Sección de Matemáticas
Universidad de La Laguna

Ana María Godoy Orozco

Herramienta para el análisis y la experimentación con datos cuantitativos

Tool for analysis and experimentation with quantitative data

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, julio de 2019

Dirigido por
Gara Miranda Valladares

Gara Miranda Valladares
Departamento de Ingeniería
Informática y de Sistemas
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Detrás de todo logro, siempre hay personas que han estado ahí y han contribuido a que se lleve a cabo.

En primer lugar, agradecer a mi tutora Gara Miranda Valladares por haber sido mi guía, por su apoyo y su valiosa ayuda en esta etapa que nos ha unido. Definitivamente me ha brindado las herramientas necesarias para completar mi Trabajo de Fin de Grado satisfactoriamente.

No podía olvidarme de mis amigos, que han hecho todo el camino más fácil y haberme dejado aprender un poquito de todos ustedes. En especial a Rubén, Sandra y Elena, por llegar y hacer que nada fuera igual que antes, y hacer que fuera mejor. Qué bonito es ser yo con ustedes, aunque ya no compartamos la misma dirección.

A mis padres, a mi hermana y a toda mi familia, por enseñarme a trabajar para conseguir, a no rendirme y aprender de mis errores y por su indispensable e inmenso apoyo, sin el cual no hubiera podido llegar hasta aquí.

Ana María Godoy Orozco
La Laguna, 8 de julio de 2019

Resumen · Abstract

Resumen

El trabajo consiste principalmente en diseñar una herramienta visual que sea capaz de realizar análisis estadístico de datos cuantitativos haciendo uso del lenguaje de programación Python y diversas librerías.

Para ello, hemos trabajado en varias tareas simultáneamente. En primer lugar, creamos las ventanas gráficas necesarias para la aplicación haciendo uso del paquete Tkinter y, en relación al análisis estadístico, buscábamos que la aplicación realizara los siguientes estudios: comparar datos cuantitativos continuos y discretos y predecir para resumir la relación entre variables. El usuario introduce los datos que desea estudiar y elige las columnas y el análisis a realizar. En la herramienta esto está implantado mediante tests estadísticos y funciones de las que se obtienen las tablas con los resultados requeridos para lo cual hemos utilizado los módulos Stats de Scipy, Statsmodels y algunos comandos específicos de Pandas. A la hora de elegir los tests distinguimos entre el número de grupos a estudiar, el tamaño del menor de éstos, el tipo de dato y el tamaño de la muestra. Debido a la estructura para el uso de un test frente a otro y a las librerías de estadística utilizadas hemos logrado una herramienta cuyo desenlace fue el esperado.

La aplicación que hemos conseguido desarrollar, a la cual hemos denominado T-Qube, es apta para realizar comparaciones de datos cuantitativos discretos y continuos y predicciones sobre los datos. Los resultados que aportamos son verosímiles, sin embargo, se mencionan también algunos aspectos a mejorar e ideas para proseguir con el trabajo.

Palabras clave: *Python – Scipy – Tkinter – Statsmodels – Pandas – – T-Qube*

Abstract

The work consists mainly of designing a visual tool that is able of performing statistical analysis of quantitative data using the programming language Python and some libraries.

For this, we have worked on several tasks simultaneously. First, we created the necessary graphic windows for the application using the package Tkinter and, in relation to the statistical analysis, we looked for the application to carry out the following studies: compare continuous and discrete quantitative data and predict according to the population. The user introduces the data that he wants to study and chooses the columns and the analysis to be performed. In the tool, this is implemented through statistical tests and functions from which the tables with the required results are obtained, for which we have used the modules Stats from Scipy, Statsmodels and some specific commands from Pandas. When choosing the tests we pick out between the number of groups to study, the size of the smallest of these, the type of data and the size of the sample. Due to the structure for the use of one test against another and to the statistical libraries used, we have achieved a tool whose outcome was as expected.

The application that we have been able to develop, which we have called T-Qube, is apt to make comparisons of discrete and continuous quantitative data and predictions about the data. The results we provide are credible, however, some aspects to improve and ideas to continue with the assignment are also mentioned.

Keywords: *Python – Scipy – Tkinter – Statsmodels – Pandas – T-Qube*

Contenido

Introducción

Como dijo S. Gudder, *"La esencia de las matemáticas no es hacer las cosas simples complicadas, sino hacer las cosas complicadas, simples"*. Tomando esta característica de las matemáticas y la necesidad de innovar y avanzar en las tecnologías aparece el hecho de realizar una serie de programas o aplicaciones que faciliten el proceso a los usuarios. De estas ideas nace el objetivo principal del trabajo: diseñar una aplicación de uso sencillo y visual para el análisis de datos estadístico que facilite al usuario realizar un estudio estadístico sobre la muestra que proponga. Para lograrlo, como herramienta principal vamos a hacer uso del lenguaje de programación *Python*.

Python es un lenguaje de programación con 30 años de antigüedad que es conocido por su versatilidad multiplataforma y multiparadigma y que, además, la licencia de código abierto permite su uso sin la necesidad de pagar por ello. Su principal objetivo es la automatización de procesos para ahorrar tanto complicaciones como tiempo. A su vez, *Python* también es ideal para trabajar con grandes volúmenes de datos ya que favorece su extracción y procesamiento. También posee una amplia biblioteca de recursos con especial énfasis en las matemáticas. Como vamos a trabajar con una gran cantidad de datos y realizar estudios estadísticos sobre ellos, estas dos últimas cualidades nombradas son una de las razones que nos llevan a hacer uso de *Python* para llevar a cabo este trabajo. Del mismo modo, su funcionalidad y el gran impacto que tiene sobre su posible uso en el futuro, son también otras causas que nos impulsan a usar este lenguaje.

Para poder llevar a cabo la herramienta lo primero que hace falta es, al recibir los datos, saber qué estudio estadístico se quiere obtener de ellos. Por ello, en el primer capítulo quedan redactados y esquematizados los pasos a seguir y cada test estadístico a elegir según las características de los datos que se manejen. Este capítulo está dividido principalmente en tres secciones en las que se trata el proceso de los datos obtenidos, las características de los mismos y,

finalmente, analizar los tests estadísticos posibles para éstos. En esta última sección distinguiremos entre tres vías distintas que se elegirán según el estudio que se quiera realizar sobre los datos: describirlos, compararlos o predecir para resumir la relación entre variables.

Seguidamente introducimos en el trabajo el segundo capítulo, en el cual abordamos el entorno de desarrollo de *Python* que utilizamos para conseguir la aplicación: PyCharm. En el mismo se comentan las librerías principales que se han utilizado a lo largo del trabajo para lograr el objetivo: *Scipy* y *Tkinter*.

A continuación, en el tercer capítulo se exponen las características de la aplicación y se realiza, mediante un fichero de ejemplo propuesto en el primer capítulo, una prueba de la herramienta. Aquí se encontrará, a modo de guía de usuario, todos los comandos utilizados para el análisis estadístico, así como el tratamiento de los datos para poder aplicarlos.

Para finalizar la memoria, se encontrará un cuarto capítulo en el cual se redactan las conclusiones obtenidas al finalizar el trabajo, se enumeran una serie de trabajos para seguir desarrollando la aplicación y una breve valoración personal.

1

Análisis de datos cuantitativos

1.1. Proceso de análisis de los datos obtenidos experimentalmente

Una de las partes más importantes de este trabajo es el uso de tests estadísticos sobre los datos. Para ayudar a elegir qué tipo de análisis de datos cuantitativos usar después de la recopilación de datos es importante conocer la siguiente información sobre el proyecto:

- ¿Cuáles son tus preguntas específicas sobre la investgación?
- ¿Qué tipo de datos quieres recoger?
- ¿Cuál es el tamaño de la muestra y los grupos?
- ¿Cuáles son las variables dependientes e independientes?

A la hora de escoger el análisis que queremos utilizar tendremos que tener en cuenta el tipo de datos, el tamaño de la muestra, la distribución de estos y más propiedades de las variables y las observaciones que se nombrarán en el caso requerido. La elección se divide según lo que queramos saber acerca de los datos: **describirlos**, **compararlos** o **predecir** para resumir la relación entre variables.

1.2. Características de los datos a analizar

En relación a los datos a introducir, como nuestra aplicación va a consistir generalmente en comparar y predecir el comportamiento de dichos datos agrupados, éstos han de ser cuantitativos, tanto discretos como continuos.

Vendrán dados en un fichero formato *.csv* que se leerá y se obtendrán los valores escritos en el fichero en forma de tabla donde se distribuirán en grupos y subgrupos.

F_1_M	F_1_H	F_4_M	F_4_H	G_1_M	G_1_H	G_4_M	G_4_H	E_1_M	E_1_H	E_4_M	E_4_H
28	36	44	59	35	43	33	26	48	41	33	57
26	32	26	48	38	24	43	31	45	43	35	41
47	41	30	33	14	16	33	30	26	36	56	40
47	32	26	43	41	22	21	20	6	30	59	35
48	31	49	55	15	24	5	32	46	40	49	31
32	30	51	33	27	41	31	30	35	33	47	57
36	34	47	53	39	41	34	34	33	30	39	57
34	40	51	56		47		49	35	18	35	57
26	33	32	12		44		35	30	26	41	48
29	35	29			44		41	26	30	45	43
39		2			10			40	39	30	36
28		47			29			26	31	45	34
37		41			31			27	31	34	45
36		31			8			38	37	30	43
27		44			46				44		3
45		47							43		38
33		27							51		35
44		48							48		58
33		35							34		
37		29							42		
40									31		
26									47		
38									52		
47									35		
36											
38											
39											

Figura 1.1. Fichero que contiene los datos con los que trabajaremos.

