realizado en el programa informático MATLAB® (Laboratorio de Matrices), a través del cual se realizarará una interfaz gráfica y se creará un entorno en donde el alumno pueda realizar diferentes tipos de ejercicios, así como comprobar sus resultados.

El presente TFG (Trabajo de Fin de Grado) cuenta con tres objetivos principales:

- Servir de ayuda para entender mejor la asignatura. Esto será posible gracias al uso de la interfazaz donde los alumnos podrán comprobar el cómo realizar los ejercicios, y visualizarlos desde otro punto de vista que no sea el visto en las clases.
- Aumentar la participación del alumno en la asignatura. Tanto para la resolución de ejercicios como para la comprobación de aquellos resueltos, el alumno deberá tener un cierto conocimiento. No será posible obtener resultados si el alumno apenas ha estudiado la asignatura. De este modo se aumenta su capacidad de aprendizaje y su capacidad resolutiva.
- Facilitar la labor docente del profesor de la asignatura. Gracias al uso de esta interfaz, este podrá ofrecer a los alumnos un punto de vista mucho más gráfico. Además, estos ejercicios podrán ser modificados para proveer al alumno de nuevos enunciados y que no le resulten repetitivos. Gracias a esto el profesor podrá dedicar más tiempo al alumno y facilitará su labor como docente.

1.3 Metodología

Para su realización se siguiran los siguientes pasos:

- Estudio de los recursos TIC (Tecnologías de la información y Comunicaciones) más utilizados en la docencia actual en general y en la estadística en particular.
- Estudio de los posibles entornos de programación a utilizar para desarrollar la herramienta, para valorar el uso de otra alternativa distinta a MATLAB. Los resultados de este punto, así como del anterior, se recogen en el Capítulo 2 de la presente memoria.
- Desarrollo de la aplicación con el entorno elegido. Los contenidos de este punto se recogen en el Capítulo 3 de la presente memoria.
- Validación de la aplicación desarrollada. Los resultados de este punto se recogen en el Capítulo 4 de la presente memoria.

- Extracción de conclusiones una vez validada la aplicación, y estudio de posibles líneas futuras de trabajo sobre la misma. El contenido de este punto se recoge en el Capítulo 5 de la presente memoria.

2 ESTADO DEL ARTE

2.1 Tecnología en las aulas

En la última década los avances tecnológicos han sido extraordinarios, y estos han sido aplicados a la educación para ofrecer lo mejor y lo más importante en experiencias para los alumnos. Está absolutamente comprobado que el uso de la tecnología multimedia mejora la capacidad de comprensión de los alumnos y al mismo tiempo reduce el tiempo de instrucción y los costes de la enseñanza [4].

Los alumnos requieren para su futuro profesional de la utilización de los medios tecnológicos para adaptarse a las condiciones del mundo actual; los cuales facilitan el aprendizaje [13]. Una de las principales ventajas del uso de medios audiovisuales en las aulas son que el aprendizaje es más permanente, un aumento significativo de la atención, del interés y la estimulación de la actividad por parte del alumno y que desarrollan la continuidad de pensamiento.

La demanda de empleo en el mundo actual, requiere de una preparación en la cual se necesita del conocimiento de todo aquello que tiene que ver con las nuevas tecnologías que se han ido desarrollando, así como de todo aquello que resulte beneficioso para la empresa de la cual vayan a formar parte, es decir, que facilite su inserción laboral y profesional.

Los docentes utilizando estos nuevos medios actuales pueden liberarse de realizar trabajos de orientación. Los nuevos métodos de enseñanza hacen que el procedimiento utilizado por los profesores sea muy distinto al usado hace tan solo 10 años, y hacen que estos puedan proporcionar información acerca de lo que imparte de forma más sencilla y asequible. A su vez esto influye en la accesibilidad del alumno a los contenidos impartidos. Los profesores ya como individuos, se encuentran liberados de trabajos rutinarios como era la simple lectura de libros de texto en clase y pueden hacer el trabajo verdaderamente profesional, creativo y entretenido, cosa que hasta no hace mucho se había descuidado. Esto a su vez supone una participación más activa en el aula que por consecuencia finaliza con una mejor comprensión de la materia [5].

El uso de la tecnología para mejorar la comunicación obliga a cambiar los métodos rutinarios por otros más ágiles para alcanzar las metas educativas. Uno de los problemas que surgen son la resistencia por parte del personal docente a realizar estos cambios debido a que están convencidos que mediante los antiguos métodos, la enseñanza es la misma. Según estudios realizados se demuestra que dicha afirmación no es correcta. En la Figura 2-1 se muestra el porcentaje de conocimientos que se adquieren a través de los sentidos, siendo bastante resaltable el de la vista.

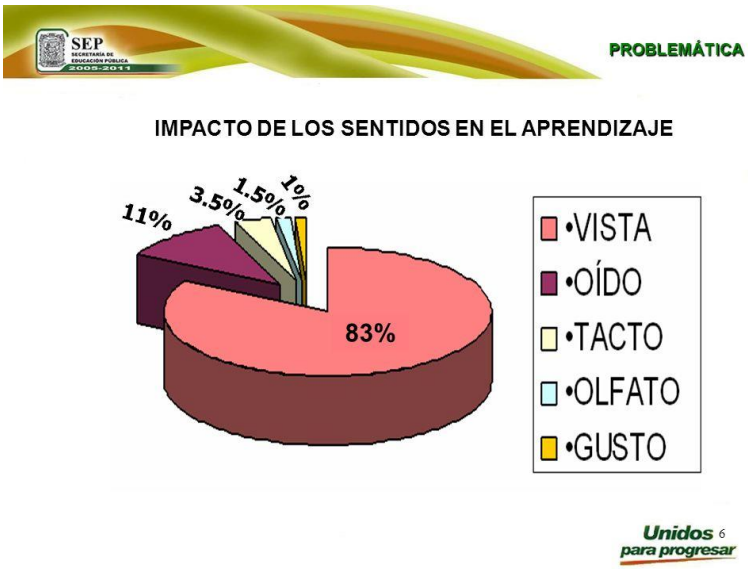

Figura 2-1 Impacto de los sentidos en el aprendizaje [19]

A pesar de esto, se cree que los medios tecnológicos deshumanizan la enseñanza, desplazan a los profesores y que la educación se hace autómata sin calidad humana, empeora la comunicación entre profesor y alumno.

En conclusión lo que se pretende con el uso de estos métodos es liberar al profesor del trabajo rutinario y centrarlo más en ser un guía para el alumno en el desarrollo de su capacidad.

2.1.1. Aulas Virtuales

Las aulas virtuales son un nuevo concepto en educación a distancia que ya se utiliza en muchas universidades a nivel mundial y en algunas otras entidades dedicadas a la ayuda y apoyo de los estudiantes [5]. El uso de aulas virtuales se ha multiplicado en las universidades, e instituciones de docencia con el paso del tiempo. Estas nacen con la idea de que los maestros puedan proporcionar a los alumnos el material de la asignatura, información que pueda ser útil así como archivos de diferentes investigadores que puedan ayudar a una mejor comprensión de la misma [7]. A su vez, permite la subida de ejercicios a la web, en la cual los alumnos pueden realizarlos y adjuntar sus soluciones, facilitando la labor del profesor a la hora de su corrección. Un ejemplo de aulas virtuales sería la Plataforma de la Universidad de Vigo, FAITIC Figura 2-2.

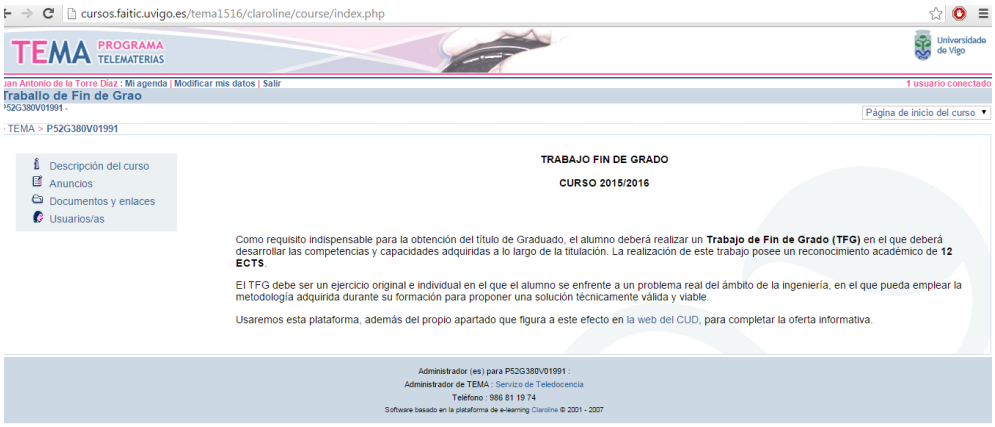

Figura 2-2 Aula virtual. FAITIC [18].

2.1.2. Recursos audiovisuales.

Otro medio de enseñanza son los recursos audiovisuales. Gracias a esto se pretende que los usuarios que los utilicen mediante los sentidos de la vista y el oído aumenten su capacidad de síntesis.

El más común y utilizado es YouTube. Gracias a esta herramienta de streaming, una persona de cualquier parte del mundo con acceso a internet, puede ver videos de cualquier tema. Dentro de las posibilidades que ofrece, en YouTube (véase en Figura 2-3) se pueden encontrar videos de enseñanza.

Otra herramienta audiovisual utilizada hoy en día son las píldoras. Son un concepto muy parecido al de YouTube utilizado por algunas universades. En ellas los profesores se graban ya sea explicando alguna parte en concreto de la asignatura o resolviendo ciertos problemas con un grado de mayor dificultad. Gracias a esto, un alumno que no haya podido asistir a la clase o esta le haya resultado compleja puede tener otra oportunidad para la asimilación de conceptos.

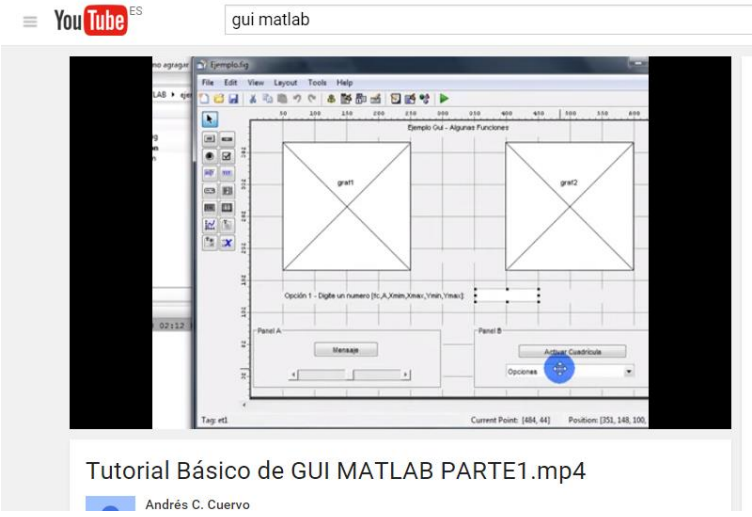

Figura 2-3 YouTube. Uso de GUI MATLAB [20]

Por último, mencionar el Power Point y Beamer (Figura 2-4). Estas dos herramientas visuales son las más utilizadas actualmente. Mediante diapositivas en las que se pueden incluir videos e imágenes los profesores muestran los contenidos de la asignatura sustituyendo en muchos casos a los libros. La tiza y la pizarra se están quedando obsoletas y gracias a este tipo de herramientas se aumenta la atención por parte de los alumnos.

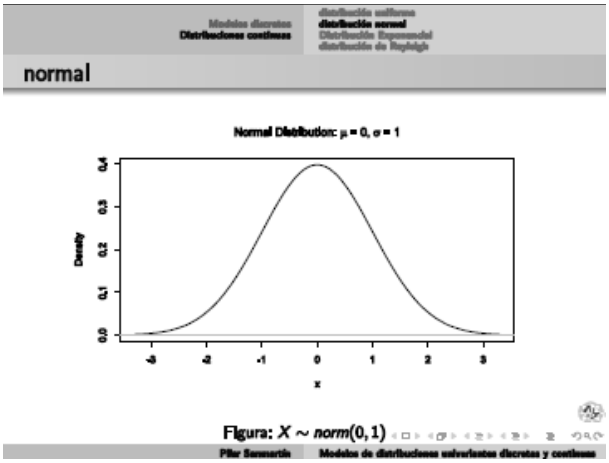

Figura 2-4 Presentación Beamer [7]

2.2 La tecnología y las matemáticas

2.2.1. Inserción de la tecnología en las matemáticas

Hoy en día, ya hay muchos trabajos acerca de la introducción de la tecnología como medio para la docencia, y no todo el mundo coincide con sus opiniones. Así Michèle Artigue, doctora y referente en educación matemática, advirtiéndonos que no son la panacea ni la solución a la complejidad e infinidad de problemáticas que conlleva el aprendizaje de la matemática, opina que [1]:

"Ciertamente estas tecnologías son socialmente y científicamente legítimas, pero a nivel de la escuela, esas legitimidades no son suficientes para asegurar la integración. Pues no se busca que la enseñanza forme alumnos aptos para funcionar matemáticamente con esas herramientas –lo que sería el caso por ejemplo de una formación de carácter profesional–: se busca mucho más. Efectivamente, lo que se espera de esas herramientas esencialmente es que permitan aprender más rápidamente, mejor, de manera más motivante, una matemática cuyos valores son pensados independientemente de esas herramientas. Lo que se necesita entonces es asegurar la legitimidad pedagógica de ellas, y eso es bien distante de asegurar su legitimidad científica o social. Esto, como hemos mostrado, genera un círculo vicioso que enferma la formación en un esquema de militancia y proselitismo, poco adecuado para otorgar herramientas a los docentes que les permitan hacer frente a las dificultades que inevitablemente van a encontrar, que les permitan identificar las necesidades matemáticas y técnicas de las génesis instrumentales y de responderlas eficazmente; poco adecuado también para permitirles la necesaria superación de una visión ingenua de la tecnología como remedio a las dificultades de la enseñanza."

Se debe tener en cuenta que solamente cuando se está seguro sobre el propósito de plantear un problema, puede decidirse claramente qué tecnología (mental, papel y lápiz, electrónica, etc.) se usará.

Lo que cambia con la tecnología son los distintos problemas que esta puede presentar. Si las clases han sido correctamente programadas o planedas, el aprendizaje puede ser muy enorme. Las tecnologías permiten al alumno acceder a una mayor variedad de problemas, así como de visualizar las matemáticas desde otro punto de vista del que no serían capaces sin ellas. Sin embargo si esto no es así el resultado podría ser desastroso. El uso de un programa difícil de interpretar por parte del alumno, o una clase incorrectamente planificada podría tener como el resultado la falta de interés por parte del alumno y como consecuencia un aprendizaje nulo de la materia.

2.2.2. Integrar recursos TIC para el estudio de la estadística

Las TIC son aquellos medios tecnológicos informáticos y de telecomunicaciones orientados a favorecer los procesos de información y comunicación, que aplicados a la enseñanza han contribuido a facilitar procesos de creación de contenidos multimedia, nuevos escenarios de apertura y entornos colaborativos. Integrar recursos TIC significa utilizar las herramientas y la información que nos ofrece la red en las actividades diarias de la clase para conseguir los objetivos del currículum y proporcionar oportunidades de aprendizaje a los alumnos [1]. El objetivo de las tecnologías es utilizarlas de modo que potencien la propuesta docente utilizándolas y aplicándolas de tal manera que faciliten y favorezcan a su vez la construcción de conocimiento por parte de los alumnos.

Hoy en día existen varios recursos TIC disponibles para el estudio de la estadística, desde los específicos de esta rama de la matemática tales como SimStat, WinIDAMS (herramienta de análisis estadístico de datos desarrollado por la UNESCO), StadiS, InfoStat y otros, hasta los más comunes en los ordenadores, tablets y teléfonos moviles que poseen los alumnos tales como las hojas de cálculo de Excel.

La introducción de estas nuevas técnicas en la planificación del estudio esta rama de las matemáticas no solo pretende la resolución de ejercicios mecánicos, sino también proporcionar al alumno distintas formas de obtener así como de compartir los resultados obtenidos. Otros ejemplos del desarrollo de estas tecnologías son los videos extraídos de YouTube o construidos con MovieMaker y presentaciones creadas por Power Point, Prezi o Beamer, entre otras.

Otro tipo de actividad que puede incluirse para introducir el tema es la caza del tesoro en la que se le da al alumno una serie de preguntas y una lista de páginas web en las que buscan las respuestas. En ella, escogiendo adecuadamente preguntas que definan las dimensiones fundamentales de un tema, los alumnos no sólo averiguan respuestas concretas, sino que profundizan en sus aspectos esenciales estimulando la adquisición de destrezas sobre tecnología de la información y comunicaciones, conocimientos prácticos sobre Internet, la web y la navegación por información online. Esto se puede realizar a través de distintos buscadores en internet. Daría como resultado una búsqueda profunda como la que se realiza en las bibliotecas, y contrastar diversas fuentes.

2.3. Herramientas informáticas para la docencia de estadística

2.3.1. SimStat

Simstat es un software estadístico de libre distribución fácil de utilizar dado que ofrece una amplia gama de estadísticas, datos intuitivos y funciones de administración de resultados así como su propio lenguaje de comandos para automatizar el análisis estadístico y escribir pequeñas aplicaciones, tutoriales interactivos, sistemas de entrevista asistida por computadora [8].

SimStat trabaja a su vez con datos numéricos, fechas y pequeñas variables alfanuméricas, con memorias y variables de los documentos que le permiten guardar en el mismo archivo de proyecto los problemas a resolver, gráficas obtenidas o resutados adquiridos (véase en Figura 2-5 y Figura 2-6). Permite al usuario una mejor comprensión de la estadística mediante las distintas herramientas que dispone. Este software es utilizado en Universidades españolas importantes tales como la de Universidad de Alcalá o la Universidad Complutense de Madrid.

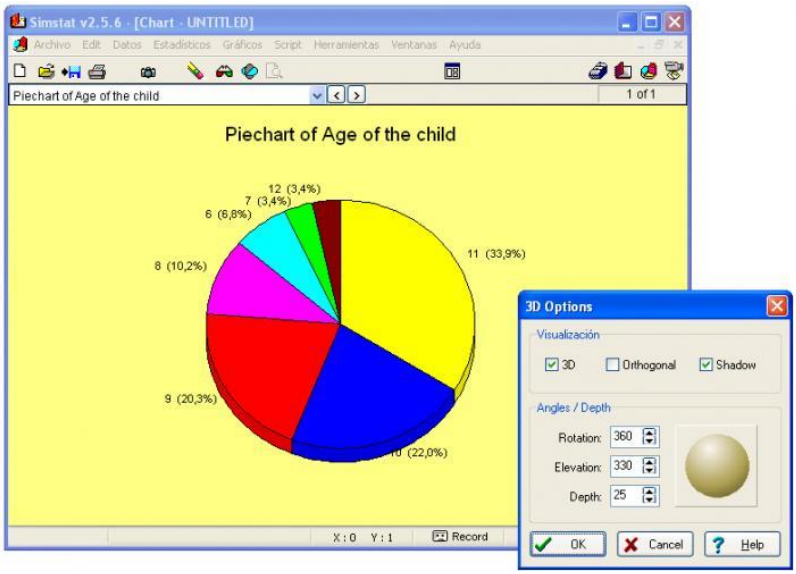

Figura 2-5 Ejemplo Porcentajes SimStat [8]

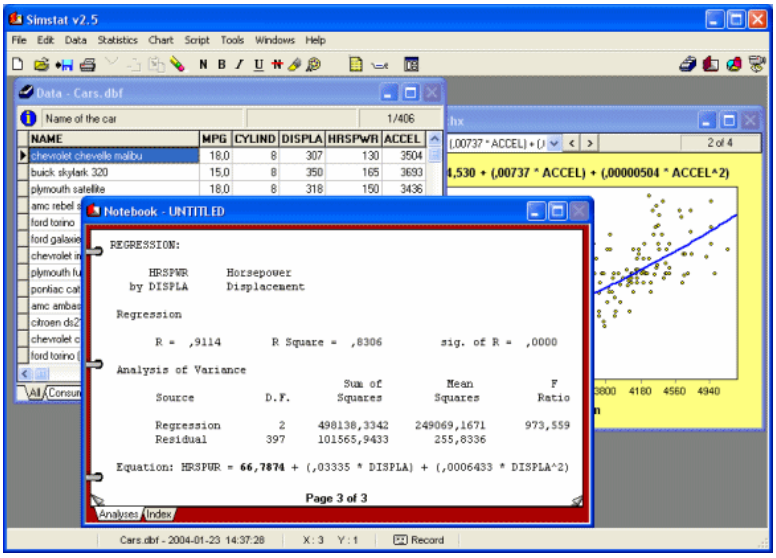

Figura 2-6 Calculo Regresión. SimStat [8]

2.3.2. WinIDAMS

IDAMS es un software destinado a la validación, la manipulación y el análisis estadístico de datos para la plataforma Windows. En él se pueden analizar diversos datos a través de una interfaz de usuario de lenguaje sencillo. Constituye una herramienta utilizada tanto para el ámbito de la investigación como para el educacional o administrativo (Figura 2-7). A su vez se puede obtener de forma gratuita a través del Secretariado de Unesco y de distribuidores oficiales [9].