En la Figura 1.1[(1)] tenemos el fichero de ejemplo que usaremos para poder ver los resultados de nuestra aplicación. En este fichero en concreto tenemos una muestra de datos cuantitativos continuos recogidos sobre una población, la cual en este caso está formada por alumnos de varias carreras de la Universidad de La Laguna. Los datos están distribuidos en 12 grupos donde el título de cada uno representa la carrera (F: filosofía; G: geografía; E: economía), el curso (1: primero y 4: cuarto) y el sexo (M: mujer y H: hombre) de los alumnos que pertenecen a la muestra. Por ejemplo, el grupo *G_1_M* está formado por las mujeres de primero del grado de Geografía.

A la hora tanto de plantear las preguntas específicas para la investigación como de saber cuál/es es/son la/s variable/s dependiente/s e independiente/s, tendremos que preguntar al usuario de la aplicación, según los datos que nos haya introducido, qué quiere saber de éstos. Es ahí donde podremos escoger el análisis a realizar para obtener respuestas sobre el estudio.

En el caso de nuestros datos, plantearemos todos los posibles análisis que éstos permitan utilizar, ya que con este ejemplo queremos comprobar la versatilidad de la aplicación ante unos datos reales. Por ello, señalamos a continuación cuáles son los tests estadísticos a utilizar en cada caso.

(1) Fuente: Quantitative Data Analysis: Choosing a statistical test

1.3. Tests estadísticos a utilizar

Como ya comentábamos en la primera sección de este capítulo, los tests estadísticos que vamos a usar a lo largo del trabajo se escogerán dependiendo de lo que queramos saber sobre los datos: **describirlos**, **compararlos** o **predecir** para resumir la relación entre variables.

1.3.1. Describir la muestra

Cuando hablamos de hacer una descripción sobre los datos que manejamos nos referimos a dar respuesta a la pregunta: *¿Cómo puedo resumir mis datos?* Esto es, al fin y al cabo, realizar una estadística descriptiva sobre los datos (ver Figura 1.2 [(2)]).

Lo primero que tenemos que saber es el tamaño de la muestra. Si ésta es menor de 10 tendremos que realizar una frecuencia de respuestas. En el caso contrario, según el tipo de datos haremos diferentes análisis.

- Si éstos son **cuantitativos discretos** utilizaremos una tabla de frecuencias y porcentajes, la moda, la mediana, los cuartiles y/o percentiles y el rango y/o rango intercuartílico. Podremos representar haciendo uso de diagrama de barras o de sectores.
- Si los datos son del tipo **cuantitativos continuos** en el análisis podemos aplicar la tabla de frecuencias y porcentajes, la moda, la mediana, los cuartiles y/o percentiles, la media, la desviación típica e intervalos de confianza al 95 %. En este caso, para la representación de los datos y resultados usaremos un histograma o un gráfico de línea.

1.3.2. Comparar los grupos de la muestra

En este apartado daremos respuesta a: *¿Cómo difieren los datos entre grupos?*, es decir, veremos tests estadísticos para la comparación de los grupos y veremos si éstos influyen o no en la muestra y si los resultados pueden ser llevados a nivel poblacional.

Para empezar, tenemos que distinguir en dos casos según el tipo de dato que sea la variable dependiente de nuestro análisis.

El primer caso es que la **variable dependiente** sea **cuantitativa discreta** (ver Figura 1.3[(3)]). En primer lugar distinguimos cuál es el grupo de menor tamaño. Si éste es de tamaño menor que 10 y no podemos combinar grupos haremos una tabla bidimensional sin test estadístico; mientras que en el

[(2)] Fuente: Quantitative Data Analysis: Choosing a statistical test
[(3)] Fuente: Quantitative Data Analysis: Choosing a statistical test

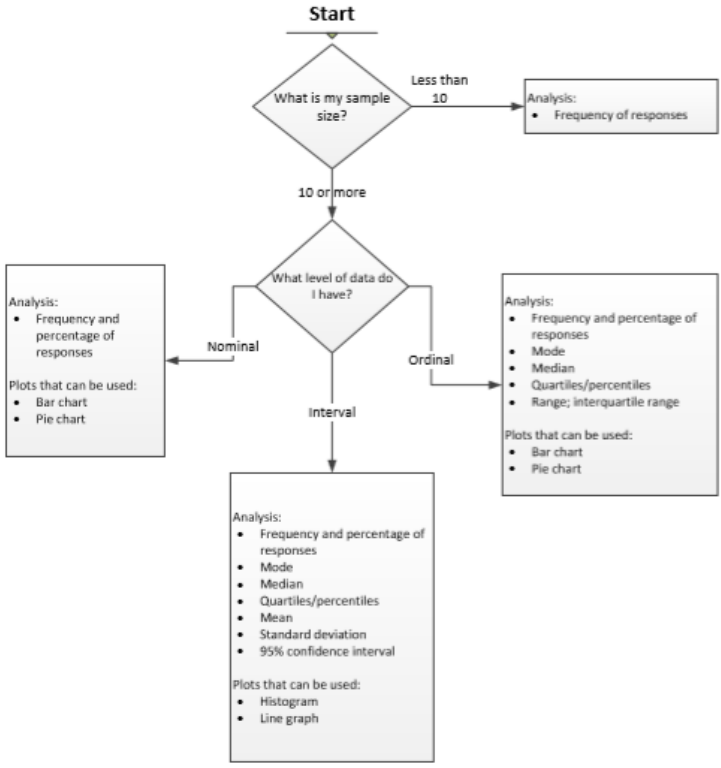


Figura 1.2. Esquema de los pasos para realizar estadística descriptiva

caso de que se puedan combinar grupos, crearemos una nueva variable con esta combinación y realizaremos lo mismo que si tuviéramos un grupo de 10 o más observaciones. A continuación, actuaremos dependiendo del tipo de comparación: entre grupos, dentro de grupos o ambas.

- *Entre grupos:* Si la comparación es entre dos grupos, usamos el test de *Mann-Whitney* que comprueba la heterogeneidad de dos muestras, y si es entre tres o más el test de *Kruskal-Wallis* que prueba si un grupo de datos proviene de la misma población, además, es idéntico al ANOVA pero con los datos reemplazados por categorías a la vez que no asume normalidad, en cambio sí asume que los datos vienen de la misma distribución.
- *Dentro de grupos:* Tenemos que fijarnos en si podemos hacer coincidir las respuestas individuales en todas las variables. Si no podemos, no hay test estadístico para este caso; si podemos y hay dos grupos, haremos el test de *Wilcoxon* que compara el rango medio de dos muestras relacionadas y

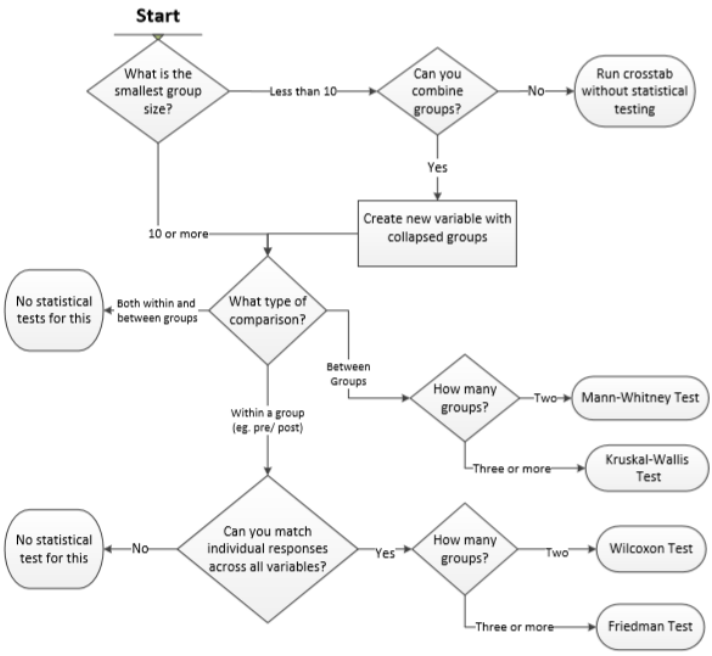


Figura 1.3. Esquema de los pasos para realizar comparación con variable dependiente cuantitativa discreta

determina si existen diferencias entre ellas. Se utiliza como alternativa a la prueba T de Student cuando no se puede suponer la normalidad de dichas muestras. Y si podemos y hay tres o más grupos, hacemos el test de *Friedman* que es equivalente a la prueba ANOVA para medidas repetidas en la versión no paramétrica, el método consiste en ordenar los datos por filas o bloques, reemplazándolos por su respectivo orden.

- *Ambas:* En este caso no hay test estadístico.