En un principio, IDAMS proviene en parte del paquete estadístico OSIRIS III.2 desarrollado al comienzo de los años setenta en el Instituto para la Investigación Social de la Universidad de Michigan, Estados Unidos de América. Después, fue modificado notoriamente, imponiendo nuevas mejoras, y hoy en día el Secretariado de la UNESCO lo mantiene altualizado gracias a expertos de distintos países en el ámbito de las matemáticas. Sus siglas significan *Internationally Developed Data Analysis and Management Software Package*: en español "Paquete de programas para el análisis y

manejo de datos desarrollado internacionalmente", debido a los diferentes expertos de distintos lugares que han formado parte en sus mejoras y creación.

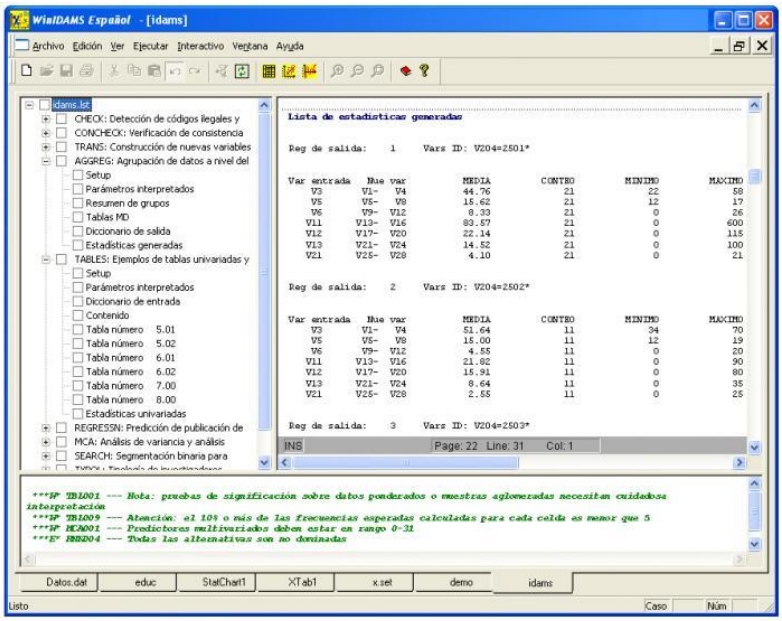

Figura 2-7 Ejemplo estadísticas. WinIDAMS [9]

2.3.3. StaDis

Stadis es un programa informático que por medio de una calculadora estadística (Figura 2-8) permite, tanto a alumnos como aficionados realizar distintos cálculos y comprender mejor esta rama de las matemáticas [10].

StaDis permite:

- Obtener cálculos estadísticos como media aritmética, media geométrica, desviación típica, varianza, máximo, mínimo, etc.
- Analizar datos mediante la creación de tablas de distribución.
- Guardar los enunciados de problemas, nuestras anotaciones o, incluso, resúmenes de los cálculos realizados (análisis bivariante, medidas de dispersión, medidas de asimetría o medidas de tendencia central).
- Generación de distintos tipos de gráficas para representar los resultados (diagramas de barras o circulares).

Destacar de este software que es el más sencillo de todos los descritos, y permite una comprensión de ciertas partes de la estadística para aquellos que sean principiantes.

Figura 2-8 Calculadora StaDis [10].

2.3.4. InfoStat

InfoStat es un software para análisis estadístico de aplicación general desarrollado bajo la plataforma Windows, que ofrece una interfaz avanzada para el manejo de datos [11].

Por su origen es universitario, más concretamente de la Universidad Argentina de Cordoba. El programa cuenta con bastantes facilidades a la hora de aplicarlo para la enseñanza de la estadística, lo que lo hace especialmente útil para el desarrollo de cursos de grado y posgrado. La liberación de InfoStat estudiantil facilita el trabajo tanto a docentes como alumnos quienes pueden organizar sus actividades de enseñanza-aprendizaje en un marco de libertad y legalidad.

Tiene la capacidad de obtener estadísticas descriptivas y gráficos (Figura 2-9) para su análisis en el ámbito de la investigación. Además cuenta con métodos avanzados de modelización estadística y análisis multivariado [11]. Una de sus principales características es su habilidad de conectarse con el programa infórmatico R (descrito posteriormente), además de permitir importar y exportar bases de datos en formato texto, Excel y Epiinfo. Esto permite a la aplicación el estar bastante actualizada y contar con diversas capacidades a la hora de la obtención de cálculos.

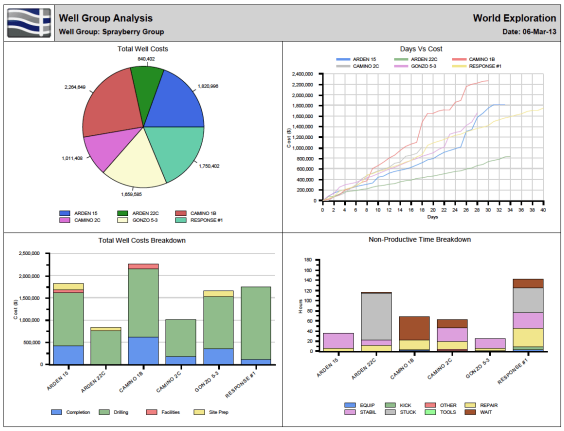

Figura 2-9 Analisis realizado por InfoStat [11].

2.3.5. Programa R

El programa R consiste en un lenguaje y entorno para el análisis de datos y gráficos que en la actualidad se utiliza no solamente en el ámbito de la enseñanza, sino también dentro de la investigación [21]. Fue desarrollado inicialmente por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993 y actualmente se usa en el campo de la investigación biomédica, la bioinformática y las matemáticas financieras. Es un software de uso libre.

Entre sus principales ventajas destaca que es de libre distribución para los alumnos. Funciona además tanto en entorno UNIX como en Windows y MacOS. Abarca el análisis de todo el temario utilizado en las asignaturas de Estadísticas de la Universidad, y está en continua mejora y actualización.

A continuación pueden observar ejemplos del uso del programa R. En la figura Figura 2-10 aparecen una serie simulada con R y en la Figura 2-11 se representan las funciones de autocorrelación simple y parcial muestrales respectivamente.

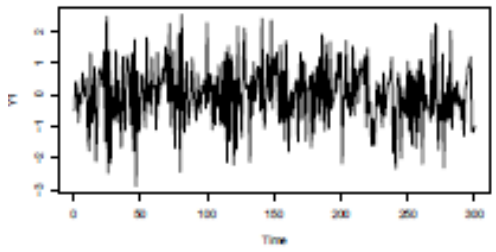

Figura 2-10 Serie simulada con R [7].

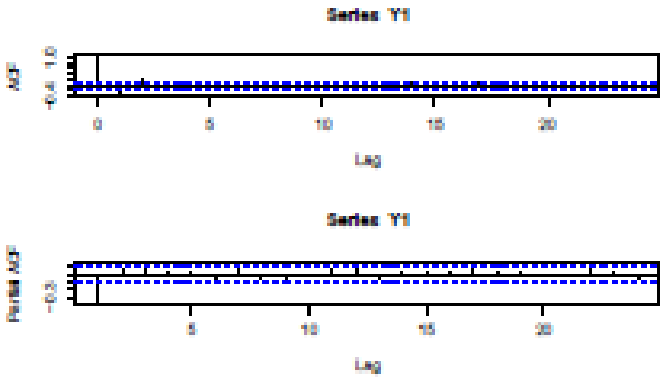

Figura 2-11 Programa R. Funciones de autocorrelación simple y parcial muestrales [7].

2.3.6. Octave

Octave o GNU[1] Octave es un software libre utilizado para cálculos numéricos. Nació como parte del proyecto GNU, y es considerado equivalente a MATLAB. Entre sus características resaltar la cantidad de paquetes que dispone para implementar distintas funcionalidades (Figura 2-12). Esta orientado hacia el análisis número [15].

[1] *GNU's Not Unix*

Su creación data de 1988. Nació con la intención de ser utilizado en un curso de reactores químicos. A partir de 1992 se decidió extenderlo. El matemático John W. Eaton se encargó de ello lanzando en 1994 la primera versión en la cual se podía realizar un amplio estudio de las distintas disciplinas de las matemáticas. El nombre surge de Octave Levenspiel, profesor de uno de los autores y conocido por sus buenas aproximaciones, por medio de cálculos elementales, a problemas numéricos en ingeniería química.

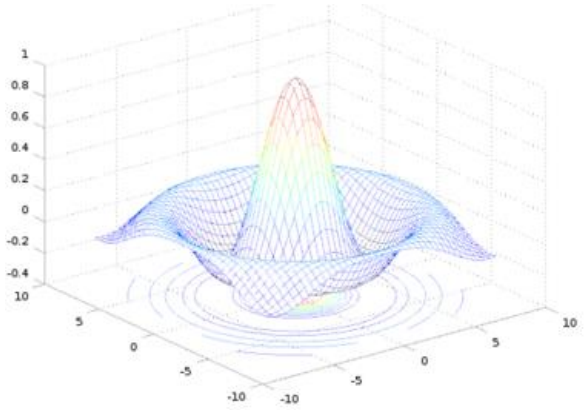

Figura 2-12 Grafica OCTAVE [15]

2.3.7. Matlab®

Por último, mencionar como software informático para el estudio de la estadística, así como de las matemáticas, el Matlab ® (laboratorio de matrices). Fue diseñado por el matemático y programador americano especialista en análisis numérico Cleve Barry Moler en el año 1984. Utiliza un lenguaje de programación propio "lenguaje M" y que fue creado en 1970 para proporcionar mayor sencillez al software de matrices sin tener que usar Fortran. MATLAB (Figura 2-13) permite, entre otras utilidades, el manejo de matrices, representar tanto datos como funciones, el desarrollo de interfaces de usuario también conocidas como GUI, (*Graphical User Interface*) y la comunicación con otros lenguajes de programación y dispositivos software [3].

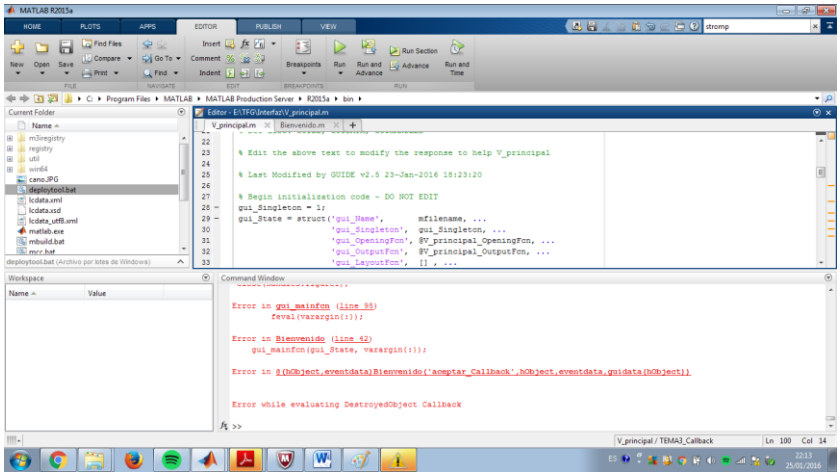

Figura 2-13 MATLAB

Dentro del Matlab se cuenta con la posibilidad de crear interfaces gráficos con el GUI. Gracias a esta herramienta, se obtendrá un mayor control sobre el diseño y el desarrollo (Figura 2-14), además se puede crear código de MATLAB que defina las propiedades y los comportamientos de todos los componentes. Utilizando el GUI cabe la posibilidad de agregar cuadros de diálogo, controles de interfaz de usuario (como botones y controles deslizantes) y contenedores (como paneles y grupos de botones), que ayudaran a hacer que el aprendizaje sea más interactivo y se puedan visualizar resultados de los diferentes ejercicios que se vayan realizando.

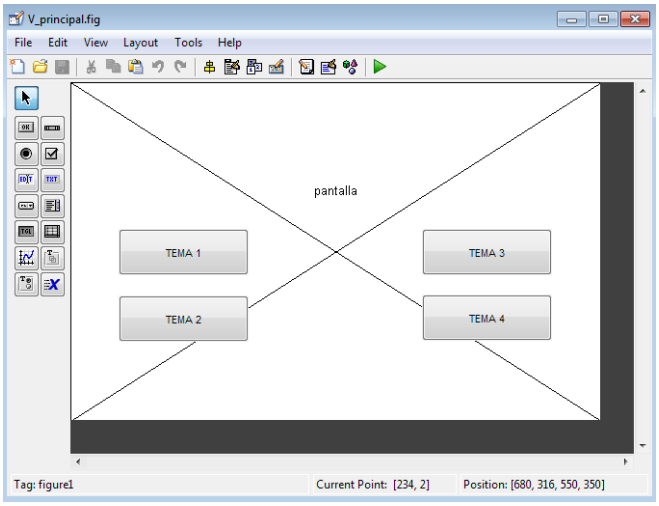

Figura 2-14 Creación GUI

Los orígenes de los interfaces de usuario, vienen de los investigadores del Stanford Research Institute liderados por Douglas Engelbart (Universidad de Berkeley), que desarrollaron en 1973 el Xerox Alto, el primer ordenador personal con una interfaz de hipervínculos en modo texto gobernado por un raton, que también inventaron (el primer prototipo en madera). Este concepto fue ampliado y trasladado al entorno gráfico por los investigadores del Xerox PARC (Palo Alto Research Center), en él se definieron los conceptos de ventanas, *checkbox*, botones de radio, menús y puntero del raton. Fue implementado comercialmente en el Xerox Star 8010 en 1981 Hoy en día, se tiene como ejemplo de GUIs los entornos de escritorio de los sistemas operativos: Windows, Mac Os, X – Windows (Linux), etc [3].

3. DESARROLLO DE LA APLICACIÓN

3.1. Introducción

Tras haber realizado un estudio acerca de todas las posibilidades de programas informáticos para realizar el trabajo, se ha deducido que el más indicado es el Matlab. SimStat, WinIDAMS, StaDis e InfoStat, son softwares de estadística descriptiva los cuales no abarcarían todo el temario correspondiente. Cualquiera de los tres programas restantes (R, Octave, Matlab®) son aptos para realizar el trabajo propuesto, pero se ha optado por el último por dos motivos:

- Disponibilidad de la última versión del mismo y todas sus *toolboxes*, al ser alumno de la Universidad de Vigo.
- Familiaridad con el entorno de programación, debido a su uso en laboratorios de asignaturas cursadas anteriormente.

Una vez se ha optado por el MATLAB como entorno de programación para desarrollar la aplicación, el siguiente paso es el desarrollo de la misma. En la Figura 3-1 se presenta un diagrama de flujo de la aplicación desarrollada, y en los siguientes apartados se presenta la realización de las distintas partes de la misma.

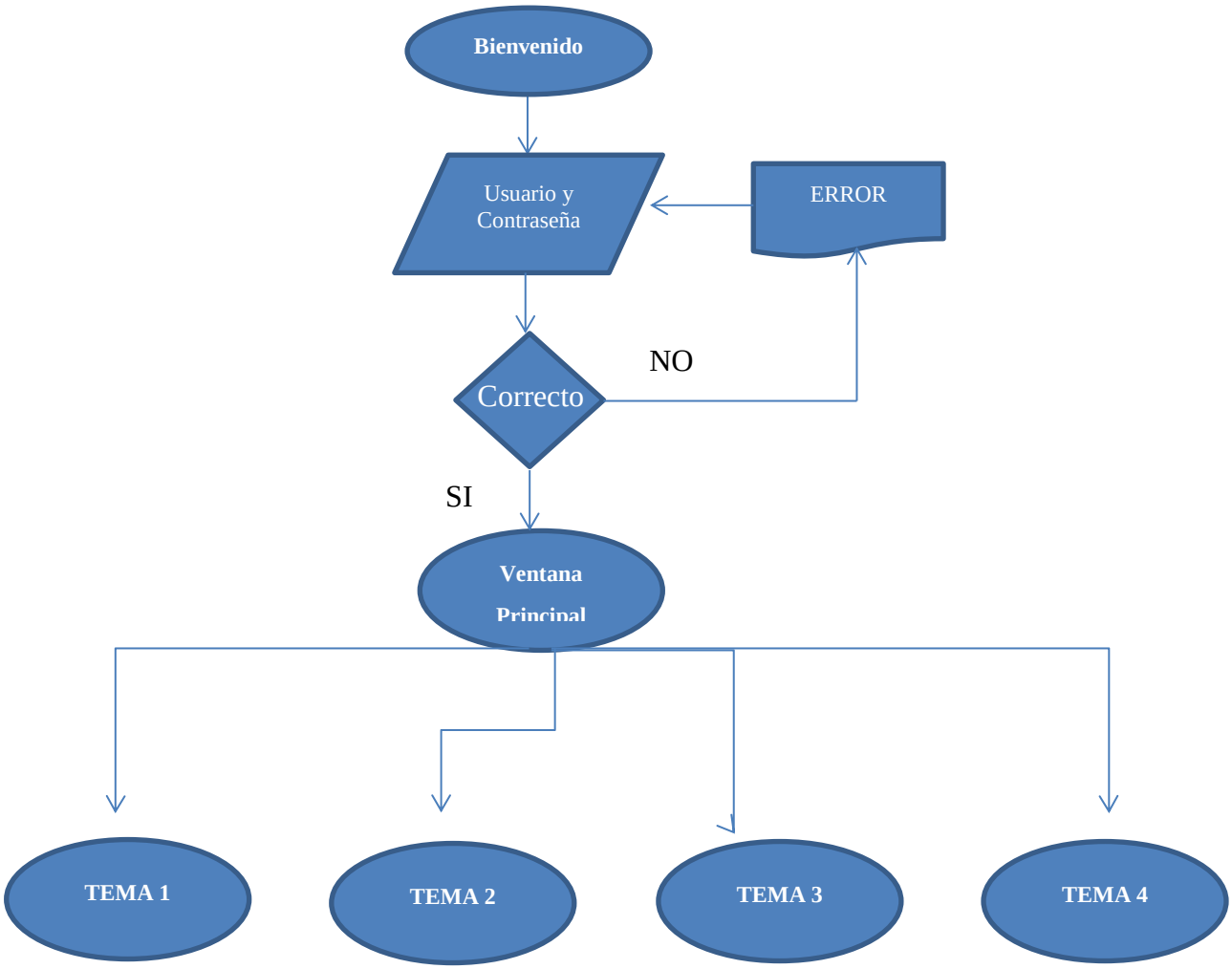

Figura 3-1 Diagrama de Flujo Aplicación

3.2. Bienvenido

La primera ventana (Figura 3-2) de la interfaz de Usuario creada que el usuario encontrará al abrir el archivo.m, será la ventana de verificación.