Para acabar, el último caso es que la **variable dependiente** sea **cuantitativa continua** (ver Figura 1.4 [(4)]). Como en los anteriores casos, lo primero que tenemos que mirar es el tamaño del grupo más pequeño puesto que si éste es menor que 10 no hay suficientes datos para realizar un test estadístico, entonces pararíamos. Sin embargo, si es de 10 o más nos fijaríamos a continuación en el tamaño de la muestra. En el caso de que sea menor de 100 y no siga una distribución normal, pasaríamos a realizar transformaciones en los datos o usar un análisis no paramétrico. Ahora, si la población está normalmente distribuida

(4) Fuente: Quantitative Data Analysis: Choosing a statistical test

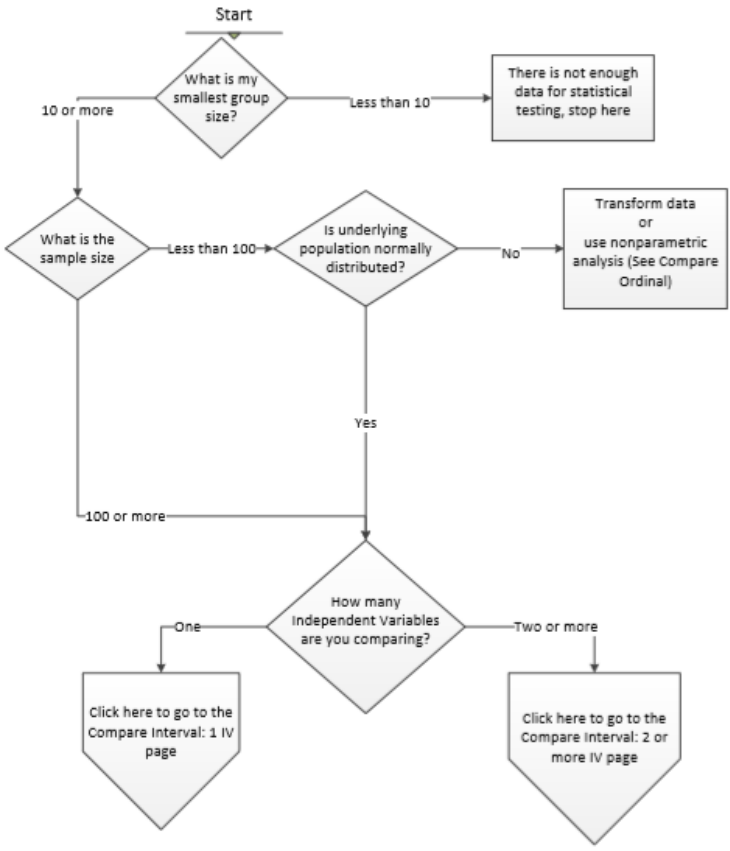


Figura 1.4. Esquema de los pasos para realizar comparación con variable dependiente cuantitativa continua

teniendo menos de 100 datos en la muestra actuaremos del mismo modo que si tuviéramos 100 o más datos, veremos cuántas variables independientes vamos a comparar y en estos casos actuaremos según el tipo de comparación entre ellas.

- *Comparamos una variable independiente y una dependiente:* es necesario tener una variable dependiente cuantitativa continua y una variable independiente cuantitativa continua o cualitativa. Hay tres tipos de comparaciones:
  - *A un valor hipotético o estándar*: Realizar una T de Student de una muestra, que es una prueba paramétrica que permite comprobar si es posible aceptar que la media de la población sea un valor determinado.
  - *Entre grupos:* Si comparamos dos grupos haremos una T de Student de muestras independientes, que compara las medias de dos grupos de casos; mientras que si comparamos tres o más usaremos el ANOVA Unidireccional

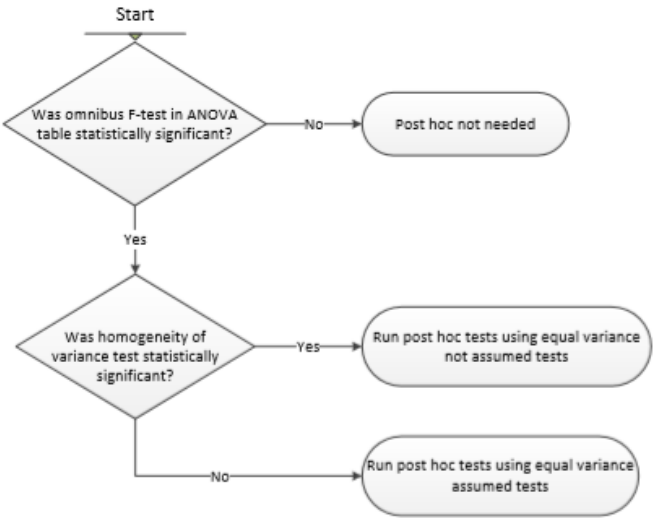


Figura 1.5. Esquema de los pasos para realizar en el ANOVA Unidireccional

como vemos en la Figura 1.5 [(5)]. Este método requiere tener una variable cuantitativa continua dependiente y una variable cualitativa independiente con 3 o más grupos. Recordar que se han de cumplir los supuestos para la comparación de datos cuantitativos continuos. El ANOVA Unidireccional se realiza con estadística descriptiva, test de homogeneidad de la varianza y estimación del tamaño del efecto.

- *Dentro de grupos:* Si comparamos dos grupos hacemos una T de Student con muestras emparejadas, es similar al otro T-Student pero se usa cuando las muestras son dependientes; esto es, cuando se trata de una única muestra que ha sido evaluada dos veces (muestras repetidas) o cuando las dos muestras han sido emparejadas o apareadas. Mientras, si la comparación es entre tres o más grupos pasamos a realizar el ANOVA Unidireccional de medidas repetidas como vemos en la Figura 1.6 [(6)]. Este método requiere tener una variable cuantitativa continua dependiente con medidas pareadas en todas las repeticiones y una variable cualitativa o cuantitativa discreta independiente con 3 o más medidas repetidas. El ANOVA Unidireccional se realiza con test de esfericidad y estimación del tamaño del efecto.

- *Comparamos dos o más variables independientes con una variable dependiente:* Tres tipos de comparaciones:

(5) Fuente: Quantitative Data Analysis: Choosing a statistical test
(6) Fuente: Quantitative Data Analysis: Choosing a statistical test

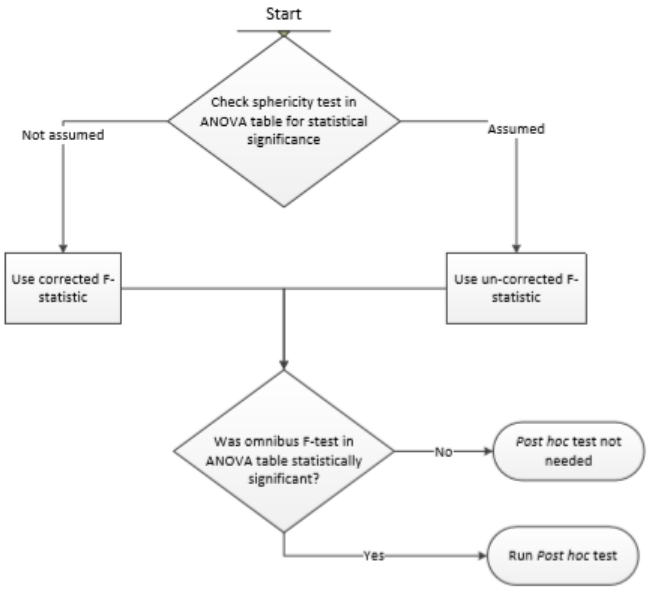


Figura 1.6. Esquema de los pasos para realizar en el ANOVA Unidireccional de medidas repetidas

- *Entre grupos:* Este método requiere tener una variable cuantitativa continua dependiente con medidas pareadas en todas las repeticiones, mínimo 20 observaciones de las variables dependientes por celda de agrupación y dos variables cualitativas o cuantitativas discretas independientes con 2 o más grupos cada una y al menos 20 observaciones de las variables dependientes por celda de agrupación. El ANOVA Bidireccional se realiza con estadística descriptiva, tablas y gráficas para medias marginales y estimación del tamaño del efecto. Interpretamos los resultados como vemos en la Figura 1.7 [(7)].
- *Dentro de grupos:* Este método requiere tener una variable cuantitativa continua dependiente, mínimo 20 observaciones de las variables dependientes por celda de agrupación, una variable cualitativa independiente con 2 o más medidas repetidas y una o más variables cualitativas independientes con 2 o más grupos. El ANOVA de Métodos Mixtos se realiza con estadística descriptiva y test de esfericidad. Interpretamos los resultados como vemos en la Figura 1.8 [(8)].

(7) Fuente: Quantitative Data Analysis: Choosing a statistical test
(8) Fuente: Quantitative Data Analysis: Choosing a statistical test

Figura 1.7. Esquema de los pasos para realizar en el ANOVA Bidireccional

- *Dentro y entre grupos:* Este método requiere tener una variable cuantitativa continua dependiente con medidas pareadas en todas las repeticiones, dos variables cualitativas o cuantitativas discretas independientes con 2 o más medidas repetidas y mínimo 20 observaciones de las variables dependientes por celda de agrupación. El ANOVA Bidireccional de medidas repetidas se realiza con test de esfericidad y estimación del tamaño del efecto. Interpretamos los resultados como vemos en la Figura 1.9 [(9)].

1.3.3. Hacer predicciones basadas en los grupos de la muestra

Por último, a la hora de predecir el comportamiento para resumir la relación entre variables, nos referimos principalmente a responder a la pregunta *¿Cómo puedo resumir la relación entre las variables?*

La respuesta varía según el tamaño de la muestra, pues si éste es menor de 50 no hay suficientes datos para un análisis estadístico, mientras que si hay

(9) Fuente: Quantitative Data Analysis: Choosing a statistical test

Figura 1.8. Esquema de los pasos para realizar en el ANOVA de Métodos Mixtos

50 o más depende del número de variables. Si tenemos dos variables haremos **correlación**, y si tenemos tres o más hacemos **regresión**.

- ***Correlación:*** Distinguimos según el tipo de datos:
  - Cuantitativa discreta: Puede ocurrir que tengamos: ambas variables cuantitativas discretas, una de ellas cuantitativa discreta y la otra cuantitativa continua o ambas variables cuantitativas continuas y no asumimos relación lineal entre ellas. En todos estos casos podemos utilizar los tests de *Rho de Spearman*, que nos da un coeficiente de correlación menos sensible que el de Pearson para los outliers de la muestra, y *tau de Kendall*, que mide el coeficiente de correlación por rangos.
  - Cuantitativa continua: Teniendo que ambas variables son cuantitativas continuas y asumiendo relación lineal entre ellas realizamos la *r de Pearson*, índice que se utiliza para medir el grado de correlación de dos variables cuantitativas continuas.

- ***Regresión:***
  - Variable dependiente cuantitativa discreta: en este caso haremos una *regresión logística ordinal* que se utiliza para modelar la relación entre

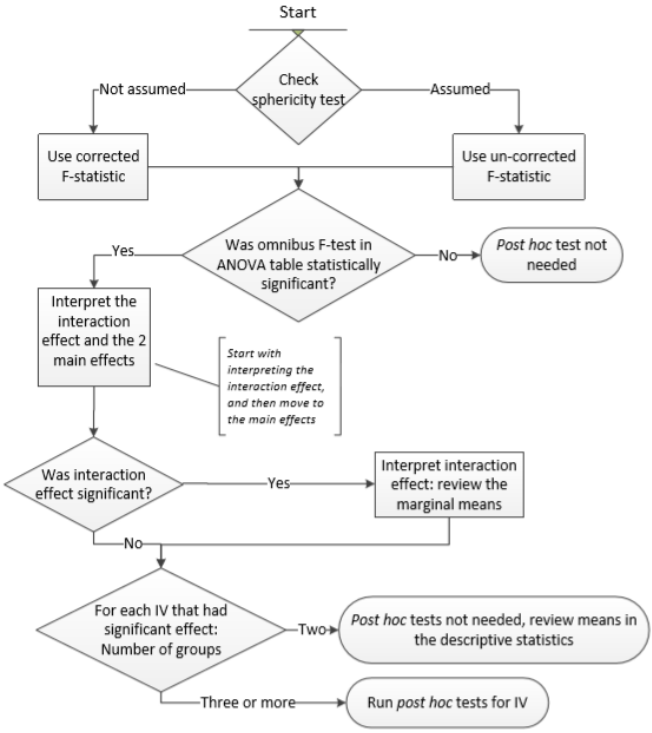


Figura 1.9. Esquema de los pasos para realizar en el ANOVA Bidireccional de medidas repetidas

un conjunto de predictores y una respuesta ordinal, siendo una respuesta ordinal aquella que tiene tres o más resultados que tienen un orden, como por ejemplo bajo, medio y alto.

- Variable dependiente cuantitativa continua: El primer paso es realizar el test de linealidad donde podemos obtener que las hipótesis se cumplen o no. En el caso de que no se cumplan haremos el test de normalidad, test de independencia de variables independientes y test de homocedasticidad. Si las hipótesis de éstos se cumplen y tenemos una única variable independiente haremos *regresión lineal*, usada para aproximar la relación de dependencia entre una variable dependiente, las variables independientes y un término de error aleatorio.
  Por otro lado, si tenemos dos o más variables independientes haremos *regresión lineal múltiple*, analiza la relación entre dos o más variables a través de ecuaciones de la misma manera que la regresión lineal permite trabajar con una variable a nivel de intervalo o razón.

Mientras, si no se cumplen las suposiciones de estos tres últimos tests nos encontramos con varias medidas de corrección que pueden ser elegidas, y habría que hacer una búsqueda específica.
Si como resultado del test de linealidad tenemos que no se cumplen las hipótesis tendremos que saber qué tipo de transformación de los datos es necesaria y de ahí distinguimos en tres casos:

- Sin transformación: debido a que la transformación no es consistente con el modelo teórico o las hipótesis del modelo. Usaremos entonces el método de *regresión no lineal*, método para encontrar un modelo no lineal para la relación entre la variable dependiente y un conjunto de variables independientes.
- Transformación no lineal de la variable dependiente y/o de la/s variable/s independiente/s: Realizaremos test de normalidad, homocedasticidad e independencia de variables independientes y si se cumplen las hipótesis usaremos la *técnica de regresión múltiple adecuada*.
  Y si no se cumplen nos encontramos con varias medidas de corrección que pueden ser elegidas, y habría que hacer una búsqueda específica.
- Añadir interacción y/o términos de orden superior de la variable independiente: Realizaremos test de normalidad, homocedasticidad e independencia de variables independientes y si se cumplen las hipótesis usaremos *regresión múltiple*, que la utilizamos para estudiar la posible relación entre varias variables independientes (predictorias o explicativas) y otra variable dependiente (criterio, explicada, respuesta).
  Y si no se cumplen nos encontramos con varias medidas de corrección que pueden ser elegidas, y habría que hacer una búsqueda específica.

2

Desarrollo de aplicaciones con Python

2.1. Entorno de desarrollo

A la hora de usar *Python*, tenemos distintas formas de hacerlo. Podemos hacer uso de la consola o de intérpretes (o entornos de desarrollo) donde podremos hacer nuestros propios módulos y funciones. En este caso, hemos decidido usar un entorno de desarrollo por la complejidad y el objetivo del trabajo. De todos los intérpretes que tiene *Python* a su servicio, como *Eclipse + PyDev, Visual Studio, PyCharm, Spyder, Atom*, etc., nos decantamos principalmente por elegir entre *Atom* simplemente como editor o *PyCharm* como IDE (Integrated Development Environment).

Aunque *Atom* tiene un amplio soporte en todas las plataformas la compatibilidad entre la compilación y la depuración no están bien integradas. Por lo cual, elegimos *PyCharm*, que aunque la configuración predeterminada puede necesitar varios ajustes para proyectos ya existentes, éste edita, ejecuta y depura *Python* fuera de la caja.

PyCharm (ver Figura 2.1) es uno de los IDEs más conocidos y uno de los mejores dedicado con funciones completas para *Python*. Admite el desarrollo de *Python* directamente fuera de la caja y tiene soporte para proyectos y control de origen. La facilidad de escritura, organización de carpetas y proyectos y, sobre todo, la detección de los errores antes de compilar el código son las principales características de este programa que nos han conducido a hacer uso del mismo.

2.2. Librerías utilizadas

A la hora de escribir el código en *Python*, puede que nos haga falta usar algunas herramientas que no están disponibles en la versión estándar del lenguaje. Para esto sirven las *librerías*, también llamadas *paquetes* o *módulos*.

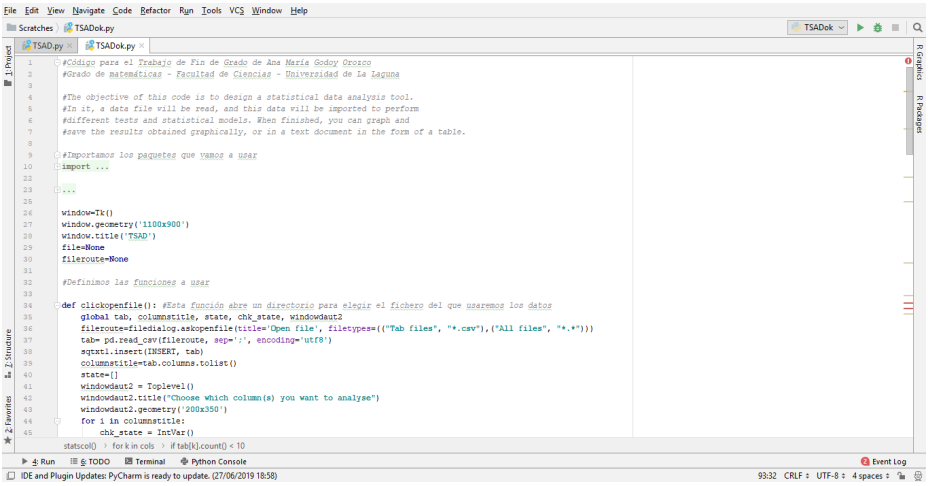


Figura 2.1. Entorno de desarrollo en PyCharm

Los módulos son la forma que tiene *Python* de almacenar definiciones (instrucciones o variables) en un archivo, de forma que se puedan usar después en un script o en una instancia interactiva del intérprete (como en nuestro caso, PyCharm). Así no tenemos que volver a definirlas cada vez. La ventaja principal de que *Python* nos permita separar un programa en módulos es, evidentemente, que podremos reutilizarlos en otros programas o módulos. Para ello, como veremos más adelante, será necesario importar los módulos que se quieran utilizar.

Python viene con una biblioteca de módulos estándar, que es muy amplia y ofrece una gran variedad de módulos que realizan funciones de todo tipo. Sin embargo, dado que el objetivo de nuestro trabajo es realizar una aplicación para el análisis de datos, vamos a necesitar algo más que las librerías de la biblioteca estándar ya que, aunque esta ofrece algunas funciones matemáticas, se nos quedan un poco cortas.

Vamos a hacer uso de dos librerías principales: *Scipy* y *Tkinter*.