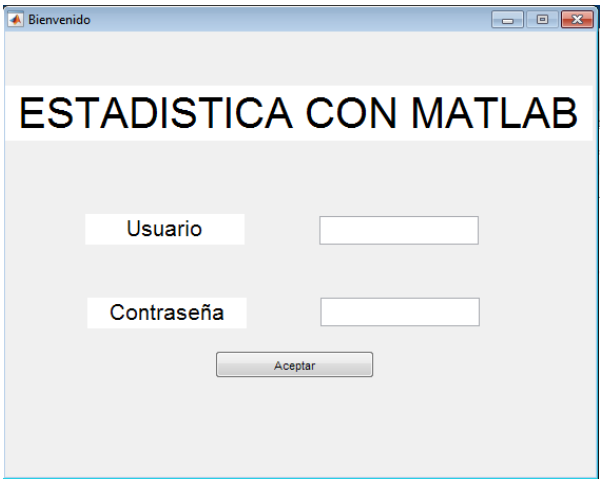

Figura 3-2 Ventana de verificación

En ella el usuario podrá encontrar dos cajas en las que introducir usuario y contraseña y un botón el cual confirmará si tanto la contraseña como el usuario son correctos. Estas dos cajas son dos *static text* y el botón un *push button*. Para su computación y el acceso a diferentes ventanas se hizo una comprobación del texto escrito en los *static text* (Figura 3-3). En el caso de que el usuario o la contraseña sean erróneos se mostrará un mensaje de error y no permitirá acceder al resto de ventanas.

El usuario y contraseña para acceder son "alumno" y "escuelanaval" respectivamente. En el caso de que se quiera cambiar el usuario o contraseña habría que acceder al código en el archivo .m y en la línea de usuario y clave cambiarlas (veasé el código en Anexo 1).

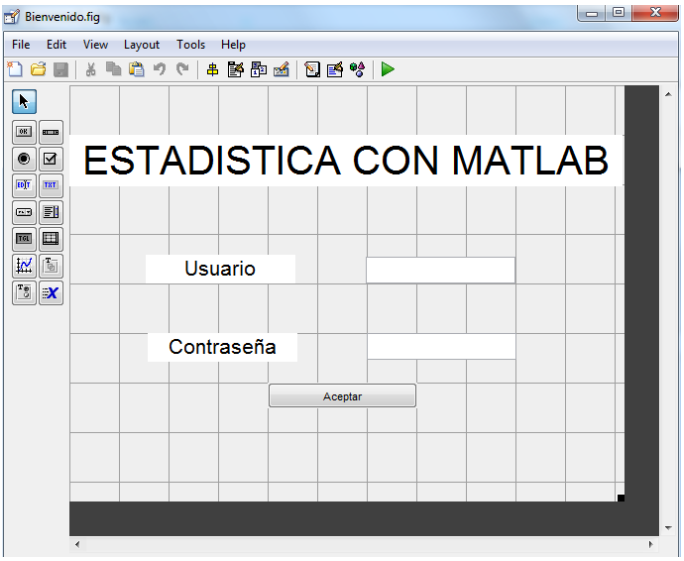

Figura 3-3 Creación Ventana Contraseña

3.3. Menú Principal

En el menú principal se han situado cuatro botones. Cada uno de ellos tiene el nombre y número del tema al que se puede acceder mediante su pulsación (Figura 3-4).

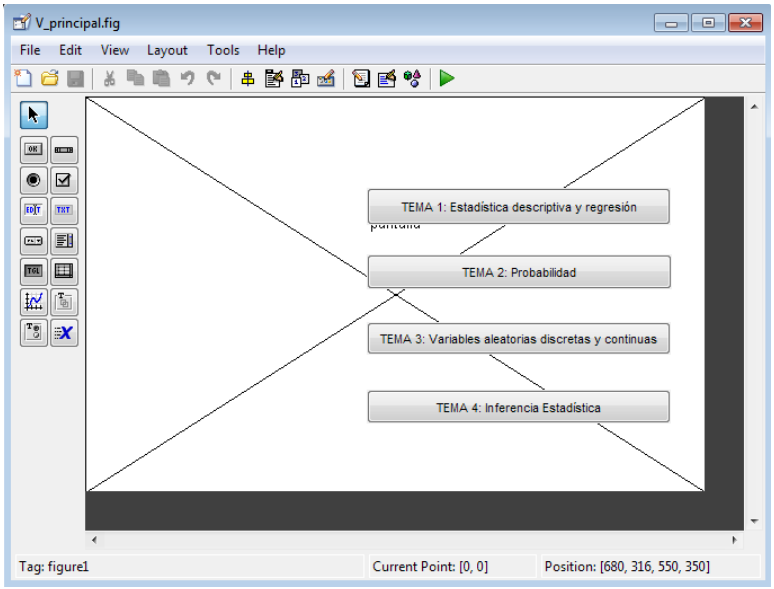

Figura 3-4 Ventana menú principal

La ventana está compuesta de manera muy simple, únicamente por cuatro *pushbuttons* los cuales mediante su pulsación dan acceso al temario de la asignatura de Estadística. El número y nombre del tema siguen la guía docente y los apuntes de la asignatura de Estadística de la Escuela Naval Militar correspondiente al curso 2015-2016 (véase el código en Anexo 2).

3.4. Tema 1. Estadística descriptiva y regresión.

La estadística descriptiva es la rama de las Matemáticas que organiza, recolecta, muestra y caracteriza un conjunto de datos (por ejemplo, edad de una población, altura de los estudiantes de una escuela, temperatura en los meses de verano, etc.) con el fin de describir apropiadamente las diversas características de ese conjunto. Trata de extraer conclusiones sobre los comportamientos de las variables, ya sean cuantitativas o cualitativas. La regresión trata de calcular la dependencia entre dos variables aleatorias.

Es por ello por lo que el tema se ha dividido tal como muestra el diagrama de la Figura 3-12:

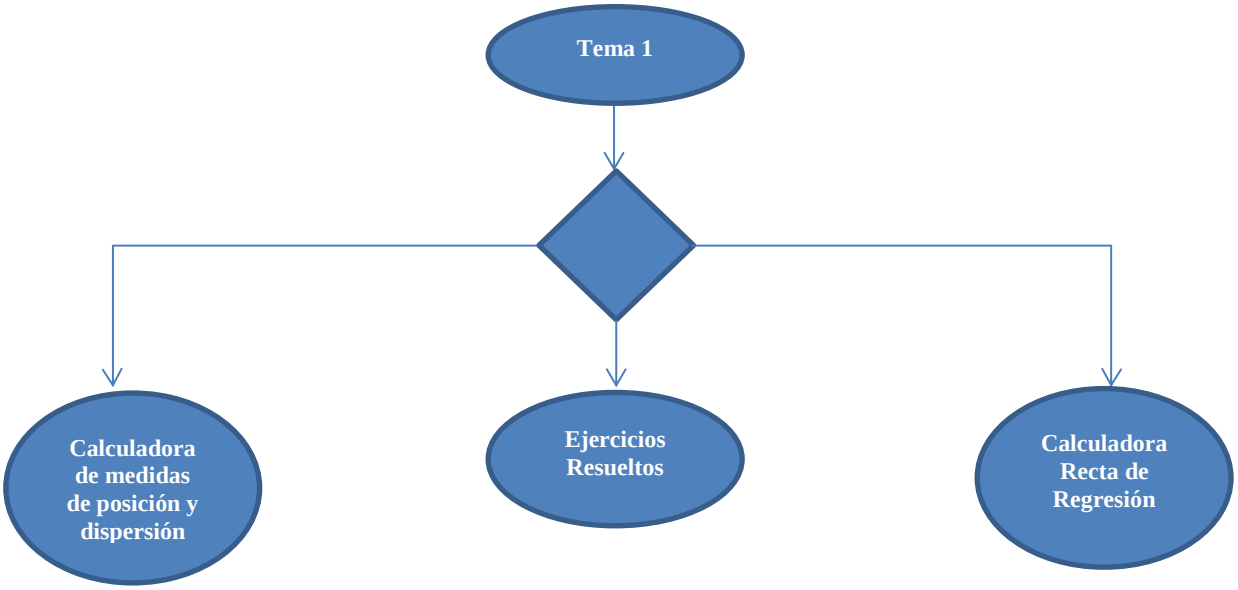

Figura 3-5 Diagrama de Flujo. Tema 1

Una vez pulsado el botón del primer tema en el menu principal, se tendrá acceso a la siguiente ventana compuesta por tres *pushbuttons* y *Pop-up menu* donde se encuentran diferentes opciones para trabajar los contenidos del tema. Son las siguientes:

- Ejercicios resueltos
- Calculadora de recta de regresión lineal
- Calculadora de Medidas de Posición, Dispersión y Forma.

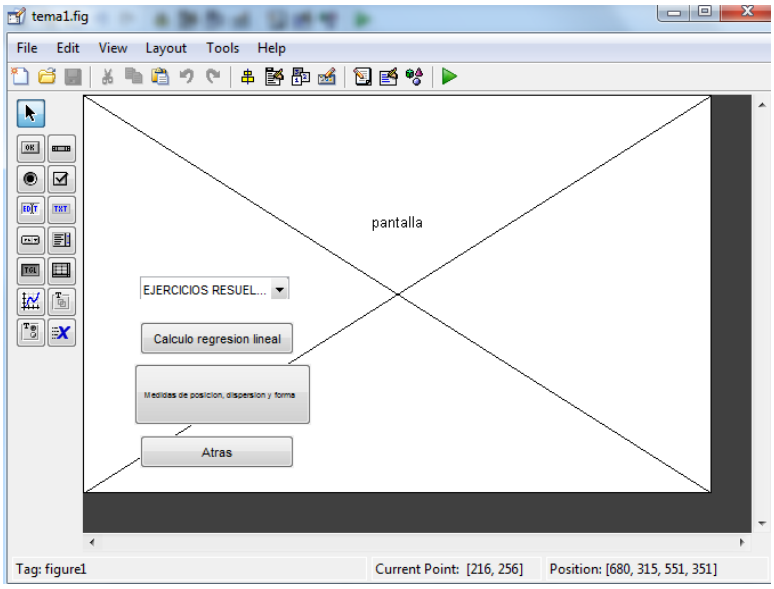

Figura 3-6 Pantalla Pricipal. Tema 1

Como ya se ha comentado, dicha venta esta compuesta por tres *push buttons* (Figura 3-6), dos para acceder al cálculo de regresión y medidas, y un tercero para volver atrás a la selección del tema. Además cuenta con un *Pop up Menu* que nos dará las opciones de los ejercicios a seleccionar (Figura 3-7).

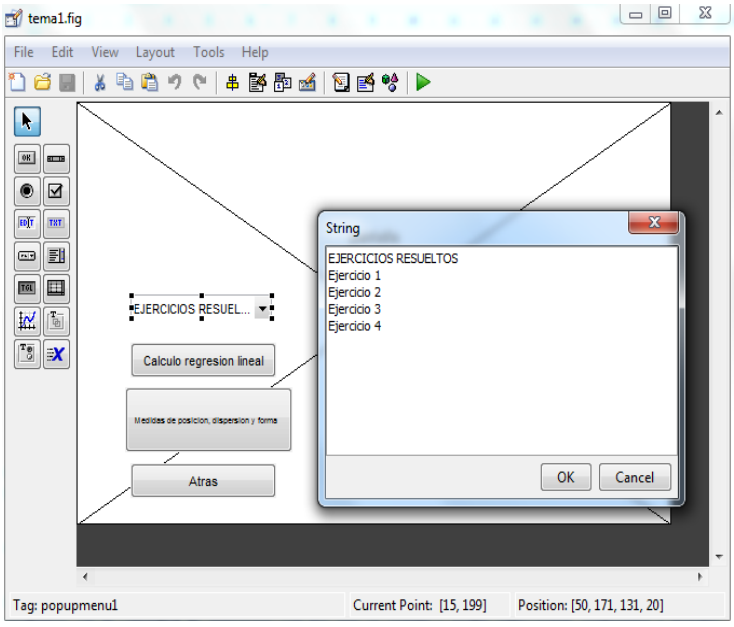

Figura 3-7 Ventana Tema 1. Creación Pop-up Menú

Para ello se han configurado las distintas opciones del *Pop-up Menú*, obteniendo el mismo resultado como si de un *pushbutton* se tratara al proceder a su pulsación. De este modo se puede comprobar la variedad de ejercicios con que se cuenta y poder seleccionar el deseado (véase código en Anexo 3).

3.4.1. Recta de Regresión

Las rectas de regresión lineal son utilizadas en estadística para aproximar la relación de dependencia entre dos variables aleatorias X e Y. En otras palabras, establece la ecuación de una recta que más se aproxime a la nube de puntos de los que se dispone.

La representación más útil de las dos variables continuas sin agrupar es el diagrama de dispersión. En un eje de coordenadas se representa la nube de puntos la cual refleja la posible relación entre las variables (x; y). Cuanto más estrecha sea la nube de puntos más relación habrá entre ellos y cuanto más dispersa menos. La más sencilla de estas relaciones es la lineal, viene dada por la Ecuación 3-1. Este modelo supone que una vez calculados los parámetros a y b, se podrían calcular los valores de X e Y [13].
Para el cálculo de los parámetros a y b se utiliza la Ecuación 3-1 y la Ecuación 3-3 respectivamente.

$$y = \mathrm{ax} + \mathrm{b}$$

Ecuación 3-1 Ecuación lineal

$$a = \frac{\frac{1}{\mathrm{n}} * \sum(x_i yi) - \sum x_i \sum y_i}{\frac{1}{\mathrm{n}} * \sum x_i^2 - (\sum x_i)^2}$$

Ecuación 3-2 Parametro "a" Recta de regresión

$$b = \frac{\sum yi - a \sum x_i}{\mathrm{n}}$$

Ecuación 3-3 Parametro "b" Recta de regresión

Dentro de la interfaz gráfica, en la ventana de "regresión" se tiene la posibilidad de calcular la ecuación de la recta dependiendo de los valores seleccionados. En ella se encuentra una calculadora en la cual se pueden introducir los datos en una tabla, ya sea uno a uno o todos a la vez (véase en la Figura 3-8). Los datos se quedarán grabados en la memoria mediante la pulsación de los *pushbuttons* "introducir datos" y borrados mediante el botón "vaciar". Esta calculadora tiene la posibilidad de representar en una gráfica la nube de puntos y la recta correspondiente. Para ello se calcula dentro de la interfaz gráfica que leería la ecuación de la recta calculada y saldría su solución en el *edit text* (véase el código en Anexo 4).

No se puede calcular si los datos no se han introducido antes en la tabla.

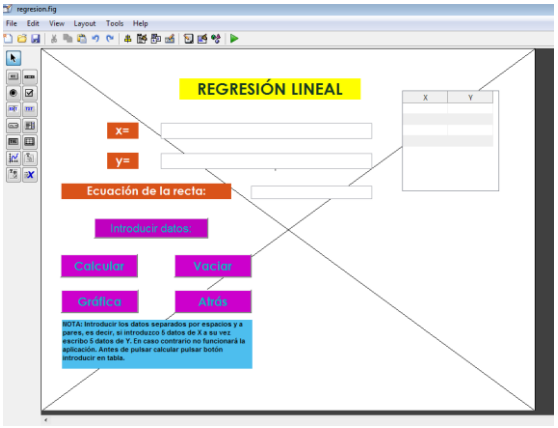

Figura 3-8 Ventana Cálculo de Recta de Regresión. Tema 1

3.4.2. Ejercicios Resueltos

En el siguiente apartado, la interfaz creada proporciona una serie de Ejercicios Resueltos (Figura 3-9). Tras haber sido desplegado el menú de los distintos ejercicios que se pueden encontrar, se selecciona uno de ellos.

En la primera ventana se visualiza el enunciado del ejercicio y una serie de botones y cajas. Se ha programado de tal manera que el usuario pueda comprobar si el resultado obtenido a la hora de hacer el ejercicio es correcto o tiene algún error. El botón de ejercicio resuelto permanecerá inactivo hasta que el usuario inserte un resultado en la interfaz. Hay gran variedad de ejercicios y diferentes normas en ellos. En algunos ejercicios se ha configurado de tal manera que el usuario intente dar con la solución y tras haberlo intentado varias veces sin haberlo logrado tenga acceso a esta. Otros por ejemplo exigen tener al menos dos o tres apartados bien. De esta manera lo que se busca es la participación y esfuerzo por parte del usuario y que no sea un simple pasatiempo.

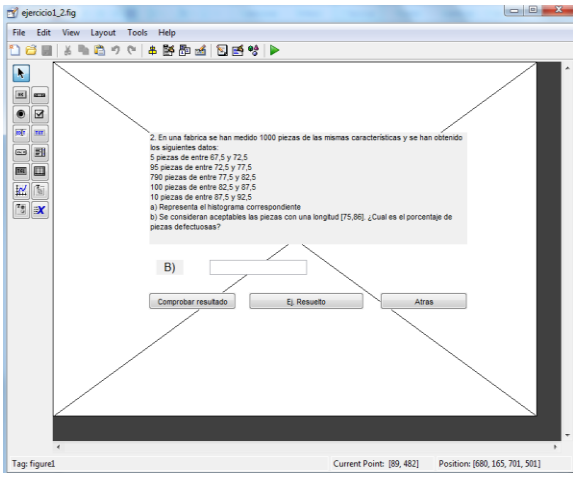

Figura 3-9 Ejercicio Resuelto. Tema 1

Al pulsar el botón de ejercicio resuelto, se accedería a nuevas ventanas donde poder ver la solución (Figura 3-10). Además, algunos cuentan con histogramas para visualizar de forma gráfica los resultados obtenidos (véase código en Anexo 5).

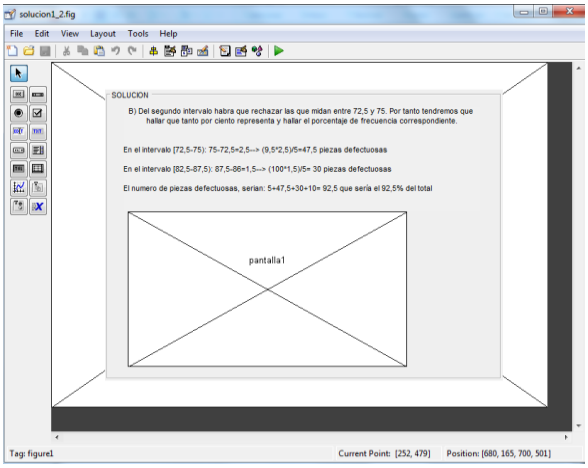

Figura 3-10 Ejercicio Resuelto

3.4.3. Medidas de dispersión y posición

Para calcular las medidas de posición y forma se introducen una serie de medidas descriptivas construidas a partir de datos muestrales, de manera que cada una de ellas muestra las características de estos. Estas medidas descriptivas se agrupan en posición y dispersión [13].

Las medidas de posición como su propio nombre indica, muestran los valores que determinan la posición de un conjunto de datos. Mientras que las medidas de dispersión tratan de medir la dispersión o concentración de las observaciones muestrales. Las más populares son aquellas que cuantifican la dispersión o variación con respecto a un valor central.

En este apartado de la interfaz, se ha creado una calculadora al igual que en la Recta de Regresión, en la cual una vez se introducen datos en la tabla, esta los guarda en la memoria y tendran la posibilidad de calcular las respectivas medidas (Figura 3-11). A su vez, esta calculadora cuenta con la opción de consultar el histograma, en el cual se pueden visualizar los datos dependiendo de sus frecuencias.

Tras la pulsación de cada uno de los *pushbuttons* haría la orden establecida, introduciendo resultados ya sean en la tabla, en cada una de las medidas de posición y dispersión o en su respectivo histograma. No se pueden calcular valores sin antes introducirlos en las tablas.

Las medidas de posición y dispersión que se calculan son las siguientes:

- **Moda**: Valor que más se repite.

- **Media aritmética**: Cantidad total de la variable distribuida a partes iguales entre cada observación (Ecuación 3-4).

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} a_i$$

Ecuación 3-4 Media Aritmética

- **Media geométrica**: Raiz n-enésima del producto de toda la cantidad de números arbitrarios tomados (Ecuación 3-5)

$$\bar{x} = \sqrt[n]{\prod_{i=1}^{n} x_i}$$

Ecuación 3-5 Media Geométrica

- **Media armónica**: Dentro de una cantidad finita de números tomados es igual al recíproco de la media aritmética de dichos valores (Ecuación 3-6)

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

Ecuación 3-6 Media Armónica

- **Media cuadrática:** Medida estadística de la magnitud de una cantidad variable. (Ecuación 3-7)

$$x_{RMS} = \frac{1}{n}\sum_{i=1}^{n} x^2$$

Ecuación 3-7 Media Cuadrática

- **Varianza**: Definida como la esperanza del cuadrado de la desviación de dicha variable respecto a su media (Ecuación 3-8)

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

Ecuación 3-8 Varianza

- **Desviación**: medida de dispersión para variables de razón (variables cuantitativas o cantidades racionales) y de intervalo. Se define como la raíz cuadrada de la varianza de la variable.

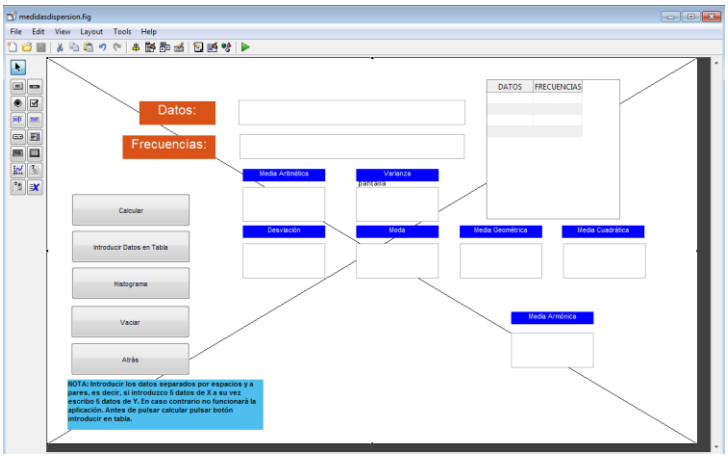

Figura 3-11 Calculadora Medidas de Posición Dispersión

Gracias a esta aplicación el usuario puede resolver problemas que tenga en la asignatura y calcular de una manera más rápida y precisa sin tener que utilizar su propia calculadora (véase código en Anexo 6).

3.5. Tema 2. Probabilidad

La probabilidad nace como un método para obtener la frecuencia de un determinado suceso en ocurrir, bajo conocimiento de todos los resultados posibles bajo condiciones estables. Expuesto de modo simple existen tres tipos de probabilidad: probable, improbable o seguro.

Se dice un suceso es seguro cuando tiene una probabilidad dee 100%, como por ejemplo sacar cara o cruz al tirar una moneda. Un suceso es probable cuando existen posibilidades de que ocurra o que no, por ejemplo obtener un dos tras arrojar un dado, y un suceso es improbable cuando existe la pequeña posibilidad de que ocurra pero esta es mínima, como por ejemplo tirar una moneda normal y que esta al caer se quede de canto.

El tema se ha dividido en una calculadora de probablidad y sus respectivos ejercicios resueltos (Figura 3-12).

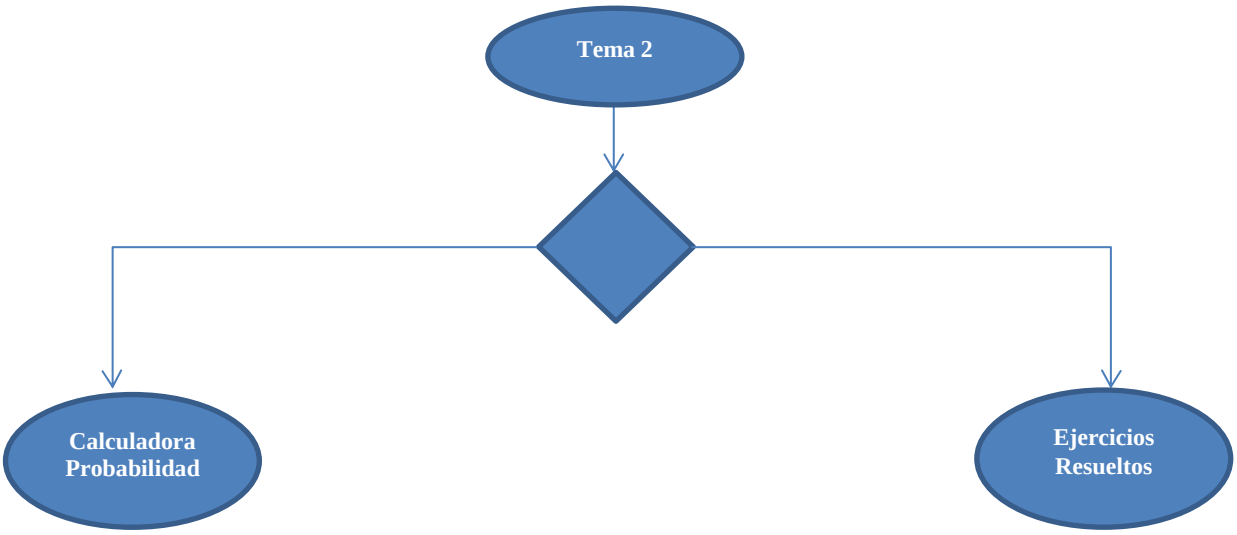

Figura 3-12 Diagrama de Flujo. Tema 2

Al pulsar el segundo tema, al igual que en el primero, se abrirá una nueva ventana teniendo acceso a sus contenidos. Esta ventana consta de dos pushbuttons que permiten acceder a la calculadora de probabilidad o volver al menú principal y un Pop-up menú donde encontraremos una nueva batería de ejercicios para que el alumno ponga en práctica sus conocimientos acerca de la probabilidad (Figura 3-13).

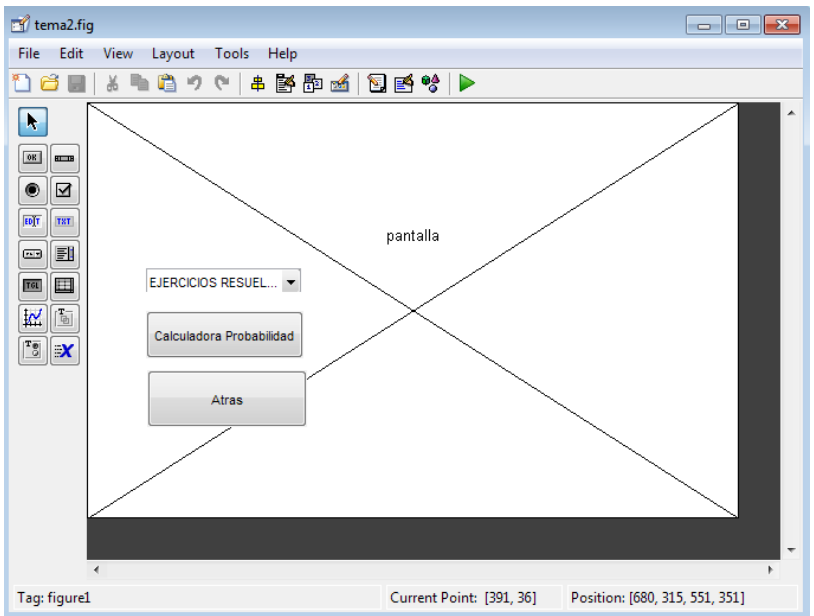

Figura 3-13 Pantalla Principal. Tema 2

3.5.1. Calculadora de Probabilidad

La idea de la calculadora de probabilidad nace tras comprobar el tiempo estimado de un alumno en hacer los cálculos para resolver los distintos ejercicios. De esta manera este tiempo se reduce con creces. La calculadora está ideada de tal forma que el usuario deba saber distinguir entre sucesos incompatibles o compatibles, y como consecuencia pueda obtener unos resultados u otros.

Para su creación se realizó un proceso similar a los anteriores. La nueva ventana cuenta con una serie de *edit text* donde se introducen las probabilidades de los sucesos A y B y donde se puede visualizar el resultado de la operación marcada. Para seleccionar cada operación se hace mediante *push buttons*. A su vez la calculadora cuenta con un formulario para que el alumno pueda saber que cálculos hace cada botón.

Dentro de la calculadora se pueden diferenciar tres *push buttons,* para sucesos incompatibles, o para sucesos compatibles dependientes e independientes.

Sucesos Compatibles:

Son aquellos que pueden suceder simultaneamente. Pueden ser dependientes o independientes.

Dependientes, aquellos en los que la probabilidad de que uno de ellos suceda depende de si el otro ha sucedido o no. Para poder trabajar con ellos se requiere no sólo conocer la probabilidad de cada uno de los mismos, sino algo más, como puede ser la probabilidad de su unión o la de su intersección. Entonces:

$$P(AUB)= P(A) + P(B) - P(A\cap B)$$

Ecuación 3-9 Probabilidad de A o B (Sucesos Compatibles Dependientes)

$$P(A\cap B)= P(A)+P(B)- P(AUB)$$

Ecuación 3-10 Probabilidad de A y B (Sucesos Compatibles Dependientes)

$$P(A/B)= \frac{P(A\cap B)}{P(B)}$$

Ecuación 3-11 Probabilidad de A condicionada a B (Sucesos Compatibles Dependientes)

$$P(A!)= 1 - P(A)$$

Ecuación 3-12 Probabilidad de que no pase A (Sucesos Compatibles Dependientes)

$$P(A\backslash B)= P(A) - P(A\cap B)$$

Ecuación 3-13 Probabilidad de A y no B (Sucesos Compatibles Dependientes)

Independientes, aquellos en los que el hecho de que uno de ellos ocurra o no, no tiene influencia en la probabilidad de ocurrencia del otro. Entonces:

$$P(AUB)= P(A) + P(B) - P(A)*P(B)$$

Ecuación 3-14 Probabilidad de A o B (Sucesos Compatibles Independientes)

$$P(A \cap B) = P(A) * P(B)$$

Ecuación 3-15 Probabilidad de A y B (Sucesos Compatibles Independientes)

$$P(A/B) = \frac{P(A \cap B)}{P(B)} = P(A)$$

Ecuación 3-16 Probabilidad de A condicionada a B (Sucesos Compatibles Independientes)

$$P(A!) = 1 - P(A)$$

Ecuación 3-17 Probabilidad de que no pase A (Sucesos Compatibles Independientes)

$$P(A \setminus B) = P(A) - P(A \cap B)$$

Ecuación 3-18 Probabilidad de A y no B (Sucesos Compatibles Independientes)

Sucesos Incompatibles:

Aquellos que no pueden suceder simultaneamente. Es decir:

$$P(AUB) = P(A) + P(B)$$

Ecuación 3-19 Probabilidad de A o B (Sucesos Incompatibles)

$$P(A \cap B) = 0$$

Ecuación 3-20 Probabilidad de A y B (Sucesos Incompatibles)

$$P(A/B) = 0$$

Ecuación 3-21 Probabilidad de A condicionada a B (Sucesos Incompatibles)

$$P(A \setminus B) = P(A)$$

Ecuación 3-22 Probabilidad de A y no B (Sucesos Incompatibles)

A continuación se puede ver en la Figura 3-14 como sería el resultado obtenido tras su creación. Para comprobar el código de cada uno de los botones consultar el Anexo 7.

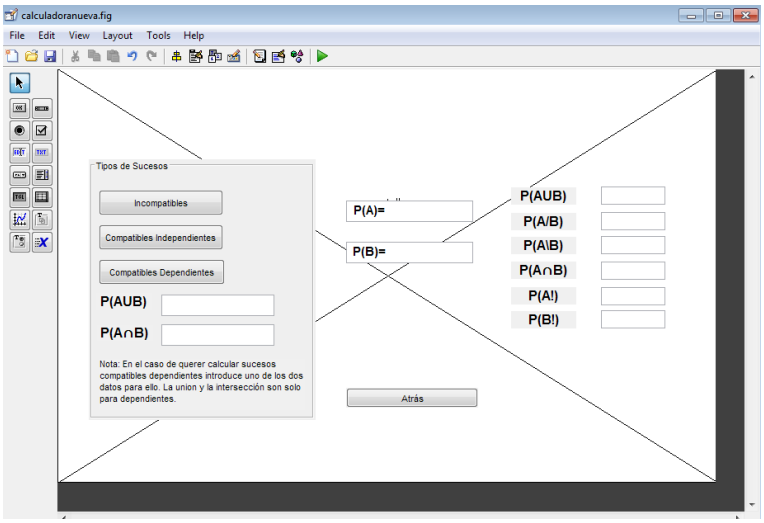

Figura 3-14 Calculadora Probabilidad. Tema 2

3.5.2. Ejercicios Resueltos

Al pulsar el menú desplegable de "Ejercicios Resueltos", como sucedía en el Tema 1, se puede acceder a la batería de ejercicios que se han proporcionado. En este caso, el acceso sería a distintos ejercicios de probabilidad computados de tal manera que el usuario pueda comprobar sus resultados y acceder a una nueva ventana donde comprobaría la resolución del ejercicio seleccionado.

3.6. Tema 3. Variables aleatorias discretas y continuas.

Dentro del tercer tema de la asignatura se estudian las variables aleatorias. Estas variables pueden ser discretas, si los valores tomados pueden ser representados por un conjunto discreto de números, como Z o N, o continuas si el conjunto de valores que puede tomar es un intervalo o conjunto de intervalos. Para ello se divide el tema de la siguiente manera (Figura 3-15):

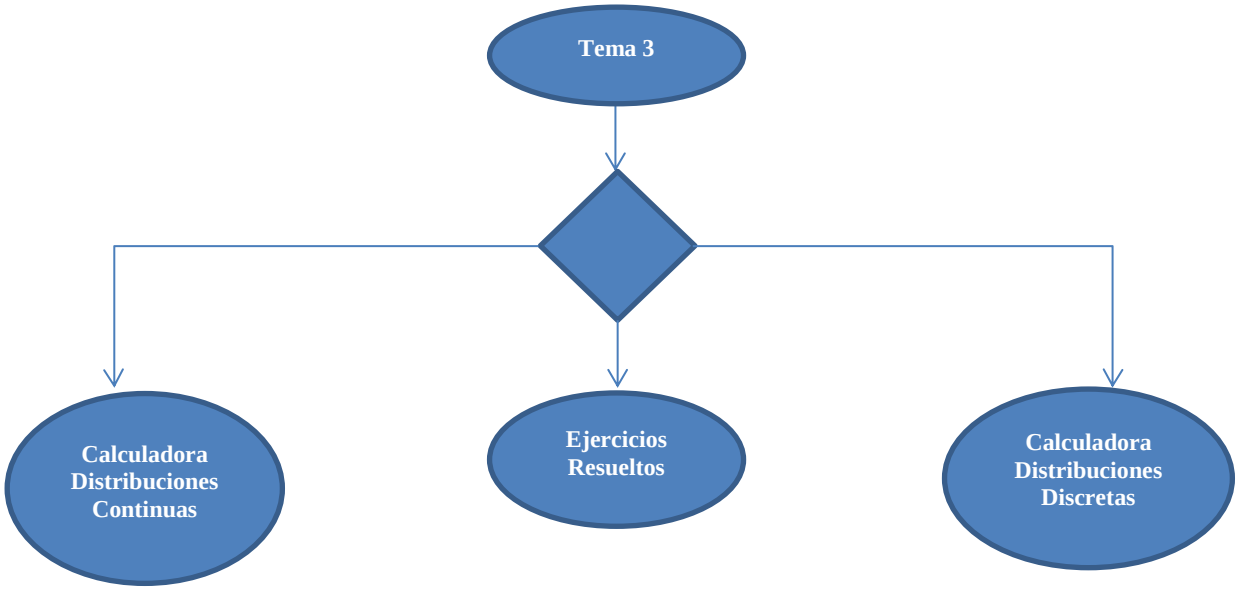

Figura 3-15 Diagrama de Flujo. Tema 3

Dentro de la ventana principal del tema 3 se encuentra un *pushbutton* y tres *Pop-up menu* (Figura 3-16), para acceder a las calculadoras correspondientes creadas para este tema y otro para poder acceder a la batería de ejercicios de este tema.

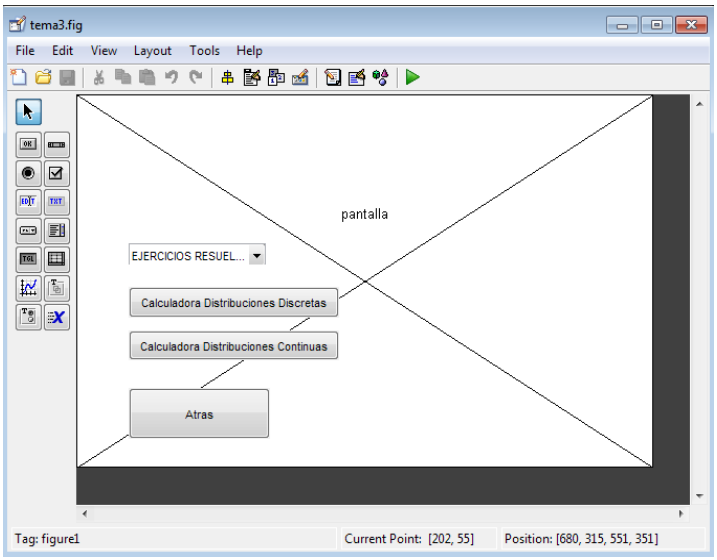

Figura 3-16 Ventana Principal. Tema 3

3.6.1. Calculadora de Distribuciones Continuas

Una distribución de probabilidad es continua cuando los resultados posibles del experimento son de variables cuantitativas en las que se puede tomar cualquier valor, y que resultan principalmente del proceso de medición. Es decir, el valor que toman puede ser cualquiera y no únicamente un número determinado, como pasaría en el caso de las variables discretas [17]. Ejemplos de variables aleatorias continuas son las distintas estaturas de un grupo de personas, el tiempo que se dedica a realizar una actividad o la temperatura de una ciudad.

Se han ideado una serie de calculadoras para las principales distribuciones continuas: exponencial, normal y uniforme (Figura 3-17). De esta manera podemos calcular las principales distribuciones dontinuas y a su vez dibujarlas para comprender mejor el resultado. Su computación se realizó mediante la implantación de las diferentes fórmulas que nos ofrecen las *toolboxes* de estadística del programa. Existen muchas más distribuciones que se pueden añadir, y están contenidas en las mencionadas *toolboxes*.

Las expresiones de la función densidad de probabilidad de las distribuciones continuas implementadas son las que aparecen reflejadas en las ecuaciones 3-23, 3-24 y 3-25:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} * e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < \infty$$

Ecuación 3-23 Distribución Normal

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{para } a \leq x \leq b \\ 0 & \text{para } x<a \text{ o } x>b \end{cases}$$

Ecuación 3-24 Distribución Uniforme

$$f(x)=\begin{cases}\lambda e^{-\lambda x} & \text{para } x\geq 0\\ 0 & \text{para caso contrario}\end{cases}$$

Ecuación 3-25 Distribución Exponencial

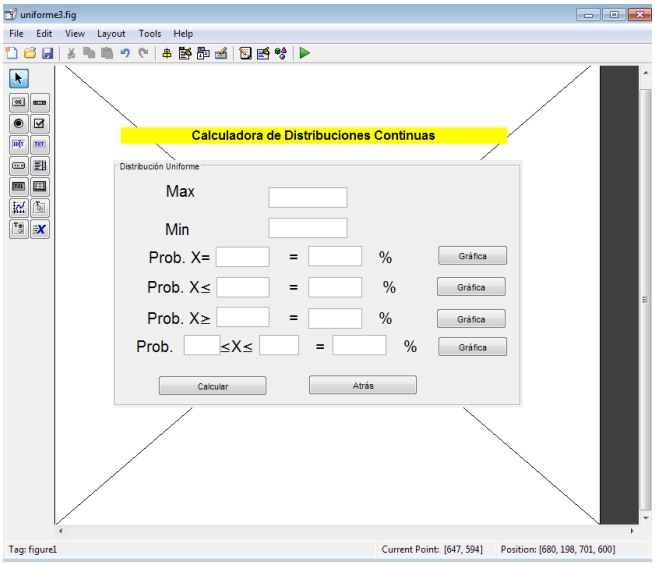

Figura 3-17 Calculadora Distribuciones Continuas. Tema 3

A su vez como se ha mencionado en otras calculadoras, se exige que los alumnos sean conscientes de los datos que están introduciendo, es decir tengan un conocimiento del tema, si esto no es así el resultado obtenido sería erróneo (véase código en Anexo 8).

3.6.2. Calculadora de Distribuciones Discretas

Las distribuciones discretas son aquellas en las que la variable puede pude tomar un número determinado de valores y no un valor cualquiera como sucede en las distribuciones continuas. Un ejemplo de esto sería el tirar una moneda, los únicos valores posibles serían cara o cruz, o el tirar un dado en el que los valores posibles son del 1 al 6 [14].