Scipy, que es considerado el paquete científico más completo, está compuesto por herramientas y algoritmos matemáticos. Éste incluye interfaces a librerías científicas muy conocidas como las que usaremos a lo largo del trabajo: *Pandas*, *NumPy*, *Matplotlib* o *Stats* (ver Figura 2.2). A continuación, ahondaremos un poco en cada una de ellas, para saber lo que contiene y porqué las vamos a usar.

- **NumPy**: Acrónimo de Numerical Python. Su característica más potente, y por la que usaremos este paquete, es que puede trabajar con matrices (array) de

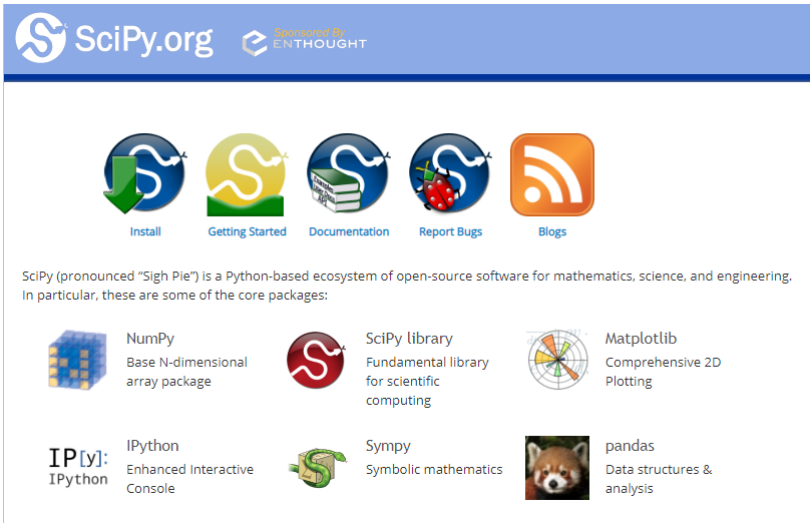


Figura 2.2. Página web de Scipy: scipy.org

ndimensiones

.

- **Matplotlib**: es una librería que permite la realización de gráficos, desde histogramas, hasta gráficos de líneas o mapas de calor. También se pueden agregar expresiones matemáticas a la gráfica. Este paquete nos será de mucha utilidad a la hora de generar los gráficos de nuestras variables o de los resultados que obtengamos.
- **Pandas**: se utiliza para operaciones y manipulaciones de datos estructurados, como tablas o vectores. En este caso, y es muy habitual hacerlo, vamos a usarlo en la fase de depuración y preparación de los datos para tratarlos.
- **Stats**: el paquete del cual obtendremos todas las funciones y tests estadísticos que usaremos para el análisis de datos que ya hemos nombrado en el primer capítulo.

Por otro lado, usaremos el paquete *Tkinter*, la interfaz estándar de *Python* para el kit de herramientas GUI de Tk. Este paquete se utiliza para realizar ventanas gráficas y visualizarlas. Es una librería que no tiene tanto uso en el ámbito matemático, pero como el objetivo de nuestro trabajo es crear una aplicación, es uno de los módulos principales que usaremos en el proyecto.

A su vez, se ha hecho uso de dos librerías específicas para poder proceder con la codificación de los modelos ANOVA y de regresión. Estos módulos son *Statsmodels* y *scikit-learn*.

- **Statsmodels**: proporciona clases y funciones para la estimación de muchos modelos estadísticos diferentes, así como para realizar pruebas estadísticas y la exploración de datos.
- **scikit-learn**: paquete con clases para introducirte al *machine learning*, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Este paquete se ha usado principalmente para aplicar los modelos de regresión lineal, regresión lineal múltiple y regresión logística.

3

T-Qube

En este capítulo, presentaremos las características y resultados que hemos obtenido al compilar el código que genera nuestra herramienta: **T-Qube** (ver Figura 3.1). Nos centraremos en explicar su funcionamiento general y se presentará, a su vez, a modo de guía de usuario.

Figura 3.1. Logo de la aplicación T-Qube.

3.1. Funcionamiento general

Para poder lograr el objetivo principal del trabajo, que recordemos consiste en diseñar una aplicación visual y de uso sencillo de análisis de datos estadísticos que facilite al usuario la realización de un estudio estadístico sobre la muestra que proponga, la herramienta va a seguir el siguiente flujo de trabajo:

- Obtendremos los datos desde un fichero *.csv* utilizando la librería *Pandas*.

- Tomaremos las columnas que el usuario quiere analizar y mediante *Numpy* pasaremos las columnas de la tabla a *ndarrays*.
- Haciendo uso principalmente de algunos comandos que nos proporciona el paquete *Stats* de *Scipy* realizaremos el estudio estadístico que el usuario desee.
- De los resultados obtenidos podremos sacar gráficas, usando *Matplotlib*, y las tablas pertenecientes a los tests estadísticos realizados, utilizando en este caso *Pandas*.
- Por último, permitiremos que el usuario pueda guardar en el directorio los resultados del estudio en formato *.jpg/.png* o *.txt*.

3.2. Diseño de la interfaz

En primer lugar, trataremos el inicio de nuestro código: diseñar la interfaz de la herramienta. Como ya comentamos en el capítulo 2, para llevar a cabo esta tarea hicimos uso de la librería *Tkinter*.

Antes que nada, tenemos que importar los paquetes que vayamos a usar o, a medida que vayamos viendo que los necesitamos, añadirlos a esta sección. Para el desarrollo de la aplicación se han instalado las siguientes librerías:

```
#Paquete para tratar con narrays
import numpy as np
#Paquete científico
import scipy
#Paquete estadístico de Scipy
from scipy import stats
#Paquete de gráficos
from matplotlib import pyplot as plt
#Paquete para trabajar con tablas
import pandas as pd
#Paquete para realizar ventanas de aplicaciones
from tkinter import *
from tkinter.ttk import *
#Paquete para mensajes en pantalla
from tkinter import messagebox
#Paquete para directorio
from tkinter import filedialog
#Paquete para Menú
from tkinter import Menu
#Paquete para cuadros de texto
from tkinter import scrolledtext
#Paquete para los ANOVA
import statsmodels.api as sm
from statsmodels.stats.anova import AnovaRM
```

```
from seaborn import *
#Paquete para modelo de regresión
from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error, r2_score
#Paquete para gráficas en 3D
from mpl_toolkits.mplot3d import Axes3D
import os
```

Para empezar, iniciamos la ventana principal o ventana padre. Para ello usamos el método *Tk()*. A ésta le añadimos algunas configuraciones adicionales. En este caso hemos diseñado las medidas, asignado un título, añadido un borde para los objetos que vamos a implementar en ella y a éste le hemos dado color.

```
#Inicializamos la ventana principal
window=Tk()
#Diseñamos las medidas de la ventana principal
window.geometry('1100x900')
#Asignamos título
window.title('T-Qube')
#Elegimos color para el fondo de la ventana
window.config(bg='powder blue')
#Añadimos borde de 25 píxeles
window.config(bd=25)
```

A continuación, partimos de la idea de crear una barra de menú en la cual vamos a añadir tres elementos y dentro de cada uno habrán opciones. Los ítems que creamos son: *File* que contiene las opciones *Open File* y *Export*, *Statistics* donde realizaremos el análisis de los datos, y *Help* en el que incluimos la licencia de la herramienta. En la Figura 3.3 podemos visualizar las opciones del primer ítem.

Cabe destacar que, al seleccionar cada una de las opciones, cada una nos lleva a diferentes acciones que ya nos detendremos a explicar más adelante.

```
#Construccion del menú en la ventana principal
#Creamos el menú
menu = Menu(window)
#Con este comando creamos items del menú
itemfile=Menu(menu,tearoff=0)
itemstats=Menu(menu, tearoff=0)
itemhelp=Menu(menu, tearoff=0)
#Para abrir un fichero
itemfile.add_command(label='Open file', command=
  clickopenfile)
#Con este comando añadimos un separador a las opciones
itemfile.add_separator()
#Añade una cascada de opciones para los ficheros
menu.add_cascade(label='File', menu=itemfile)
#Añade una cascada de opciones para los cálculos
  estadísticos
menu.add_cascade(label='Statistics', menu=itemstats,command=
  stats)
#Para exportar datos
itemfile.add_command(label='Export', state='normal',
          command=windowsavefile)
#Calcular estadísticas
itemstats.add_command(label='Calculate statistics', state='
  normal')
#Para la licencia de la aplicación
itemhelp.add_command(label='About', command=about)
menu.add_cascade(label='Help', menu=itemhelp)
#Con este comando se añade el menú a la ventana principal
window.config(menu=menu)
```

Para finalizar el diseño de la ventana principal, la hemos dividido en dos subespacios o *Frames*. Además, en cada uno de ellos hemos colocado un cuadro de texto (uno a la izquierda y el otro yuxtapuesto a su derecha) donde introduciremos los datos y resultados que vayamos obteniendo. Los cuadros de texto son otro tipo de *widgets*, a parte de los botones que hemos ido viendo, pertenecientes a la librería *Tkinter*.

```
#Creamos dos partes para la ventana
part1=Frame(window)
part2=Frame(window)
#Posicionamos una a la izquierda y otra a la derecha
part1.pack(expand=YES, fill=BOTH, side='left')
part2.pack(expand=YES, fill=BOTH, side='right')
```

Figura 3.2. Ventana principal de la aplicación.

```
#Añadimos cuadros de texto en los que introduciremos los
  resultados y los datos introducidos
sqtxt1 = scrolledtext.ScrolledText(part1)
sqtxt1.pack(expand=YES, fill=BOTH)
sqtxt2= scrolledtext.ScrolledText(part2)
sqtxt2.pack(expand=YES, fill=BOTH)
```

El comando *.pack()* sirve para posicionar los elementos en la ventana a la que pertenecen. En él usamos *expand=YES* y *fill=BOTH* para que los subespacios rellenen la ventana en ambos ejes y se expanda. Es necesario ponerlo para que aparezcan en la ventana los *widgets*.