Se creó de la misma manera que la Calculadora de Distribuciones Continuas, contando así con los mismos botones para desarrollar su funcionalidad (Figura 3-18). La calculadora cuenta con tres posibilidades, calcular la Distribución Binomial, la Distribución de Poisson y la Distribución Geometrica, las cuales hemos considerado esenciales (véase código en Anexo 9).

Las expresiones de las funciones de masa de probabilidad de las distribuciones discretas implementadas aparecen reflejadas en las ecuaciones 3-26, 3-27 y 3-28:

$$P(X=x) = \binom{n}{x} p^x (1-p)^{n-x}\ ,\ 0<x<n$$

Ecuación 3-26 Distribución Binomial

$$P(X=x) = \frac{e^{-\lambda}\lambda^k}{k!}\ ,\ x=0, 1, 2, 3\ldots$$

Ecuación 3-27 Distribución Poisson

$$P(X=x) = (1 - p)^{x-1}p \ , \ x=0, 1, 2, 3\ldots$$

Ecuación 3-28 Distribución de Geometrica

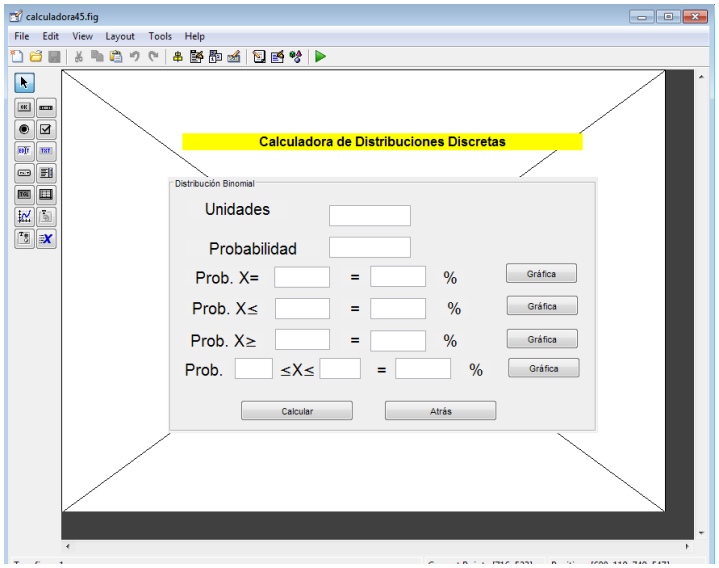

Figura 3-18 Calculadora Distribuciones Discretas. Tema 3

3.6.3. Ejercicios Resueltos

Para finalizar el Tema 3 de variables aleatorias, ya sean discretas o continuas, se vuelve a contar con otra batería de ejercicios del tema donde el alumno podrá comprobar los conocimientos adquiridos tras el estudio del mismo. La implementación es similar a la de los temas anteriores por lo que no se volverá a exponer.

3.7. Tema 4. Inferencia estadística.

La estadística inferencial es una parte de la estadística que comprende los métodos y procedimientos que por medio de la inducción determinan propiedades de una población estadística, a partir de una pequeña parte de la misma. La estadística inferencial comprende como aspectos importantes:

- La toma de muestras o muestreo
- La estimación de parámetros o variables estadísticas
- El contraste de hipótesis
- El diseño experimental.
- La inferencia bayesiana
- Los métodos no paramétricos

Dentro del cuarto tema, tras el análisis de los contenidos que abarca, se opta por la realización de una calculadora de intervalos de confianza, además de la consabida batería de ejercicios resueltos (Figura 3-19).

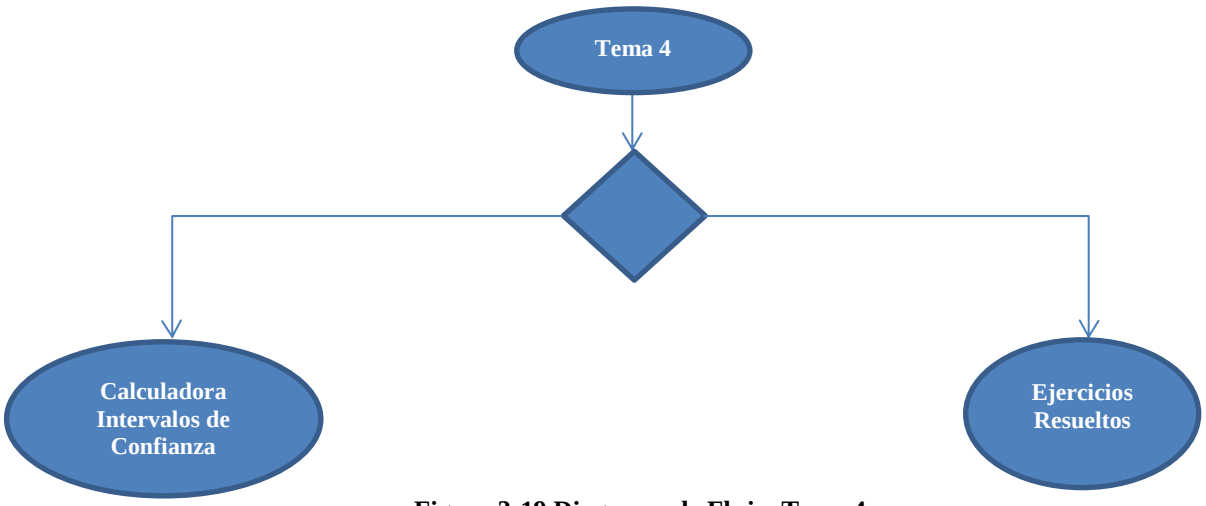

Figura 3-19 Diagrama de Flujo. Tema 4

Como viene pasando en cada uno de los tres anteriores temas, al entrar en la pantalla principal del tema se encuentra un *pop up menu* con los ejercicios resueltos y dos *pushbuttons* que dan acceso a la calculadora de Intervalo de Confianza o la vuelta a la ventana principal de selección de tema.

3.7.1. Calculadora Intervalos de Confianza

La estadística inferencial es el proceso de uso de los resultados derivados de las muestras de una población para sacar conclusiones acerca de sus características. La estadística inferencial nos permite realizar estimaciones desconocidas. Existe dos tipos de estimaciones, las puntuales y las de intervalo.

Con esta calculadora se obtiene el intervalo, rango de números, que se construye a partir de la estimación puntual. Este se construye de manera que haya mas probabilidad que el parámetro que se está estudiando esté localizado dentro de dicho intervalo [16].

Su creación cuenta con una serie de *edit text* para introducir los datos requeridos, y dos *pushbuttons* para realizar los cálculos que se requieran gracias a las ecuaciones establecidas en el *script* de Matlab (véase código en Anexo10). En la Figura 3-20 se muestra el resultado obtenido.

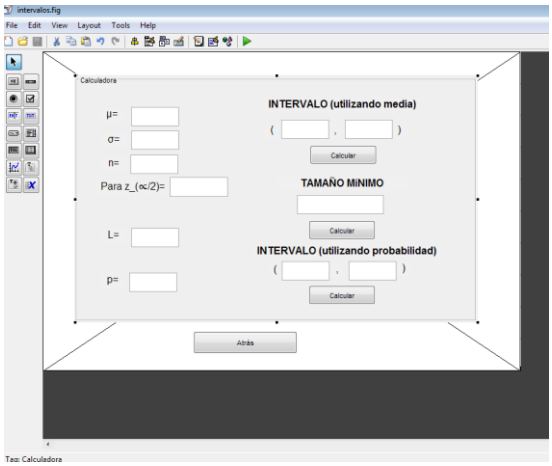

Figura 3-20 Calculadora Intervalos de Confianza. Tema 4

Para el cálculo de los intervalos como el del tamaño míno que debe tener un suceso para ocurrir, se procedió de la siguiente manera:

- Los intervalos se calculan utilizando la Ecuacion 3-28 ya sea si tenemos la probabilidad de ocurrencia, o utilizando la Ecuacion 3-29 teniendo la media de los sucesos.

$$Intervalo = (p - z_{\alpha/2} * \sqrt[2]{\left(\frac{p(1-p)}{n}\right)}, (p + z_{\alpha/2} * \sqrt[2]{\left(\frac{p(1-p)}{n}\right)})$$

Ecuación 3-29 Intervalo de Confianza con probabilidad

$$Intervalo = (\mu - \frac{z_{\alpha/2}}{\sqrt[2]{n}}, \mu + \frac{z_{\alpha/2}}{\sqrt[2]{n}},)$$

Ecuación 3-30 Intervalo de Confianza con media

- Para el cálculo del tamaño mínimo se utilizo la siguiente Ecuacion 3-30.

$$N=2*z_{\alpha/2} * \frac{\sigma}{L}$$

Ecuación 3-31 Tamaño mínimo

Siendo L la longitud mínima y σ la desviación

3.7.2. Ejercicios Resueltos

Por ultimo hacer mención como en todos los temas a los ejercicios resueltos propuestos tras el estudio del contenido. El alumno cuenta con cuatro nuevos ejercicios que cuentan con las posibilidades de conocer en todo momento qué apartado es correcto o incorrecto, y poder comprobar su solución.

4. PRESENTACIÓN DE RESULTADOS.

En los siguientes apartados se mostrarán los resultados de validación de la aplicación. Dicha validación sigue un proceso dividido en dos partes:

- Comprobación del correcto funcionamiento de la aplicación. Se comprueba el correcto funcionamiento de la aplicación frente a la introducción de datos esperados en los edit text, así como frente a la pulsación de los distintos pushbutton y/o Pop-up menu de las distintas ventanas.
- Comprobación de robustez de la aplicación. Se comprueba si, introduciendo datos que no son los esperados, la aplicación responde de forma correcta.

4.1. Acceso a la Aplicación

Para acceder a la aplicación habrá que seguir los siguientes pasos:

- Ejecutar el programa Matlab desde su acceso directo en el escritorio o bien desde las aplicaciones.
- Ejecutar el programa principal dentro de Matlab, sin más que arrastrar el script .m hasta la ventana de comandos (command window) de Matlab o abriendo el GUIDE. El archivo que deberán seleccionar será Bienvenido.m.

Otra de las maneras de acceder a la aplicación será el abrir un ejecutable (Benvenido.exe), que teniendo la misma versión o posterior a Matlab2015a permitirá iniciar sin tener que abrir el software. Para acceder a través del archivo.exe. si no se tiene instalado el Matlab, hay que instalar previamente el instalador del Runtime (MCRInstaller.exe).

4.2. Bienvenido

La aplicación desarrollada en el presente TFG presenta una pantalla de inicio donde el alumno dependiendo de en qué Universidad esté utilizando el programa tendrá unos diferentes datos de acceso.

Una vez se inicia la aplicación, la primera ventana que aparece pide usuario y contraseña (Figura 4-1). Al introducir el usuario “alumno” y contraseña “escuelanaval” se accede a la aplicación en sí, si esto no es así el resultado sería un mensaje de error, el cual no deja avanzar com el programa.

Cuando se verifiquen los datos de acceso se accede a una nueva ventana donde se podrán seleccionar cualquiera de los temas de la asignatura de estadística. Dependiendo del tema que se haya seleccionado el alumno tendrá acceso a una serie de aplicaciones donde podrá poner a prueba sus conocimientos o le servirán de ayuda a la hora de realizar distintos ejercicios.

Figura 4-1 Bienvenido

4.3. Ventana Principal

Una vez se ha entrado, se comprueba mediante selección de botones en la pantalla principal el acceso a los contenidos de cada uno de los temas de la materia. Se tienen 4 opciones, cada una con el nombre del Tema de la asignatura, véase en la Figura 4-2.

Figura 4-2 Ventana Principal

Dentro de cada tema se pueden ver las posibilidades que estos ofrecen. La aplicación se ha centrado en dos cosas, ejercicios resueltos para poner a prueba la capacidad del alumno y a su vez enseñarle a resolverlos en el caso de duda, y calculadoras para resolver los principales problemas de cada uno de los temas. Algunas de estas calculadoras incluyen gráficas para hacer de los ejercicios más visuales y mejorar el entendimiento por parte del alumno.

4.4. Ejercicios Resueltos

Dentro de cada uno de los temas se encuentran los menús desplegables de los ejercicios resueltos. Cada uno cuenta con su propia batería de problemas y su selección da acceso a ellos. En el momento de acceder a la pantalla de alguno de los ejercicios se puede comprobar cómo la opción "ejercicio resuelto" no se puede seleccionar y lo único que se puede hacer es volver atrás o comprobar si es correcto o no el resultado, véase en Figura 4-3.

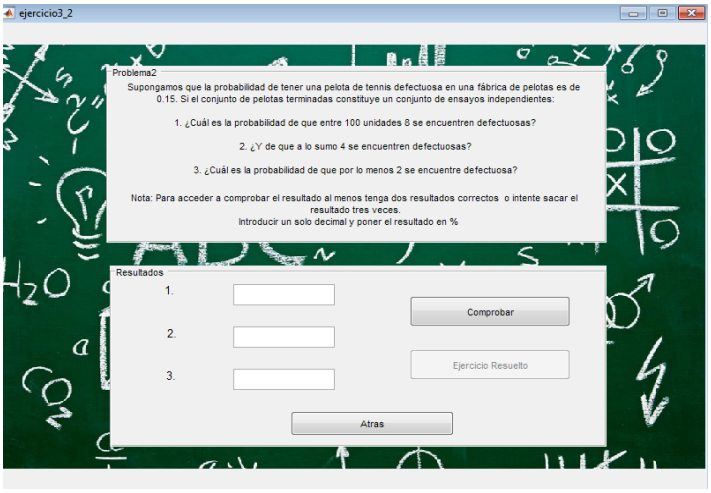

Figura 4-3 Ejercicios Resueltos

Una vez introducimos los datos, si el resultado es correcto, saldrá un mensaje para comunicarlo y si no es así se indica el apartado o apartados incorrectos (Figura 4-4). Tras comprobarlo varias veces da acceso al botón Ejercicio resuelto, que da acceso a la solución desarrollada. De esta manera el alumno intenta hacer los ejercicios suponiéndole un reto. Este proceso ha sido realizado para todos los demás ejercicios dando el mismo resultado.

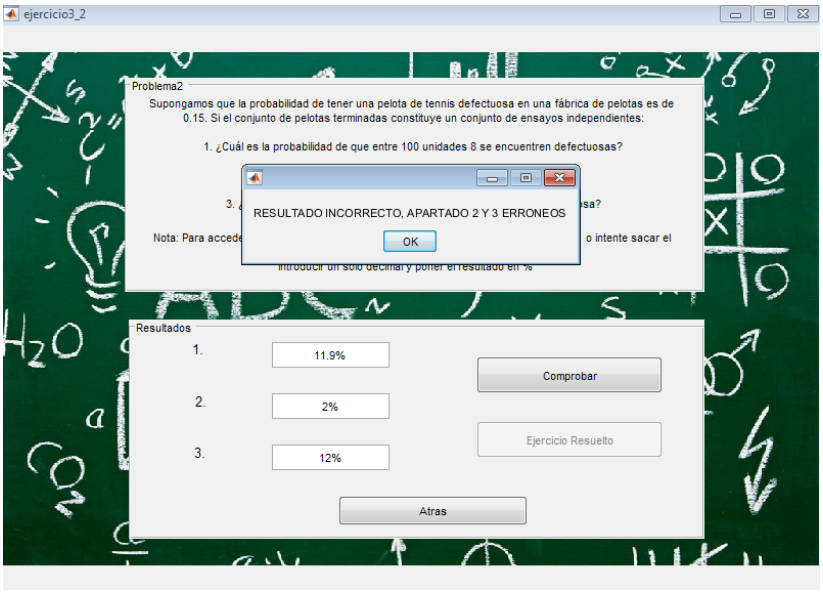

Figura 4-4 Ejercicios Resueltos

4.5. Calculadoras

Como ya se ha mencionado, cada tema cuenta con una serie de calculadoras que ayudarán a resolver los ejercicios que se plantean en la asignatura. Exiten un total de 6 calculadoras implementadas para todos los temas.

4.5.1. Calculadora de Medidas de posición

En esta calculadora lo primero que se visualiza es una pantalla grande con dos cajas alargadas donde pone las letras X e Y, cada una equivalente a datos y frecuencias respectivamente. Al introducir los datos separados por un espacio, se puede empezar a hacer uso de ella.

La calculadora ofrece diferentes opciones. El primer paso a realizar será el introducir los valores en la tabla, sin este paso la calculadora no funcionará. Acto seguido se presiona el botón calcular y dará el resultado de cada una de las variables que se requieran, es decir, media aritmética, geéometrica, cuadrática y armónica, varianza, desviación y moda (Figura 4-5). También en el caso que se desee, se puede obtener un histograma donde visualizar los datos con sus respectivas frecuencias. De esta manera, se consigue que el proceso de realizar la tabla de datos y sus respectivas operaciones por parte del usuario se reduzcan considerablemente.

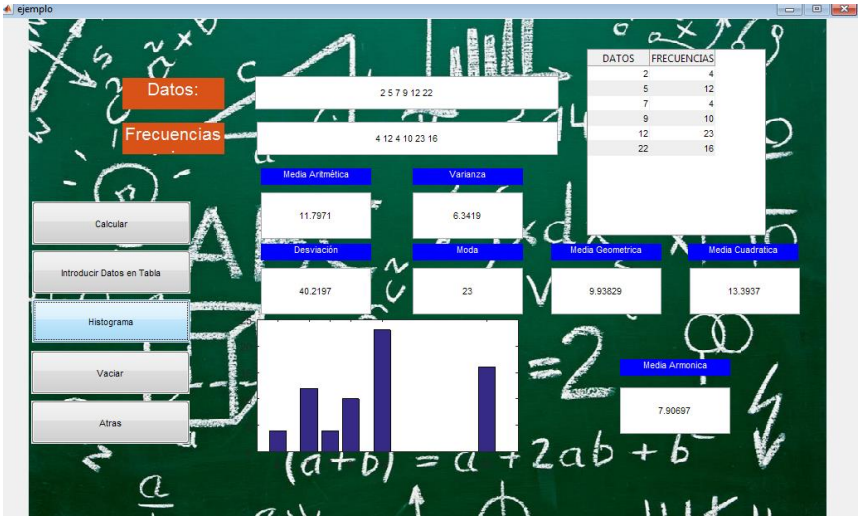

Figura 4-5 Medidas de posición, dispersión y forma

4.5.2. Calculadora de Regresión Lineal

Esta calculadora permite calcular la ecuación de regresión lineal tras un proceso muy parecido al anterior. En primer lugar se introducen datos y se meten dentro de la tabla. Acto seguido al presionar los botones de “calcular” y “gráfica” se obtienen los resultados solicitados. Tras la computación del código la calculadora es capaz de proporcionar la ecuación exacta de regresión. Dentro de la gráfica se puede ver la nube de puntos que se ha introducido y la relación entre los mismos (Figura 4-6).

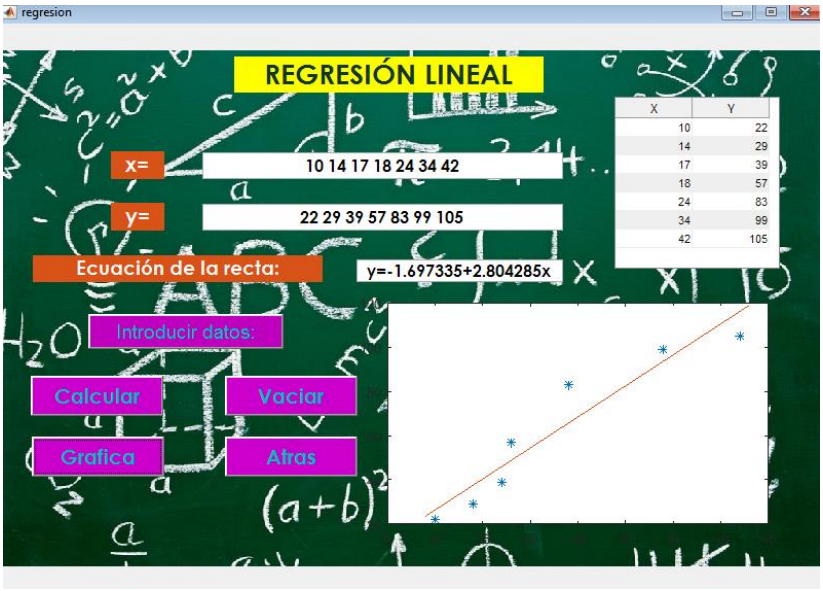

Figura 4-6 Regresión Lineal

4.5.3. Calculadora de Probabilidad

Una vez se ha entrado en la calculadora de probabilidad se encontrará una pantalla como la Figura 4-7. En esta calculadora, se introducen los datos de las probabilidades de dos sucesos A y B pertenecientes a un mismo experimento aleatorio y a partir de ellos, definiendo el tipo de sucesos que sean, se obtendrán los resultados correspondientes.