Por lo tanto, el resultado que obtenemos al realizar estas acciones es el mostrado en la Figura 3.2.

3.3. Datos a analizar

Una vez creada la ventana principal, se parte de la interfaz mostrada en la Figura 3.2. El primer paso que ha de dar el usuario para hacer uso la aplicación será, en el *Menu*, seleccionar la opción *File* y señalar *Open file*, como podemos ver en la Figura 3.3.

Aquí se abrirá una ventana de directorio en la cual el usuario señalará el fichero que desea analizar. Para conseguir que se abra el directorio del ordenador

Figura 3.3. Ventana principal de la aplicación donde señalamos la función *Open file*.

utilizamos el comando, para el cual ya se ha inicializado la variable como vector vacío.

```
fileroute=[]
fileroute = filedialog.askopenfile(title='Open file',
  filetypes=(('Tab files', '*.csv'), ('All files', '*.*'))
```

El fichero que se selecciona tendrá que tener formato de tabla *.csv*, y así queda señalado en la misma ventana del directorio, donde hemos obligado que sólo se muestren ficheros con dicha extensión además de los de cualquier tipo de forma secundaria (ver Figura 3.4).

En este caso, y como ya señalamos en el primer capítulo (ver Figura 1.1), trataremos con el fichero de datos denominado *RP30_Resultados_TFG_AnaGodoy_v3*.

Al elegir en el directorio dicho documento, lo siguiente que queremos es obtener los datos que contiene. Para ello, hemos usado la librería *Pandas* que recoge los datos en forma de tabla de la siguiente manera:

```
tab= pd.read_csv(fileroute, sep=';', encoding='utf8')
```

Si queremos ahora mostrar los datos del fichero en uno de los cuadros de texto, en este caso en el que posicionamos a la izquierda, usaremos lo siguiente:

```
sqtxt1.insert(INSERT, tab)
```

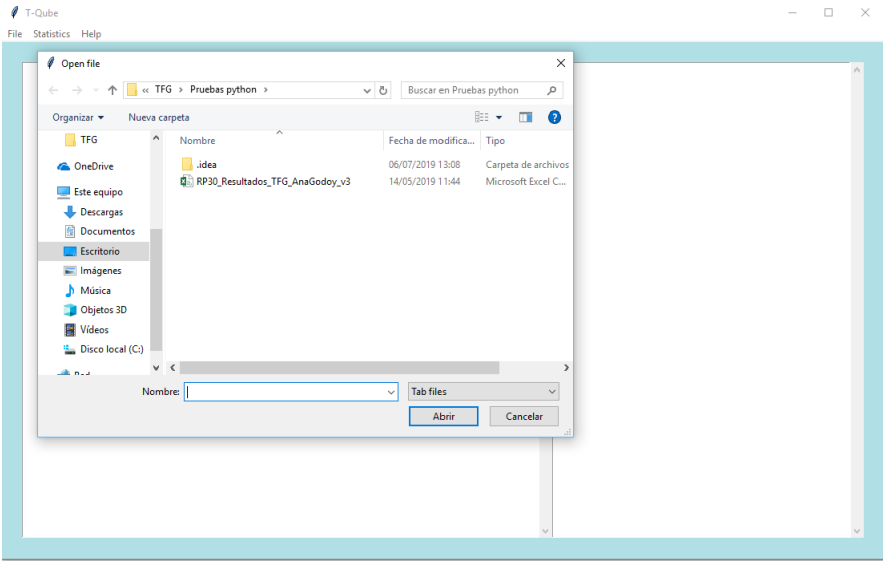


Figura 3.4. Ventana principal de la aplicación donde aparece el directorio para seleccionar el fichero.

En la Figura 3.5 podemos visualizar la tabla de datos obtenida de nuestro fichero de ejemplo. Los valores *NaN* corresponden a elementos vacíos en la tabla, del mismo modo que los puntos suspensivos (. . .) que aparecen en la mitad son las filas que no pueden visualizarse en el cuadro de texto.

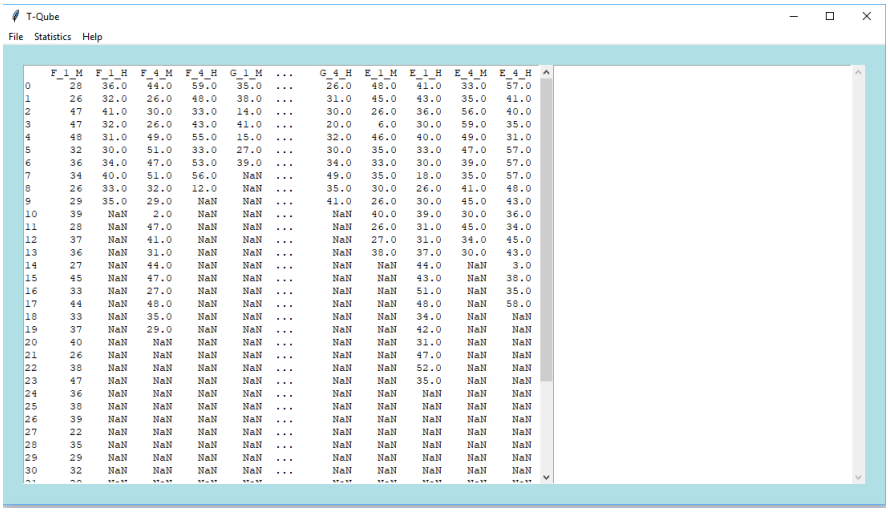
Para finalizar el paso de carga de datos, en el momento que aparecen los datos, vamos a crear una ventana secundaria o *ventana hija* en la que se añadirán los nombres de las columnas que contiene el fichero. Ésta sirve para preguntarle al usuario qué columnas son las que quiere analizar.

Esto lo conseguimos usando principalmente el siguiente comando con el que obtenemos en un vector los nombres de todas las columnas del fichero de carga.

```
columnstitle=tab.columns.tolist()
```

Luego, lo que nos faltaría es crear la ventana secundaria, que lo haremos usando *TopLevel()* y como configuración utilizaremos los mismos métodos que en la ventana principal.

Como no sabemos cuántas columnas va a tener el fichero que se introduzca, hemos hecho un bucle *for* que recorrerá todos los elementos del vector que contiene los nombres de las columnas. Dentro del bucle, añadimos los botones para seleccionar. En este caso hemos usado *CheckButtons*, puesto que queremos



	F_1_M	F_1_H	F_4_M	F_4_H	G_1_M	...	G_4_H	E_1_M	E_1_H	E_4_M	E_4_H
0	28	36.0	44.0	59.0	35.0	...	26.0	48.0	41.0	33.0	57.0
1	26	32.0	26.0	48.0	38.0	...	31.0	45.0	43.0	35.0	41.0
2	47	41.0	30.0	33.0	14.0	...	30.0	26.0	36.0	56.0	40.0
3	47	32.0	26.0	43.0	41.0	...	20.0	6.0	30.0	59.0	35.0
4	48	31.0	49.0	55.0	15.0	...	32.0	46.0	40.0	49.0	31.0
5	32	30.0	51.0	33.0	27.0	...	30.0	35.0	33.0	47.0	57.0
6	36	34.0	47.0	53.0	39.0	...	34.0	33.0	30.0	39.0	57.0
7	34	40.0	51.0	56.0	NaN	...	49.0	35.0	18.0	35.0	57.0
8	26	33.0	32.0	12.0	NaN	...	35.0	30.0	26.0	41.0	48.0
9	29	35.0	29.0	NaN	NaN	...	41.0	26.0	30.0	45.0	43.0
10	39	NaN	2.0	NaN	NaN	...	NaN	40.0	39.0	30.0	36.0
11	28	NaN	47.0	NaN	NaN	...	NaN	26.0	31.0	45.0	34.0
12	37	NaN	41.0	NaN	NaN	...	NaN	27.0	31.0	34.0	45.0
13	36	NaN	31.0	NaN	NaN	...	NaN	38.0	37.0	30.0	43.0
14	27	NaN	44.0	NaN	NaN	...	NaN	NaN	44.0	NaN	3.0
15	45	NaN	47.0	NaN	NaN	...	NaN	NaN	43.0	NaN	38.0
16	33	NaN	27.0	NaN	NaN	...	NaN	NaN	51.0	NaN	35.0
17	44	NaN	48.0	NaN	NaN	...	NaN	NaN	48.0	NaN	58.0
18	33	NaN	35.0	NaN	NaN	...	NaN	NaN	34.0	NaN	NaN
19	37	NaN	29.0	NaN	NaN	...	NaN	NaN	42.0	NaN	NaN
20	40	NaN	NaN	NaN	NaN	...	NaN	NaN	31.0	NaN	NaN
21	26	NaN	NaN	NaN	NaN	...	NaN	NaN	47.0	NaN	NaN
22	38	NaN	NaN	NaN	NaN	...	NaN	NaN	52.0	NaN	NaN
23	47	NaN	NaN	NaN	NaN	...	NaN	NaN	35.0	NaN	NaN
24	36	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
25	38	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
26	39	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
27	22	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
28	35	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
29	29	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
30	32	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN

Figura 3.5. Ventana principal de la aplicación donde aparecen los datos cargados.

que se pueda elegir más de una opción. Los hemos añadido a la ventana secundaria y como texto el nombre de cada columna. Además, creamos un vector en el que se guardarán ceros y unos en cada elemento dependiendo de si el botón es señalado (1) o no (0).

Del mismo modo, se ha adicionado un botón que al clicar sobre él nos desvía hacia la función *stascol*.

```
    state=[]
    windowdaut2 = Toplevel()
    windowdaut2.title('Choose which column(s) you want to
    analyse')
    windowdaut2.geometry('200x350')
    for i in columnstitle:
      chk_state = IntVar()
      chk_state.set(0)
      chk = Checkbutton(windowdaut2, text=i, var=chk_state)
      chk.pack()
      state.append(chk_state)
    botcheck = Button(windowdaut2, text='ANALYSE', command=
    statscol)
    botcheck.pack()
    windowdaut2.mainloop()
```

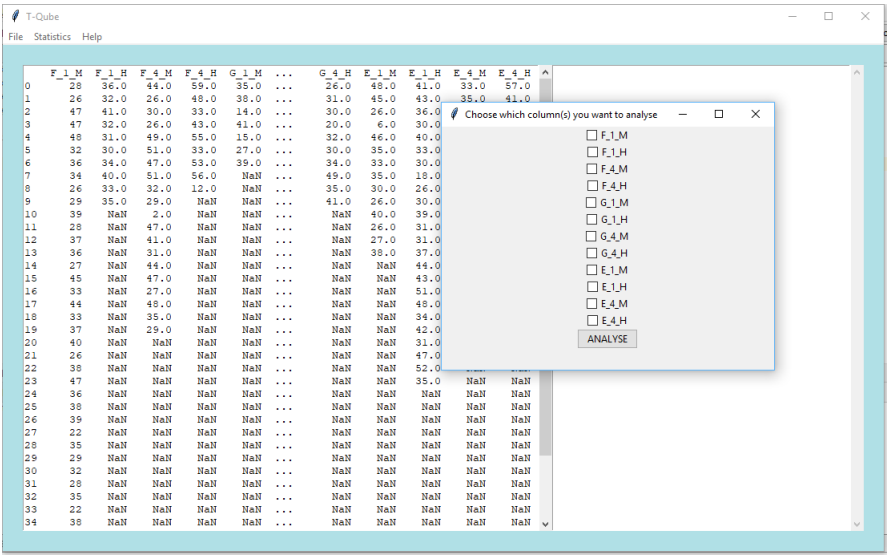


Figura 3.6. Ventana secundaria donde se le pregunta al usuario qué columnas quiere analizar.

Podemos observar en la Figura 3.6 que tenemos 12 columnas. El usuario seleccionará las que desee y pulsará el botón de *Analyse* para proceder con el estudio de las columnas señaladas.

Como hemos comentado, se creó una función *statscol* que es llamada al pulsar el botón de *Analyse* en la ventana secundaria. En esta función obtendremos valores importantes como el tamaño de la muestra, un vector con las columnas seleccionadas y se cuentan los valores del grupo de menor tamaño entre otros.

Lo primero es señalar las variables de la función que queremos que sean globales, esto es que las podamos usar en otras funciones del código. Seguidamente iniciamos las variables que vamos a usar y, con el método *.iconify()* hacemos que la ventana secundaria anterior se minimice. Este método lo usaremos en todas las demás ventanas secundarias que se abran.

Luego, creamos un bucle que recorre desde cero hasta la longitud del vector que contiene los ceros y unos dependiendo de las variables seleccionadas en la ventana anterior, y en él, en la posición que haya un uno, se añadirá un grupo y se guardará el título de la columna a la que corresponde esa posición. Así, tendremos el número de grupos señalados y un vector que contiene las columnas seleccionadas.

```
global cols, small_group, groups, samplesize, vectcol
windowdaut2.iconify()
groups=0
cols=[]
vectcol=[]
vstate=[chk_state.get() for chk_state in state]
for j in range(len(vstate)):
  if vstate[j] == 1:
    groups=groups+1
    cols.append(columnstitle[j])
```

A continuación, si el usuario señala sólo un grupo, muestra una ventana con un mensaje de error, puesto que los análisis que vamos a realizar requieren de 2 o más grupos (ver Figura 3.7). También se guarda en una variable si el grupo de menor tamaño tiene menos de 10 valores (esto se elige así porque el documento de análisis de datos cuantitativos que estamos siguiendo marca esta cifra). Contamos los valores de cada columna seleccionada mediante un bucle y si ya alguna de ellas es menor que 10, el bucle para y queda guardada la variable como *True*. De la misma forma, tenemos una variable para ver si el tamaño de la muestra es mayor o igual a 100 midiendo la longitud de la tabla de datos.

```
if groups==1:
  messagebox.showerror('ERROR', 'You must choose more than
  one column')
  window.destroy()

small_group = False
for k in cols:
  if tab[k].count() < 10:
    small_group = True
    break

samplesize=False
if len(tab)>=100:
  samplesize=True
```

Podemos concluir que en esta función obtenemos las variables que usaremos para la elección de cada test en el momento del estudio.

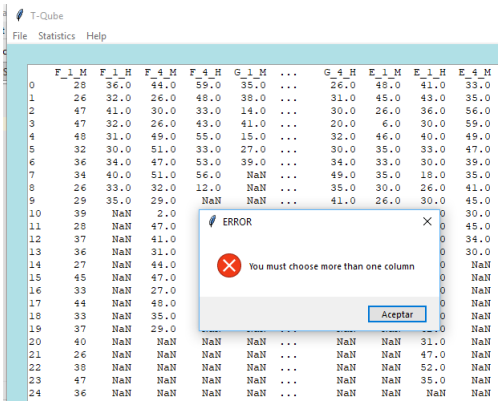


Figura 3.7. Ventana secundaria donde se señala un error en las elecciones hechas.

3.4. Estadística descriptiva

Una vez seleccionadas las columnas, en el cuadro de texto de la derecha se mostrará la estadística descriptiva (media, desviación típica, mínimo, máximo y cuartiles) de las variables seleccionadas.

Queremos guardar los valores de cada columna seleccionada y mostrar la estadística descriptiva de éstas en el segundo cuadro de texto. Entonces realizamos un bucle *for* que recorre el vector de columnas escogidas. Dentro, iremos añadiendo los valores de éstas en listas a un vector, por lo que tendríamos una lista de listas. También insertaremos en el segundo cuadro de texto de la ventana principal la estadística descriptiva (media, desviación típica, mínimo, máximo y cuartiles) sólo de las variables que se eligen, con el comando *.describe()* que nos proporciona la librería *Pandas*.

```
for h in cols:
    vectcol.append((tab[h].values))
    sqtxt2.insert(INSERT, tab[h].describe())
```

En la Figura 3.8 en particular se han señalado las columnas *F_1_M* y *E_4_H*.

Si ahora seleccionamos en el menú la opción *Calculate statistics*, emerge una ventana secundaria en la cual se podrá elegir el tipo de análisis que desee realizar el usuario: *comparar datos cuantitativos discretos*, *comparar datos cuantitaivos continuos* o *predecir el comportamiento* (ver Figura 3.8).

A continuación, distinguiremos en cada uno de los análisis posibles según el número de grupos, el tamaño del grupo menor y el tamaño de la muestra qué

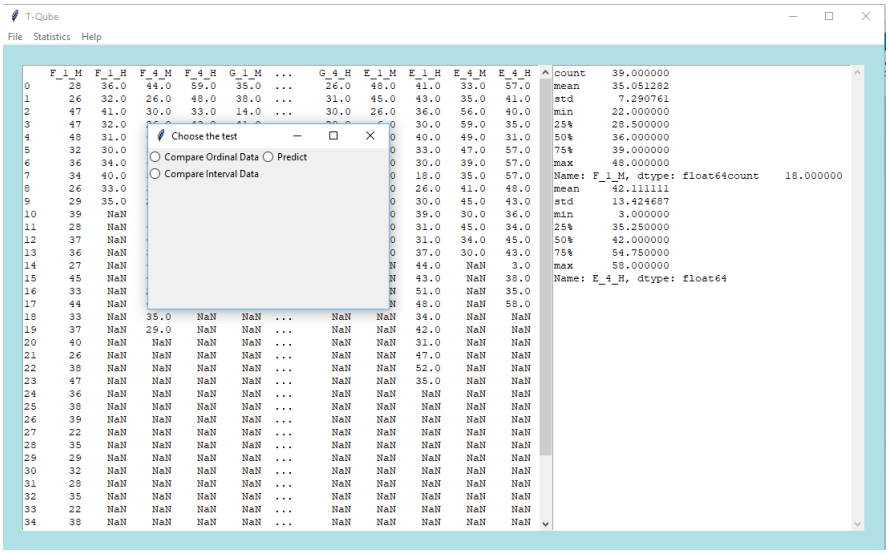


Figura 3.8. Ventana secundaria donde se le pregunta al usuario qué análisis quiere realizar.

tests estadísticos se llevarán a cabo en cada caso y mostraremos un ejemplo de cada uno de ellos.