Esta calculadora permitirá al usuario ahorrar tiempo a la hora de realizar cálculos y comprobar la diferencia entre los tipos de sucesos.

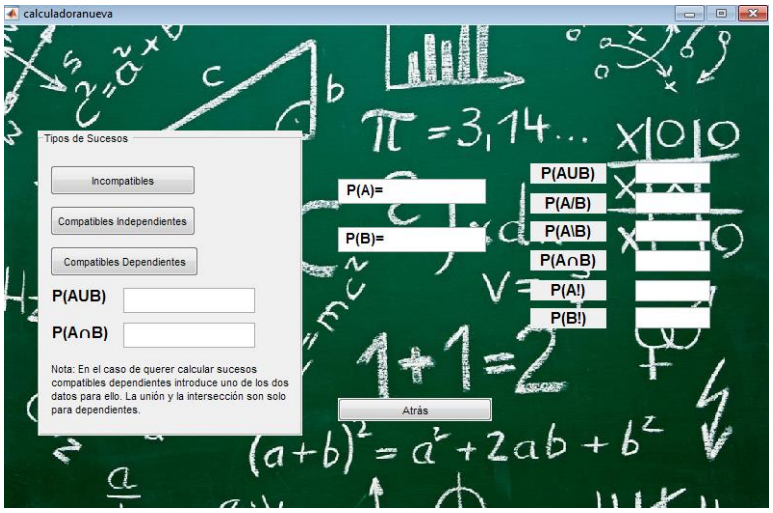

Figura 4-7 Calculadora Probabilidad

4.5.4. Calculadora de Distribuciones Continuas y Discretas

Dentro del tema 3 se tiene la opción de acceder a las calculadoras de distribuciones continuas y discretas. Dentro de cada opción se puede elegir el tipo de distribución a realizar.

El cálculo de distribuciones es un proceso largo en el cual hace falta incluso comprobar datos en tablas para hallar los resultados correctamente. Gracias a Matlab y el uso de sus *toolboxes* de estadística este tiempo se puede reducir notablemente.

Para poder trabajar con las calculadoras primeramente se han de introducir los parámetros que caracterizan la distribución seleccionada (media, desviación típica, número de intentos, probabilidad

de éxito....). Una vez introducidos, dentro de cada calculadora se tiene la opción de elegir la probabilidad de que un suceso sea igual, mayor o menor a los valores que el usuario determine. Gracias a su uso el alumno puede agilizar el proceso y obtener resultados precisos y fiables (Figura 4-8). A su vez, todas estas calculadoras cuentan con la opción de visualizar gráficas que ayuden a comprender mejor el resultado, véase en la Figura 4-9.

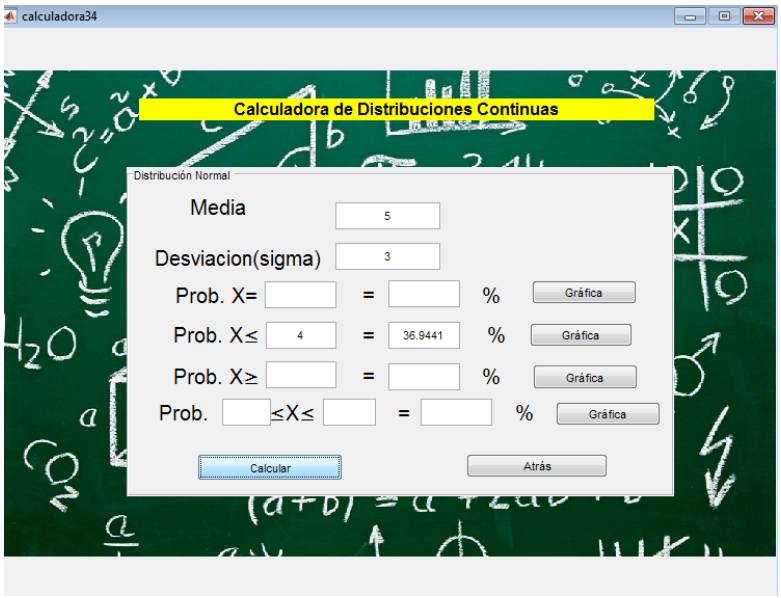

Figura 4-8 Calculadora de Distribuciones Continuas

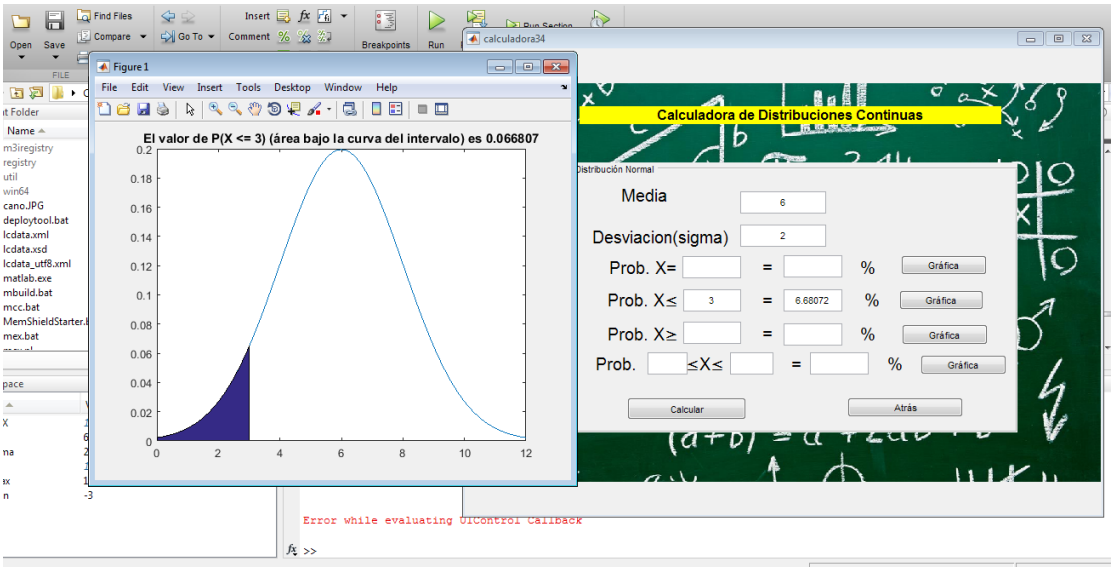

Figura 4-9 Gráfica Distribuciones

4.5.5. Calculadora de Intervalos de Confianza

Tras haber accedido al cuarto y ultimo tema de la aplicación se tendra acceso a la última calculadora. Dentro de este tema, los ejercicios más tratados son aquellos acerca de intervalos de confianza, y es por ello el motivo por el que se ideó la calculadora. Como el resto, en esta se

introducen los datos del problema (media, deviación típica y tamaño de la muestra), y mediante las fórmulas computadas en el código se es capaz de calcular lo exigido.

Gracias a los problemas del tema y algunos otros se ha podido verificar que los resultados son los esperados dando asi por finalizada la aplicación con éxito.

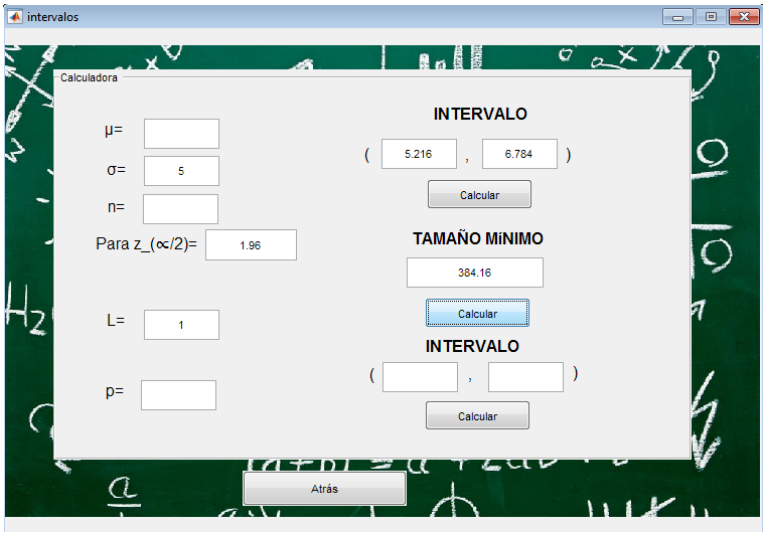

Figura 4-10 Calculadora de Intervalos de Confianza

4.6. Robustez de la herramienta

Uno de los fallos que podía dar la aplicación era debido a la mala introducción de datos dentro de las calculadoras. Esto se arregló mediante la computación de código de manera que estas avisan del error en los datos ya sea porque bien faltan parte de ellos para realizar los cálculos, se han introducido valores no númericos, se ha introducido más de un valor en las casillas o si los datos proporcionados no tienen sentido.

De esta manera se consigue comprobar lo que el usuario esta haciendo mal a la hora de su uso, y así poder remediarlo. En las Figuras 4-11 a 4-14 se presentan algunos ejemplos de la batería de pruebas realizada para comprobar la respuesta de la aplicación frente a la introducción de datos erróneos.

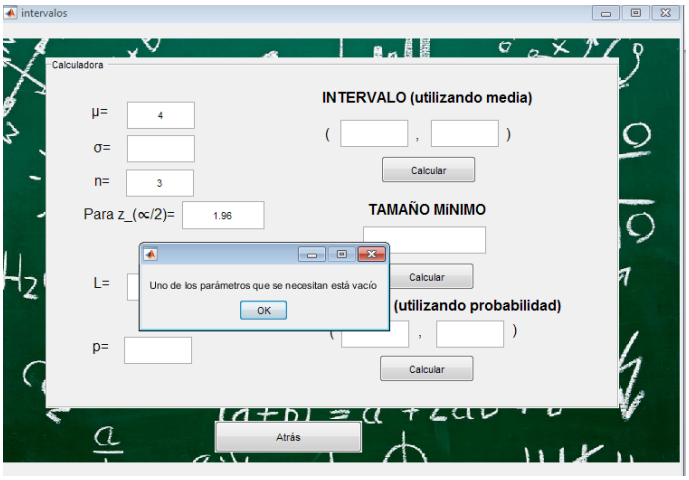

Figura 4-11 Error por falta de datos

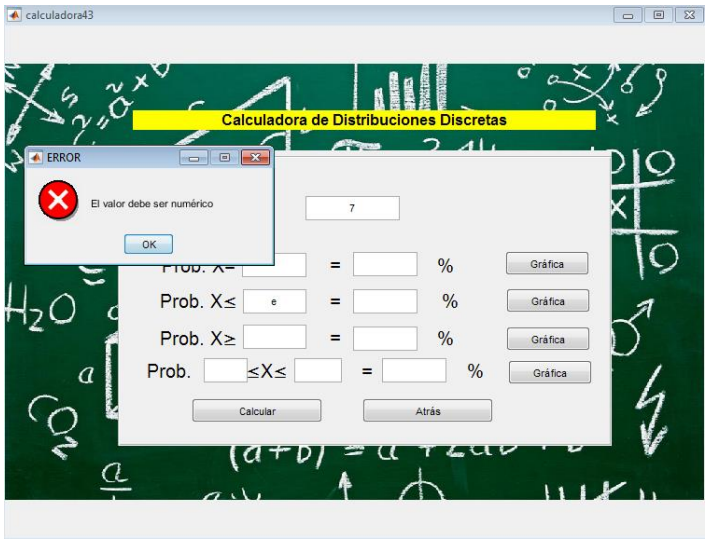

Figura 4-12 Error por valores no numéricos

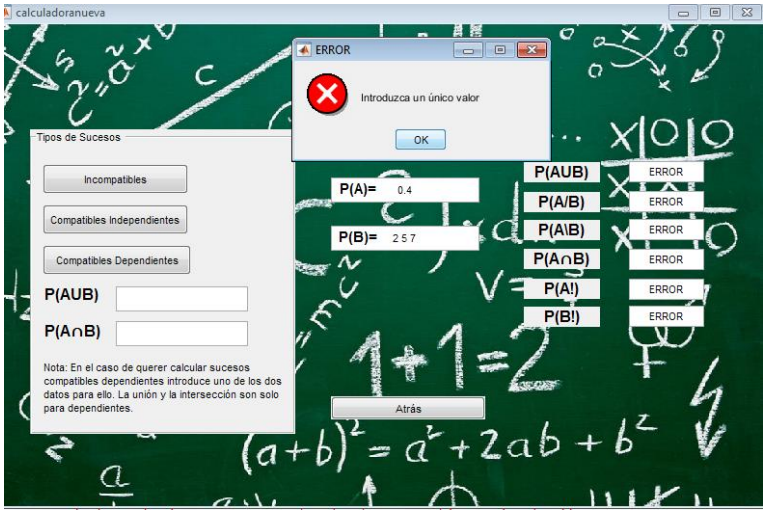

Figura 4-13 Error por introducir más de un dato

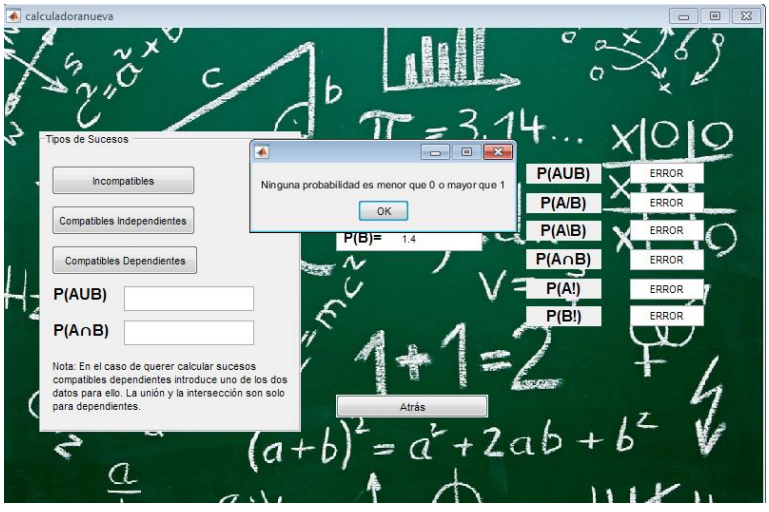

Figura 4-14 Error datos no reales

5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1. Conclusiones

Una vez terminadá por completo la aplicación se pueden sacar bastantes conclusiones acerca de cómo ha evolucionado el trabajo. En primer lugar se ha creado la herramienta de manera que cumple los objetivos fijados. La herramienta se creó para ayudar al alumno en la materia, sobre todo en la parte práctica. Gracias a ella se comprueba que el tiempo que el alumno emplea en los problemas se reduce y puede obtener a su vez resultados exactos.

Una de las cosas que se buscaba era la interactuación del alumno con la máquina. Se buscaba que el alumno fuese capaz de entender la asignatura, y eso también ha sido posible gracias a las calculadoras. Estan fueron implementadas de forma que el alumno tuviese que saber la materia con anterioridad a la realización de los problemas y de esta manera supiese en todo momento qué datos introducir en cada casilla.

Otro de los objetivos que se cumplió, aunque no tan bien como se hubiese esperado, fue el de facilitar la labor docente del profesor. Gracias a la aplicación se puede trabajar la materia con facilidad pero el profesor aun así tendrá que supervisar el trabajo del alumno debido a que la realización de ejercicios no implica el entender la materia al 100%, en este sentido se puede decir que habría que mejorar.

Por otra parte se establecieron una serie de ejercicios en el temario en los cuales el alumno puede poner a prueba sus conocimientos y le permiten el saber si hacen los distintos apartados correctamente así como visualizar su solución. Aun así este aspecto se puede mejorar también proporcionando más variedad de ejercicios, debido a que el alumno una vez terminados los propuestos no tendría más. En este aspecto debería tomar parte el tutor y modificarlos cada cierto tiempo.

Ya por último, la creación de la aplicación ha permitido al autor de este trabajo comprender la importancia de la estadística tanto en la Armada como algo útil y necesario para ser oficial de marina así como su aplicación en el mundo actual. Gracias al programa no solo se pueden resolver problemas que aparezcan en libros sino también conocer estadísticas acerca de datos reales.

5.2. Líneas futuras

De acuerdo con las conclusiones anteriormente citadas y después de haber realizado este estudio, se proponen las siguientes líneas futuras para mejorar el proyecto:

- Ampliar la gama de ejercicios proporcionando más variedad al alumno dándole así mayor capacidad para probar sus conocimientos. En concreto, sería interesante el desarrollar un

módulo en la aplicación que permitiese introducir ejercicios y soluciones sin tener que acceder al código de la misma, tal como habría que hacerlo actualmente.

- Establecer formularios. Unicamente existen en la calculadora de probabilidad, pero se podría mejorar proporcionando al alumno formularios de cada operación que esta realizando y el como utilizarla. De esta manera se garantiza su aprendizaje.

- Introducir videos de explicaciones. Las píldoras antes mencionadas en el estado del arte son un método por el cual el alumno puede volver a asistir a la clase realizada online y poder volver hacia atrás si hay algo que este no entienda

- Ampliar las calculadoras de distribuciones. Ahora mismo están incluidas únicamente las más utilizadas en cuanto a las discretas y continúas.

- Introducir mejoras a la hora de calcular probabilidad. La calculadora de probabilidad esta creada para calcular únicamente con dos sucesos, mientras que esta se podría mejorar introduciendo más y nuevas formulas.
- Crear un módulo que permita gestionar distintos usuarios, en términos de seguimiento personalizado del uso de la aplicación.

6. BIBLIOGRAFÍA

[1] Belfiori, «Enseñanza de Estadística con Recursos TIC,» 2014.

[2] «MathWorks,» [En línea]. Available: http://es.mathworks.com/discovery/matlab-gui.html. [Último acceso: 31 Enero 2016].

[3] M. Arellano, «Exposición MATLAB para Análisis Económico, Nivel Básico, Introducción a la creación de Interfaces Gráficas con GUIDE,» 2013.

[4] E. M. S. Sánchez, «Universidad de Huelva,» [En línea]. Available: http://www.uhu.es/cine.educacion/didactica/0071tecnologiaaulas.htm. [Último acceso: 27 Enero 2016].

[5] L. F. Rivera Galicia, «Recursos informáticos para la docencia en Estadística: Técnicas de Muestreo y Análisis de Datos,» 2013.

[6] B. Pateiro Lopez, «Introducción a lenguajes avanzados de computación: MATLAB en la docencia en química».

[7] J. M. Pavia, «Docencia en Estadística, Experiencia de Innovación».

[8] «Software Shop,» [En línea]. Available: www.software-shop.com/SimStat. [Último acceso: 27 Enero 2016].

[9] «UNESCO» [En línea]. Available: http://www.unesco.org/webworld/idams/selfteaching/spa/spresentation.htm. [Último acceso: 27 Enero 2016].

[10] V. MArtin, «Stadis,» Malavida, [En línea]. Available: http://stadis.malavida.com/. [Último acceso: 27 Enero 2016].

[11] «INFOSTAT» Universidad Nacional de Córdoba, [En línea]. Available: http://www.infostat.com.ar/.

[12] «Metodología Didactica,» Unidad de Promoción y desarrollo "La Salina", Salamanca, 2013.

[13] M. A. Campo Cabana y F. J. Fernandez Fernandez, «Estadística. Notas y Ejercicios Resueltos,» Pontevedra, 2014.

[14] Docentes Innovadores, [En línea]. Available: http://docentesinnovadores.net/Archivos/5734/DISTRIBUCIONES%20CONTINUAS.pdf. [Último acceso: 14 Febrero 2016].

[15] OCTAVE, [En línea]. Available: https://www.gnu.org/software/octave/. [Último acceso: 29 Enero 2016].

[16] Docentes Innovadores, [En línea]. Available: http://docentesinnovadores.net/Archivos/5734/DISTRIBUCIONES%20DISCRETAS.pdf. [Último acceso: 14 Febrero 2016].

[17] M. O. S. Ibujes, «Monografias,» [En línea]. Available: http://www.monografias.com/trabajos91/estimacion-intervalos-confianza/estimacion-intervalos-confianza.shtml#ixzz41lKh4cKv. [Último acceso: 29 Febrero 2016].

[18] «Faitic» Universidad de Vigo, [En línea]. Available: http://www.faitic.uvigo.es/index.php/es/. [Último acceso: Febrero Marzo 2016].

[19] Secretaria de Educación Pública, [En línea]. Available: http://www.secretariadeducacionpublica.com. [Último acceso: 02 Marzo 2016].

[20] «YouTube, GUI - Interfaz Gráfica de Usuario en MATLAB,» [En línea]. Available: https://www.youtube.com/watch?v=jBecrEnkSV0. [Último acceso: 18 Marzo 2016].

[21] «R Project» R, [En línea]. Available: www.r-project.org. [Último acceso: 1 Abril 2015].

ANEXO I: CÓDIGO CONTRASEÑA

```
function varargout = Bienvenido(varargin)
% BIENVENIDO MATLAB code for Bienvenido.fig
%      BIENVENIDO, by itself, creates a new BIENVENIDO or raises the existing
%      singleton*.
%
%      H = BIENVENIDO returns the handle to a new BIENVENIDO or the handle to
%      the existing singleton*.
%
%      BIENVENIDO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in BIENVENIDO.M with the given input arguments.
%
%      BIENVENIDO('Property','Value',...) creates a new BIENVENIDO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Bienvenido_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Bienvenido_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Bienvenido

% Last Modified by GUIDE v2.5 24-Jan-2016 22:00:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Bienvenido_OpeningFcn, ...
                   'gui_OutputFcn',  @Bienvenido_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Bienvenido is made visible.
function Bienvenido_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Bienvenido (see VARARGIN)

% Choose default command line output for Bienvenido
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Bienvenido wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Bienvenido_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in aceptar.
function aceptar_Callback(hObject, eventdata, handles)
USUARIO=get(handles.edit2,'string');
usuario='alumno';
CLAVE=get(handles.edit1,'string');
clave='escuelanaval';
l=strcmp(USUARIO,usuario);
h=strcmp(CLAVE,clave);
if l==1 && h==1;
 V_principal;
 close(handles.figure1);
else
 msgbox('CONTRASEÑA O USUARIO INCORRECTO');
end

% hObject    handle to aceptar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

ANEXO II: CÓDIGO VENTANA PRINCIPAL

```
function varargout = V_principal(varargin)
% V_PRINCIPAL MATLAB code for V_principal.fig
%      V_PRINCIPAL, by itself, creates a new V_PRINCIPAL or raises the existing
%      singleton*.
%
%      H = V_PRINCIPAL returns the handle to a new V_PRINCIPAL or the handle to
%      the existing singleton*.
%
%      V_PRINCIPAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in V_PRINCIPAL.M with the given input arguments.
%
%      V_PRINCIPAL('Property','Value',...) creates a new V_PRINCIPAL or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before V_principal_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to V_principal_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help V_principal

% Last Modified by GUIDE v2.5 23-Jan-2016 18:23:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @V_principal_OpeningFcn, ...
                   'gui_OutputFcn',  @V_principal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before V_principal is made visible.
function V_principal_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to V_principal (see VARARGIN)

% Choose default command line output for V_principal
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes V_principal wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = V_principal_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im=imread('estadistica.jpg');
axes(handles.pantalla);
imshow(im);

% --- Executes on button press in TEMA1.
function TEMA1_Callback(hObject, eventdata, handles)
% hObject    handle to TEMA1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema1;
close(handles.figure1);


% --- Executes on button press in TEMA2.
function TEMA2_Callback(hObject, eventdata, handles)
% hObject    handle to TEMA2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema2;
close(handles.figure1);


% --- Executes on button press in TEMA3.
function TEMA3_Callback(hObject, eventdata, handles)
% hObject    handle to TEMA3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema3;
close(handles.figure1);


% --- Executes on button press in TEMA4.
function TEMA4_Callback(hObject, eventdata, handles)
% hObject    handle to TEMA4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema4;
close(handles.figure1);
```

ANEXO III: CÓDIGO TEMA 1

```
function varargout = tema1(varargin)
% TEMA1 MATLAB code for tema1.fig
%      TEMA1, by itself, creates a new TEMA1 or raises the existing
%      singleton*.
%
%      H = TEMA1 returns the handle to a new TEMA1 or the handle to
%      the existing singleton*.
%
%      TEMA1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TEMA1.M with the given input arguments.
%
%      TEMA1('Property','Value',...) creates a new TEMA1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before tema1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to tema1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help tema1

% Last Modified by GUIDE v2.5 08-Feb-2016 11:23:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tema1_OpeningFcn, ...
                   'gui_OutputFcn',  @tema1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before tema1 is made visible.
function tema1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to tema1 (see VARARGIN)

% Choose default command line output for tema1
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes tema1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = tema1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
varargout{1} = handles.output;
im=imread('estadistica.jpg');
axes(handles.pantalla);
imshow(im);

% --- Executes on button press in atras.
function atras_Callback(hObject, eventdata, handles)
% hObject    handle to atras (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
V_principal;
close(handles.figure1);


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
a=get(hObject,'value');
switch a
    case 2
        ejercicio1_1;
        close(handles.figure1);
    case 3
        ejercicio1_2;
        close(handles.figure1);
    case 4
        ejercicio1_3;
        close(handles.figure1);
    case 5
        ejercicio1_4;
        close(handles.figure1);
end


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
regresion;
close(handles.figure1);


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
medidasdispersion;
close(handles.figure1);
```

ANEXO IV: CÓDIGO CALCULADORA REGRESIÓN

```
function varargout = regresion(varargin)
% REGRESION MATLAB code for regresion.fig
%      REGRESION, by itself, creates a new REGRESION or raises the existing
%      singleton*.
%
%      H = REGRESION returns the handle to a new REGRESION or the handle to
%      the existing singleton*.
%
%      REGRESION('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in REGRESION.M with the given input arguments.
%
%      REGRESION('Property','Value',...) creates a new REGRESION or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before regresion_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to regresion_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help regresion

% Last Modified by GUIDE v2.5 14-Apr-2016 08:08:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @regresion_OpeningFcn, ...
                   'gui_OutputFcn',  @regresion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before regresion is made visible.
function regresion_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to regresion (see VARARGIN)

% Choose default command line output for regresion
```

```
handles.output = hObject;
datos=get(handles.uitable1,'Data');
datos=[];
set(handles.uitable1,'Data',datos);
axes(handles.axes1);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes regresion wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = regresion_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im=imread('numeros.jpg');
axes(handles.pantalla);
imshow(im);
axes(handles.axes1);



% --- Executes on button press in calcular.
function calcular_Callback(hObject, eventdata, handles)
% hObject    handle to calcular (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
xyz=(get(handles.uitable1,'data'));
x=xyz(:,1);
y=xyz(:,2);
n=length(x);
A1=0; A2=0; A3=n; B1=0; B2=0; A9=0;
for i=1:n
        A1=A1+x(i)^2;
        A2=A2+x(i);
        A4=A2;
        B1=B1+x(i)*y(i);
        B2=B2+y(i);
        A9=A2^2;
end
    a=(B1*A3-B2*A2)/(A1*A3-A9);
    b=(B2-a*A2)/(A3);
    funcion=sprintf('y=%8.2f+%8.2fx',b,a);
    set(handles.edit3,'string',funcion);



% --- Executes on button press in atras.
function atras_Callback(hObject, eventdata, handles)
% hObject    handle to atras (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema1;
close(handles.figure1);
```

```
% --- Executes on button press in grafica.
function grafica_Callback(hObject, eventdata, handles)
% hObject    handle to grafica (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
xyz=(get(handles.uitable1,'data'));
if isempty(xyz)
    errordlg('Meter los datos primero en la tabla');
    return
end
x=xyz(:,1);
y=xyz(:,2);
n=length(x);
A1=0; A2=0; A3=n; B1=0; B2=0; A9=0;
for i=1:n
        A1=A1+x(i)^2;
        A2=A2+x(i);
        A4=A2;
        B1=B1+x(i)*y(i);
        B2=B2+y(i);
        A9=A2^2;
end
    a=(B1*A3-B2*A2)/(A1*A3-A9);
    b=(B2-a*A2)/(A3);
    funcion=sprintf('y=%8.2f+%8.2fx',b,a);
    set(handles.edit3,'string',funcion);
    xx=(min(x)-1):0.2:(max(x)+1);
yy=a.*xx+b;
axes(handles.axes1);
plot(x,y,'*',xx,yy);


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.




function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=(get(handles.edit1,'string'));
y=(get(handles.edit2,'string'));
if isempty(x)||isempty(y)
    errordlg('Algún campo está vacío', 'Error');
    return;
end;
x=str2num(x);
y=str2num(y);

if isempty(x)||isempty(y)
    errordlg('Alguno de los datos es erróneo','Error');
```

```
    return;
else
    if (size(x,1) ~= size(y,1))||(size(x,2) ~= size(y,2))
       errordlg('Los campos no tienen el mismo número de elementos','Error');
       return;
    end;
end;

datos=get(handles.uitable1,'Data');

aux = [x' y'];
datos = [datos; aux];
set(handles.uitable1,'Data',datos);



% if isempty(datos)
%     datos(:,1)=x;
%     datos(:,2)=y;
% else
%     datos(:,1)= [datos(:,1) x];
%     datos(:,2)=[datos(:,2) y];
% end




% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.uitable1,'Data',[]);
set(handles.edit1,'String','');
set(handles.edit2,'String','');
set(handles.edit3,'String','');
axes(handles.axes1);
cla reset;
```

ANEXO V: CÓDIGO EJERCICIOS RESUELTOS

```
function varargout = problema4_4(varargin)
% PROBLEMA4_4 MATLAB code for problema4_4.fig
%      PROBLEMA4_4, by itself, creates a new PROBLEMA4_4 or raises the existing
%      singleton*.
%
%      H = PROBLEMA4_4 returns the handle to a new PROBLEMA4_4 or the handle to
%      the existing singleton*.
%
%      PROBLEMA4_4('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROBLEMA4_4.M with the given input arguments.
%
%      PROBLEMA4_4('Property','Value',...) creates a new PROBLEMA4_4 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before problema4_4_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to problema4_4_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help problema4_4

% Last Modified by GUIDE v2.5 24-Feb-2016 18:28:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @problema4_4_OpeningFcn, ...
                   'gui_OutputFcn',  @problema4_4_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before problema4_4 is made visible.
function problema4_4_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to problema4_4 (see VARARGIN)
global a
a=0;
% Choose default command line output for problema4_4
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes problema4_4 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = problema4_4_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im=imread('numeros.jpg');
axes(handles.pantalla);
imshow(im);
 set(handles.pushbutton2,'Enable','off');

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global a
a=a+1;
D=get(handles.edit2,'string');
B=get(handles.edit3,'string');
C=get(handles.edit4,'string');
d='89';
b='5.21';
c='6.78';
e=strcmp(B,b);
f=strcmp(C,c);
h=strcmp(D,d);
if a==3
     set(handles.pushbutton2,'Enable','on');
end
if e==1 & f==1 & h==1
    set(handles.pushbutton2,'Enable','on');
    msgbox('CORRECTO');
end
if e==0 & f==1 & h==1
    msgbox('INTERVALO INCORRECTO');
end
if e==1 & f==0 & h==0
    msgbox('INTERVALO INCORRECTO, TAMAÑO MÍNIMO INCORRECTO');
end
if e==0 & f==0 & h==0
    msgbox('INTERVALO INCORRECTO, TAMAÑO MÍNIMO INCORRECTO');
end
if e==0 & f==1 & h==0
    msgbox('INTERVALO INCORRECTO, TAMAÑO MÍNIMO INCORRECTO');
end
if e==0 & f==0 & h==1
    msgbox('INTERVALO INCORRECTO');
end
if e==1 & f==1 & h==0
```

```
    msgbox('INTERVALO CORRECTO, TAMAÑO MÍNIMO INCORRECTO');
end
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
solucion4_4

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema4;
close(handles.figure1);



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

ANEXO VI: CÓDIGO MEDIDAS DE POSICIÓN

```
function varargout = medidasdispersion(varargin)
% MEDIDASDISPERSION MATLAB code for medidasdispersion.fig
%      MEDIDASDISPERSION, by itself, creates a new MEDIDASDISPERSION or raises the
existing
%      singleton*.
%
%      H = MEDIDASDISPERSION returns the handle to a new MEDIDASDISPERSION or the
handle to
%      the existing singleton*.
%
%      MEDIDASDISPERSION('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MEDIDASDISPERSION.M with the given input
arguments.
%
%      MEDIDASDISPERSION('Property','Value',...) creates a new MEDIDASDISPERSION
or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before medidasdispersion_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to medidasdispersion_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help medidasdispersion

% Last Modified by GUIDE v2.5 09-Apr-2016 14:20:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @medidasdispersion_OpeningFcn, ...
                   'gui_OutputFcn',  @medidasdispersion_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before medidasdispersion is made visible.
function medidasdispersion_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to medidasdispersion (see VARARGIN)

% Choose default command line output for medidasdispersion
handles.output = hObject;
datos=get(handles.uitable1,'Data');
datos=[];
set(handles.uitable1,'Data',datos);




% Update handles structure
guidata(hObject, handles);

% UIWAIT makes medidasdispersion wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = medidasdispersion_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im=imread('numeros.jpg');
axes(handles.pantalla);
imshow(im);



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
xyz=get(handles.uitable1,'data');
if isempty(xyz)
    errordlg('Meter los datos primero en la tabla');
    return
end
x=xyz(:,1);
y=xyz(:,2);
[moda,posmax]=max(y);
xmax=x(posmax)
  set(handles.edit4,'string',xmax);
n=length(x);
s=0;
for i=1:n
s=s+x(i)*y(i);
end
media=(s/sum(y));
  set(handles.edit3,'string',media);
a=0;
for i=1:n
a=a+(x(i)^2*y(i));
end
varianza=((a/sum(y))-media^2);
  set(handles.edit5,'string',varianza);
  desviacion=sqrt(varianza);
  set(handles.edit6,'string',desviacion);
  b=0;
for i=1:n
b=b+x(i)^2*y(i);
end
media1=sqrt((b/sum(y)));
  set(handles.edit8,'string',media1);
  c=1;
for i=1:n
c=c*x(i)^y(i);
end
media2=(c^(1/sum(y)));
  set(handles.edit7,'string',media2);
   d=0;
for i=1:n
d=d+(y(i)/x(i));
end
media3=(d*(1/sum(y)))^-1;
  set(handles.edit9,'string',media3);



% --- Executes on button press in histograma.
function histograma_Callback(hObject, eventdata, handles)
% hObject    handle to histograma (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
xyz=get(handles.uitable1,'data');
x=xyz(:,1);
y=xyz(:,2);
axes(handles.axes1);

abc=bar(x,y);
```

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema1;
close(handles.figure1);


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=(get(handles.edit1,'string'));
y=(get(handles.edit2,'string'));
if isempty(x)||isempty(y)
    errordlg('Algún campo está vacío', 'Error');
    return;
end;
x=str2num(x);
y=str2num(y);

if isempty(x)||isempty(y)
    errordlg('Alguno de los datos es erróneo','Error');
    return;
else
    if (size(x,1) ~= size(y,1))||(size(x,2) ~= size(y,2))
       errordlg('Los campos no tienen el mismo número de elementos','Error');
       return;
    end;
end;

datos=get(handles.uitable1,'Data');

aux = [x' y'];
datos = [datos; aux];
set(handles.uitable1,'Data',datos);

% if isempty(datos)
%     datos(:,1)=datos1;
%     datos(:,2)=frecuencia;
% else
%     datos(:,1)= [datos(:,1) datos1];
%     datos(:,2)=[datos(:,2) frecuencia];
% end


% --- Executes on button press in vaciar.
function vaciar_Callback(hObject, eventdata, handles)
% hObject    handle to vaciar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.uitable1,'Data',[]);
set(handles.edit1,'String','');
set(handles.edit2,'String','');
set(handles.edit3,'String','');
set(handles.edit4,'String','');
set(handles.edit5,'String','');
```

```
set(handles.edit6,'String','');
set(handles.edit7,'String','');
set(handles.edit8,'String','');
set(handles.edit9,'String','');
axes(handles.axes1);
cla reset;



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

ANEXO VII: CÓDIGO CALCULADORA PROBABILIDAD

```
function varargout = calculadoranueva(varargin)
% CALCULADORANUEVA MATLAB code for calculadoranueva.fig
%      CALCULADORANUEVA, by itself, creates a new CALCULADORANUEVA or raises the
existing
%      singleton*.
%
%      H = CALCULADORANUEVA returns the handle to a new CALCULADORANUEVA or the
handle to
%      the existing singleton*.
%
%      CALCULADORANUEVA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CALCULADORANUEVA.M with the given input
arguments.
%
%      CALCULADORANUEVA('Property','Value',...) creates a new CALCULADORANUEVA or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before calculadoranueva_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to calculadoranueva_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help calculadoranueva

% Last Modified by GUIDE v2.5 11-Apr-2016 23:24:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @calculadoranueva_OpeningFcn, ...
                   'gui_OutputFcn',  @calculadoranueva_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before calculadoranueva is made visible.
function calculadoranueva_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to calculadoranueva (see VARARGIN)

% Choose default command line output for calculadoranueva
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes calculadoranueva wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = calculadoranueva_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im=imread('numeros.jpg');
axes(handles.pantalla);
imshow(im);



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function A_Callback(hObject, eventdata, handles)
% hObject    handle to A (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of A as text
%        str2double(get(hObject,'String')) returns contents of A as a double
```

```
% --- Executes during object creation, after setting all properties.
function A_CreateFcn(hObject, eventdata, handles)
% hObject    handle to A (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function B_Callback(hObject, eventdata, handles)
% hObject    handle to B (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of B as text
%        str2double(get(hObject,'String')) returns contents of B as a double


% --- Executes during object creation, after setting all properties.
function B_CreateFcn(hObject, eventdata, handles)
% hObject    handle to B (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function C_Callback(hObject, eventdata, handles)
% hObject    handle to C (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of C as text
%        str2double(get(hObject,'String')) returns contents of C as a double


% --- Executes during object creation, after setting all properties.
function C_CreateFcn(hObject, eventdata, handles)
% hObject    handle to C (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function D_Callback(hObject, eventdata, handles)
% hObject    handle to D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of D as text
%        str2double(get(hObject,'String')) returns contents of D as a double


% --- Executes during object creation, after setting all properties.
function D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=(get(handles.A,'string'));
y=(get(handles.B,'string'));
if isempty(x)||isempty(y)
    msgbox('Uno de los parámetros  principales está vacío');
    return;
end;
x=str2num(x);
y=str2num(y);
if size(x,1)>1 || size(x,2)>1 || size(y,1)>1 || size(y,2)>1
   errordlg('Introduzca un único valor','ERROR');
else
    if isempty(x) || isempty(y)
        errordlg('Introduzca un valor numerico')
        return
    else
        if x>1 || x<0 || y>1 || y<0
      M= 'ERROR';
     set(handles.edit1,'string',M);
     set(handles.edit2,'string',M);
     set(handles.edit3,'string',M);
     set(handles.edit4,'string',M);
     set(handles.edit5,'string',M);
     set(handles.edit6,'string',M);
     msgbox('Ninguna probabilidad es menor que 0 o mayor que 1')
     return
end
```

```
    barra1=x+y-x*y;
  set(handles.edit1,'string',barra1);
barra12=x;
  set(handles.edit2,'string',barra12);
barra13=x-x*y;
  set(handles.edit3,'string',barra13);
barra14=x*y;
  set(handles.edit4,'string',barra14);
barra15=1-x;
  set(handles.edit5,'string',barra15);
barra16=1-y;
  set(handles.edit6,'string',barra16);
    end
end


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=(get(handles.A,'string'));
y=(get(handles.B,'string'));
if isempty(x)||isempty(y)
    msgbox('Uno de los parámetros  principales está vacío');
    return;
end;
x=str2num(x);
y=str2num(y);
if size(x,1)>1 || size(x,2)>1 || size(y,1)>1 || size(y,2)>1
   errordlg('Introduzca un único valor','ERROR');
else
    if isempty(x) || isempty(y)
        errordlg('Introduzca un valor numerico')
        return
    else
        if x>1 || x<0 || y>1 || y<0
      M= 'ERROR';
     set(handles.edit1,'string',M);
     set(handles.edit2,'string',M);
     set(handles.edit3,'string',M);
     set(handles.edit4,'string',M);
     set(handles.edit5,'string',M);
     set(handles.edit6,'string',M);
     msgbox('Ninguna probabilidad es menor que 0 o mayor que 1')
     return
        end
barra1=x+y;
set(handles.edit1,'string',barra1);
if x+y>1
    errordlg('La suma de dos probabilidades incompatibles nunca es mayor que 1');
     M= 'ERROR';
     set(handles.edit1,'string',M);
     set(handles.edit2,'string',M);
     set(handles.edit3,'string',M);
     set(handles.edit4,'string',M);
     set(handles.edit5,'string',M);
     set(handles.edit6,'string',M);
     return
end
bara12=0;
  set(handles.edit2,'string',bara12);
```

```
bara13=x;
  set(handles.edit3,'string',bara13);
bara14=0;
  set(handles.edit4,'string',bara14);
bara15=1-x;
  set(handles.edit5,'string',bara15);
bara16=1-y;
  set(handles.edit6,'string',bara16);
    end
end

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=(get(handles.A,'string'));
y=(get(handles.B,'string'));
z=(get(handles.D,'string'));
w=(get(handles.C,'string'));

if isempty(x)||isempty(y)
    msgbox('Uno de los parámetros  principales está vacío');
    return;
end;
x = str2num(x);
y = str2num(y);
if y>1 || y<0 || x>1 || x<0
    errordlg('Las probabilidades son entre 0 y 1');
    return
end
if size(x,1)>1 || size(x,2)>1 || size(y,1)>1 || size(y,2)>1
   errordlg('Introduzca un único valor','ERROR');
   return;
else
    if size(x) == [0 0]
        errordlg('El valor debe ser numérico','ERROR');
    end;
    if size(y) == [0 0]
        errordlg('El valor debe ser numérico','ERROR');
    end;
end
if isempty(w) && isempty(z)
    errordlg('Tienes que proporcionar al menos una de las dos probabilidades')
    return
end
w = str2num(w);
z = str2num(z);
if size(w,1)>1 || size(w,2)>1 || size(z,1)>1 || size(z,2)>1
   errordlg('Introduzca un único valor','ERROR');
end;
if isempty(w) && ~isempty(z)
bra1=x+y-z;
  set(handles.edit1,'string',bra1);
bra12=z/y;
  set(handles.edit2,'string',bra12);
bra13=x-z;
  set(handles.edit3,'string',bra13);
bra14=z;
  set(handles.edit4,'string',bra14);
bra15=1-x;
  set(handles.edit5,'string',bra15);
```

```
bra16=1-y;
  set(handles.edit6,'string',bra16);
end
  if isempty(z) && ~isempty(w)
bra1=w;
  set(handles.edit1,'string',bra1);
bra14=x+y-w;
  set(handles.edit4,'string',bra14);
bra12=bra14/y;
  set(handles.edit2,'string',bra12);
bra13=x-bra14;
  set(handles.edit3,'string',bra13);
bra15=1-x;
  set(handles.edit5,'string',bra15);
bra16=1-y;
  set(handles.edit6,'string',bra16);
  end
if ~isempty(z) && ~isempty(w)
    msgbox('Unicamente introduce uno de los dos datos, borra uno de los dos y
pulsa de nuevo')
end



% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema2;
close(handles.figure1);
```

ANEXO VIII: CÓDIGO CALCULADORA DISTRIBUCIONES CONTINUAS

```
function varargout = calculadora34(varargin)
% CALCULADORA34 MATLAB code for calculadora34.fig
%      CALCULADORA34, by itself, creates a new CALCULADORA34 or raises the
existing
%      singleton*.
%
%      H = CALCULADORA34 returns the handle to a new CALCULADORA34 or the handle
to
%      the existing singleton*.
%
%      CALCULADORA34('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CALCULADORA34.M with the given input arguments.
%
%      CALCULADORA34('Property','Value',...) creates a new CALCULADORA34 or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before calculadora34_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to calculadora34_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help calculadora34

% Last Modified by GUIDE v2.5 12-Apr-2016 09:13:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @calculadora34_OpeningFcn, ...
                   'gui_OutputFcn',  @calculadora34_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before calculadora34 is made visible.
function calculadora34_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to calculadora34 (see VARARGIN)

% Choose default command line output for calculadora34
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes calculadora34 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = calculadora34_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im=imread('numeros.jpg');
axes(handles.pantalla);
imshow(im);


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema3;
close(handles.figure1);


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x= get(handles.edit1,'string');
y= get(handles.edit2,'string');
if isempty(x)||isempty(y)
    msgbox('Uno de los parámetros  principales está vacío');
    return;
end;
x = str2num(x);
y = str2num(y);
if x<y
    errordlg('El máximo es menor que el mínimo');
    return;
end
if size(x,1)>1 || size(x,2)>1 || size(y,1)>1 || size(y,2)>1
   errordlg('Introduzca un único valor','ERROR');
   return;
else
    if size(x) == [0 0]
        errordlg('El valor debe ser numérico','ERROR');
    end;
    if size(y) == [0 0]
        errordlg('El valor debe ser numérico','ERROR');
    end;
end;
a=get(handles.edit3,'string');
b=get(handles.edit4,'string');
c=get(handles.edit5,'string');
d=get(handles.edit6,'string');
e=get(handles.edit7,'string');
if isempty(a),
    set(handles.edit8,'string','');
else
```

```
    a = str2num(a);
    if isempty(a);
        errordlg('El valor debe ser numérico','ERROR');
    else
    A=0;
set(handles.edit8,'string',A);
    end
end;
if isempty(b),
       set(handles.edit9,'string','');
else
    b = str2num(b);
    if isempty(b)
        errordlg('El valor debe ser numérico','ERROR');
    else
    B=normcdf(b,x,y)*100;
set(handles.edit9,'string',B);
    end
end;
if isempty(c),
        set(handles.edit10,'string','');
else
    c = str2num(c);
    if isempty(c)
        errordlg('El valor debe ser numérico','ERROR');
    else
    aaa=1;
bbb=normcdf(c,x,y);
C=(aaa-bbb)*100;
set(handles.edit10,'string',C);
    end
end;
if isempty(d) || isempty(e)
        set(handles.edit11,'string','');
    else
    d = str2num(d);
    e = str2num(e);
    if isempty(d)||isempty(e)
        errordlg('El valor debe ser numérico','ERROR');
    else
    if e>d
ccc=normcdf(e,x,y);
ddd=normcdf(d,x,y);
D=(ccc-ddd)*100;
set(handles.edit11,'string',D);
    end
    end
if d>e
    msgbox('Intervalo mal definido');
    D= 'ERROR';
    set(handles.edit11,'string',D);
end
if d==e
   D=0;
    set(handles.edit11,'string',D);
end
end;
if size(a,1)>1 ||size(a,2)>1 || size(b,1)>1 ||size(b,2)>1 || size(c,1)>1
||size(c,2)>1 || size(d,1)>1 ||size(d,2)>1 || size(e,1)>1 ||size(e,2)>1
     errordlg('Introduzca un único valor','ERROR');
     return;
end;
```

```
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mu= str2num(get(handles.edit1,'string'));
sigma= str2num(get(handles.edit2,'string'));
xmin = mu - 3*sigma;
xmax = mu + 3*sigma;
x = linspace(xmin,xmax, 200);
fdpX = normpdf(x,mu,sigma);
dato = str2num(get(handles.edit4,'string'));
pXmenorigual3 = normcdf(dato,mu,sigma);
xarea=linspace(xmin,dato);
figure;
plot(x,fdpX);
hold on;
area(xarea,normpdf(xarea,mu,sigma));
title(['El valor de P(X <= ',num2str(dato),') (área bajo la curva del intervalo)
es ', num2str(pXmenorigual3)]);
hold off;
```

```
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mu= str2num(get(handles.edit1,'string'));
sigma= str2num(get(handles.edit2,'string'));
xmin = mu - 3*sigma;
xmax = mu + 3*sigma;
x = linspace(xmin,xmax,200);
fdpX = normpdf(x,mu,sigma);
dato = str2num(get(handles.edit3,'string'));
pXigual3 = 0;
figure;
plot(x,fdpX);
hold on;
stem(dato,normpdf(dato,mu,sigma));
title(['El valor de P(X = ',num2str(dato),') (área bajo la curva del intervalo) es
', num2str(pXigual3)]);
hold off;
```

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mu= str2num(get(handles.edit1,'string'));
sigma= str2num(get(handles.edit2,'string'));
xmin = mu - 3*sigma;
xmax = mu + 3*sigma;
x = linspace(xmin,xmax, 200);
fdpX = normpdf(x,mu,sigma);
dato = str2num(get(handles.edit5,'string'));
pXmenorigual3 = normcdf(dato,mu,sigma);
xarea=linspace(dato,xmax);
figure;
plot(x,fdpX);
hold on;
ttt=1-pXmenorigual3;
area(xarea,normpdf(xarea,mu,sigma));
title(['El valor de P(X >= ',num2str(dato),') (área bajo la curva del intervalo)
es ', num2str(ttt)]);
hold off;
```

```
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
mu= str2num(get(handles.edit1,'string'));
sigma= str2num(get(handles.edit2,'string'));
xmin = mu - 3*sigma;
xmax = mu + 3*sigma;
x = linspace(xmin,xmax, 200);
fdpX = normpdf(x,mu,sigma);
dato = str2num(get(handles.edit6,'string'));
```

```
dato2 = str2num(get(handles.edit7,'string'));
pXmenorigual3 = normcdf(dato2,mu,sigma)-normcdf(dato,mu,sigma);
xarea=linspace(dato,dato2);
figure;
plot(x,fdpX);
hold on;
area(xarea,normpdf(xarea,mu,sigma));
title(['El valor de P(',num2str(dato),'<= X <= ',num2str(dato2),') (área bajo la
curva del intervalo) es ', num2str(pXmenorigual3)]);
hold off;
```

ANEXO VIII: CÓDIGO CALCULADORA DISTRIBUCIONES DISCRETAS

```
function varargout = calculadora43(varargin)
% CALCULADORA43 MATLAB code for calculadora43.fig
%      CALCULADORA43, by itself, creates a new CALCULADORA43 or raises the
existing
%      singleton*.
%
%      H = CALCULADORA43 returns the handle to a new CALCULADORA43 or the handle
to
%      the existing singleton*.
%
%      CALCULADORA43('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CALCULADORA43.M with the given input arguments.
%
%      CALCULADORA43('Property','Value',...) creates a new CALCULADORA43 or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before calculadora43_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to calculadora43_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help calculadora43

% Last Modified by GUIDE v2.5 13-Apr-2016 08:42:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @calculadora43_OpeningFcn, ...
                   'gui_OutputFcn',  @calculadora43_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before calculadora43 is made visible.
function calculadora43_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to calculadora43 (see VARARGIN)

% Choose default command line output for calculadora43
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes calculadora43 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = calculadora43_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im=imread('numeros.jpg');
axes(handles.pantalla);
imshow(im);


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema3;
close(handles.figure1);


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x= get(handles.edit1,'string');
if isempty(x)
    msgbox('Uno de los parámetros  principales está vacío');
    return;
end;
x = str2num(x);
if size(x,1)>1 || size(x,2)>1
   errordlg('Introduzca un único valor','ERROR');
   return;
else
    if size(x) == [0 0]
        errordlg('El valor debe ser numérico','ERROR');
    end;
end;
a=get(handles.edit2,'string');
b=get(handles.edit3,'string');
c=get(handles.edit4,'string');
d=get(handles.edit5,'string');
e=get(handles.edit6,'string');
if isempty(a),
    set(handles.edit7,'string',' ');
else
    a = str2num(a);
    if isempty(a);
        errordlg('El valor debe ser numérico','ERROR');
    else
    A=poisspdf(a,x)*100;
set(handles.edit7,'string',A);
    end
end;
if isempty(b),
    set(handles.edit8,'string',' ');
else
    b = str2num(b);
    if isempty(b)
        errordlg('El valor debe ser numérico','ERROR');
    else
    B=poisscdf(b,x)*100;
set(handles.edit8,'string',B);
    end
end;
if isempty(c),
    set(handles.edit9,'string',' ');
else
    c = str2num(c);
    if isempty(c)
        errordlg('El valor debe ser numérico','ERROR');
    else
    aaa=1;
bbb=poisscdf((c-1),x);
C=(aaa-bbb)*100;
set(handles.edit9,'string',C);
    end
end;
if isempty(d) || isempty(e)
   set(handles.edit10,'string',' ');
    else
```

```
    d = str2num(d);
    e = str2num(e);
    if isempty(d)||isempty(e)
        errordlg('El valor debe ser numérico','ERROR');
    else
    if e>d
ccc=poisscdf(e,x);
ddd=poisscdf(d,x);
D=(ccc-ddd)*100;
set(handles.edit10,'string',D);
    end
    end
if d>e
    msgbox('Intervalo mal definido');
    D= 'ERROR';
    set(handles.edit10,'string',D);
end
if d==e
   D=0;
    set(handles.edit10,'string',D);
end
end;
if size(a,1)>1 ||size(a,2)>1 || size(b,1)>1 ||size(b,2)>1 || size(c,1)>1
||size(c,2)>1 || size(d,1)>1 ||size(d,2)>1 || size(e,1)>1 ||size(e,2)>1
     errordlg('Introduzca un único valor','ERROR');
     return;
end;


function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=str2num(get(handles.edit1,'string'));
a=str2num(get(handles.edit2,'string'));
r=0:x+5;
fmpx=poisspdf(r,x);
ttt=poisspdf(a,x);
figure;
stem(r,fmpx);
hold on
xlim([-2 (x+10)]);
ylim([0 (0.3)])
stem((a),poisspdf((a),x));
title(['El valor de P(X = ',num2str(a),') (Probabilidad en rojo) es ',
num2str(ttt)]);
hold off;

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=str2num(get(handles.edit1,'string'));
a=str2num(get(handles.edit3,'string'));
r=0:x+5;
fmpx=poisspdf(r,x);
ttt=poisscdf(a,x);
figure;
stem(r,fmpx);
hold on
xlim([-2 (x+10)]);
ylim([0 (0.3)])
stem((0:a),poisspdf((0:a),x));
title(['El valor de P(X <= ',num2str(a),') (Suma de las probabilidades en rojo) es
', num2str(ttt)]);
hold off;

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=str2num(get(handles.edit1,'string'));
a=str2num(get(handles.edit4,'string'));
r=0:x+5;
fmpx=poisspdf(r,x);
ttt=1-poisscdf((a-1),x);
figure;
stem(r,fmpx);
hold on
xlim([-2 (x+10)]);
ylim([0 (0.3)])
stem((a:x+8),poisspdf((a:x+8),x));
title(['El valor de P(X >= ',num2str(a),') (Suma de las probabilidades en rojo) es
', num2str(ttt)]);
hold off;
```

```
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=str2num(get(handles.edit1,'string'));
a=str2num(get(handles.edit5,'string'));
b=str2num(get(handles.edit6,'string'));
r=0:x+5;
fmpx=poisspdf(r,x);
ttt=poisscdf(b,x)-poisscdf(a,x);
figure;
stem(r,fmpx);
hold on
xlim([-2 (x+10)]);
ylim([0 (0.3)])
stem((a:b),poisspdf((a:b),x));
title(['El valor de P(',num2str(a),'<= X <= ',num2str(b),') (La suma de las
probabilidades en rojo) es ', num2str(ttt)]);
hold off;
```

ANEXO X: CÓDIGO CALCULADORA INTERVALOS DE CONFIANZA

```
function varargout = intervalos(varargin)
% INTERVALOS MATLAB code for intervalos.fig
%      INTERVALOS, by itself, creates a new INTERVALOS or raises the existing
%      singleton*.
%
%      H = INTERVALOS returns the handle to a new INTERVALOS or the handle to
%      the existing singleton*.
%
%      INTERVALOS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INTERVALOS.M with the given input arguments.
%
%      INTERVALOS('Property','Value',...) creates a new INTERVALOS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before intervalos_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to intervalos_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help intervalos

% Last Modified by GUIDE v2.5 14-Apr-2016 09:05:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @intervalos_OpeningFcn, ...
                   'gui_OutputFcn',  @intervalos_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before intervalos is made visible.
function intervalos_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to intervalos (see VARARGIN)
```

```
% Choose default command line output for intervalos
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes intervalos wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = intervalos_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
im=imread('numeros.jpg');
axes(handles.pantalla);
imshow(im);


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x=(get(handles.edit1,'string'));
y=(get(handles.edit2,'string'));
z=(get(handles.edit7,'string'));
h=(get(handles.edit8,'string'));
if isempty(x)||isempty(y) || isempty(z)||isempty(h)
    msgbox('Uno de los parámetros que se necesitan está vacío');
```

```
    return;
end
    x=str2num(x);
    y=str2num(y);
    z=str2num(z);
    h=str2num(h);
    if size(x,1)>1 || size(x,2)>1 || size(y,1)>1 || size(y,2)>1 || size(h,1)>1 ||
size(h,2)>1 || size(z,1)>1 || size(z,2)>1
   errordlg('Introduzca un único valor','ERROR');
   return;
else
    if size(x) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
    end;
    if size(y) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
    end;
    if size(z) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
    end;
    if size(h) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
    end;
end;
    U=x-h*y/(sqrt(z));
  set(handles.edit3,'string',U);
T=x+h*y/(sqrt(z));
  set(handles.edit4,'string',T);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
w=(get(handles.edit2,'string'));
y=(get(handles.edit5,'string'));
h=(get(handles.edit8,'string'));
if isempty(w)||isempty(y) || isempty(h)
    msgbox('Uno de los parámetros que se necesitan está vacío');
    return;
end
    w=str2num(w);
    y=str2num(y);
    h=str2num(h);
    if size(w,1)>1 || size(w,2)>1 || size(y,1)>1 || size(y,2)>1 || size(h,1)>1 ||
size(h,2)>1
   errordlg('Introduzca un único valor','ERROR');
   return;
   else
    if size(h) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
    end;
    if size(y) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
    end;
    if size(w) == [0 0]
```

```
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
    end;
    end
N=(2*h*w/y)^2;
  set(handles.edit6,'string',N);

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tema4;
close(handles.figure1);


function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
p=(get(handles.edit11,'string'));
z=(get(handles.edit8,'string'));
n=(get(handles.edit7,'string'));
if isempty(p) || isempty(z) || isempty(n)
    errordlg('Uno de los parámetros que se necesitan está vacío');
```

```
    return
end
p=str2num(p);
z=str2num(z);
n=str2num(n);
if p>1 || p<0
    errordlg('Las probabilidades son siempre entre 1 y 0');
    return
end
if size(p,1)>1 || size(p,2)>1 || size(z,1)>1 || size(z,2)>1 || size(n,1)>1 ||
size(n,2)>1
   errordlg('Introduzca un único valor','ERROR');
   return;
   else
    if size(p) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
        set(handles.edit9,'string','');
        set(handles.edit10,'string','');
        return
    end;
    if size(z) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
        set(handles.edit9,'string','');
        set(handles.edit10,'string','');
        return
    end
    if size(n) == [0 0]
        errordlg('Uno de los valores necesarios introducidos no es
numerico','ERROR');
        set(handles.edit9,'string','');
        set(handles.edit10,'string','');
        return
    end;
    end
a=1-p;
raiz=(p*a)/n;
answer=p-z*sqrt(raiz);
answer1=p+z*sqrt(raiz);
set(handles.edit9,'string',answer);
set(handles.edit10,'string',answer1);

function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end
```