3.5. Comparar los datos

Primero abordaremos el análisis de comparación que queda dividido en dos estudios: comparación de datos cuantitativos discretos y de datos cuantitativos continuos.

3.5.1. Comparación de datos cuantitativos discretos

Al seleccionar esta opción, si el tamaño del grupo más pequeño es menor que 10, es decir, si la variable es *True*, se muestra por pantalla una ventana informando de que no hay test estadístico en este caso. Este resultado lo obtendremos también varias veces más adelante, en los casos donde los requisitos que se pidan para los datos nos conduzcan a tests que no se han podido implementar o que, simplemente, no se llevan a cabo.

En caso contrario, aparecerá otra ventana secundaria que da a elegir entre *Between Groups* o *Within Groups* (ver Figura 3.9) y se cierran todas las ventanas directamente con el comando *.destroy()*. En este caso, como en la última ventana

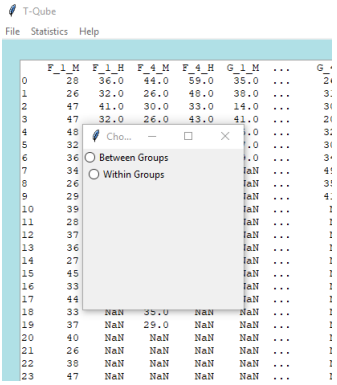


Figura 3.9. Ventana secundaria donde preguntan el tipo de comparación.

nombrada, hacemos uso de *RadioButtons* puesto que sólo queremos que una de las opciones sea seleccionada. Además, al elegir una de ellas, nos llevan hacia respectivas funciones donde se realizan los tests según lo estipulado en el primer capítulo.

```
if small_group:
  messagebox.showinfo('NO TEST', 'There is no statistical
  test to compare this data')
else:
  select = IntVar()
  windowdaut3 = Toplevel()
  windowdaut3.title('Choose the type of comparison')
  windowdaut3.geometry('200x200')
  comp1 = Radiobutton(windowdaut3, text='Between Groups',
  value=1, variable=select, command=comp_ord1)
  comp2 = Radiobutton(windowdaut3, text='Within Groups',
  value=2, variable=select, command=comp_ord2)
  comp1.grid(column=0, row=0)
  comp2.grid(column=0, row=1)
  windowdaut3.mainloop()
```

Cabe comentar también que, al igual que el comando *.pack()*, hemos utilizado en este caso *.grid()* para ubicar exactamente los botones.

Recordemos que en este caso, bajo las restricciones, se hacen cuatro tests estadísticos: Mann-Whitney, Kruskal-Wallis, Wilcoxon y Friedman. Hemos implementado cada uno de ellos con los siguientes comandos de la librería *Stats* de *Scipy*:

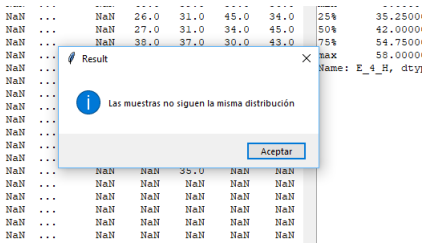


Figura 3.10. Ventana secundaria donde aparece como resultado que no tienen la misma distribución.

```
scipy.stats.mannwhitneyu(x, y, use_continuity=True,
  alternative=None)

#Usamos puntero para los valores a estudiar
scipy.stats.kruskal(*args, **kwargs)

scipy.stats.wilcoxon(x, y=None, zero_method='wilcox',
  correction=False, alternative='two-sided')

#Usamos puntero para los valores a estudiar
scipy.stats.friedmanchisquare(*args)

#x e y son los arrays de los datos a estudiar
```

Estos tests nos devuelven un estadístico y un *p-valor*. Como lo que nos interesa es ver si siguen o no la misma distribución las variables introducidas, a partir del valor de significancia obtenido en el test podemos tener que éste sea menor que un valor *alfa=0.01* o que sea mayor. En ambos casos aparecerá un mensaje informando del resultado: en el primer suceso, aparecerá que no siguen la misma distribución (ver Figura 3.10), mientras que en el segundo, que sí la siguen (ver Figura 3.11).

Además, al seleccionar estos tests, emerge también una ventana con la gráfica de los datos que se han introducido. En ella quedan representados los datos de cada columna señalada inicialmente por el usuario: en el eje X, se muestra el índice al que pertenece; y en el eje Y, el valor de la columna en ese índice. En la Figura 3.12 podemos ver la gráfica que aparece al seleccionar las variables *F_1_M* y *E_4_M* y elegir el test pertinente.

El comando *.plot()* que hemos usado hasta ahora devuelve una lista de instancias de cada dibujo. Una instancia es una referencia a un elemento que creamos, en este caso la línea en gráfica. En el caso de la Figura 3.12 se han seleccionado dos variables, por eso hay dos líneas gráficas. Si el usuario al principio señala tres variables para analizar, habría tres líneas gráficas representadas. Es

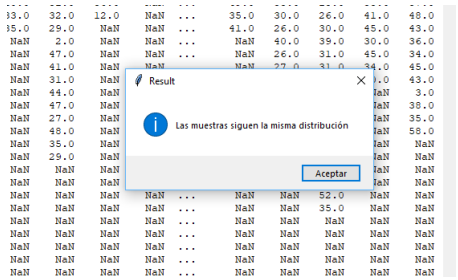


Figura 3.11. Ventana secundaria donde aparece como resultado que tienen la misma distribución.

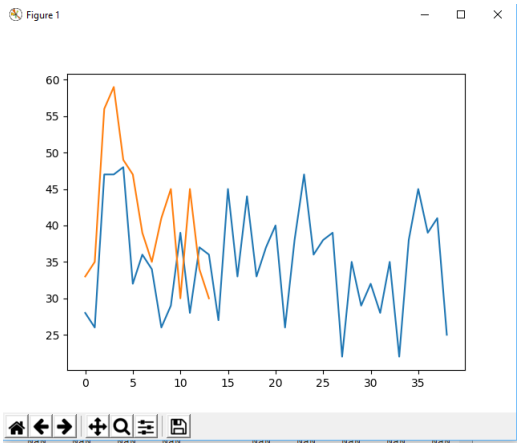


Figura 3.12. Ventana secundaria donde aparece la gráfica de las variables elegidas.

decir, hay tantas líneas gráficas como columnas a analizar escoja el usuario inicialmente.

3.5.2. Comparación de datos cuantitativos continuos

En esta elección, lo primero que vamos a hacer, puesto que lo necesitaremos para realizar algunos tests, es crear un *DataFrame* con los valores de las columnas que han sido elegidas. Esto lo haremos mediante el siguiente código, donde las columnas serán las listas de los valores contenidos en ellas.

```
data={'values': vectcol}
df=pd.DataFrame(data=data, index=[cols])
```

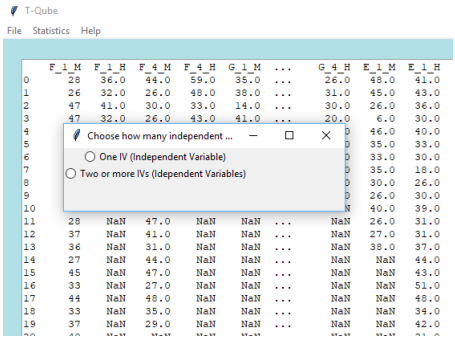


Figura 3.13. Ventana secundaria donde preguntan el número de variables independientes.

El proceso empieza igual que en el caso de datos cuantitativos discretos, si el tamaño de grupo más pequeño es menor que 10, es decir, si la variable es *True*, se muestra por pantalla una ventana informando de que no hay test estadístico.

En caso contrario, si la variable del tamaño de la muestra tiene valor *True* aparece una ventana secundaria para saber si hay una o dos o más variables independientes (ver Figura 3.13).

Por otra parte, si el tamaño muestral es *False*, haremos un test de normalidad a todos los datos proporcionado por el módulo *Stats*.

```
scipy.stats.normaltest(a, axis=0, nan_policy='propagate')
```

El test de normalidad devuelve dos parámetros: un estadístico y un *p-valor*. Si el *p-valor* que obtenemos es menor que el alfa que marcamos anteriormente, emerge una ventana con el mensaje de que hay que usar transformaciones o análisis no paramétrico; si es mayor o igual, se procede del mismo modo que si el tamaño muestral fuera *True*.

Al elegir en la ventana de la Figura 3.13 la primera opción, nos conduce a otra ventana en la cual se pregunta el tipo de comparación: intra grupos o inter grupos (ver Figura 3.9).

Si elegimos la primera, dependiendo del número de grupos, se realiza la prueba T para muestras independientes o la prueba ANOVA Unidireccional; al elegir la segunda, hacemos la prueba T para muestras relacionadas o un ANOVA de medidas repetidas. Los comandos utilizados son los que vemos a continuación. Comentar que, excepto el último de ellos que pertenece al paquete de *Statsmodels*, los demás son comandos de la librería *Stats*.

```
scipy.stats.ttest_ind(a, b, axis=0, equal_var=True,
  nan_policy='propagate')

#Usamos puntero para recorrer el vector de listas
scipy.stats.f_oneway(*args)

scipy.stats.ttest_rel(a, b, axis=0)

#Los valores a y b son arrays de los datos a estudiar

aovrm = statsmodels.stats.anova.AnovaRM(data, depvar,
  subject, within=None, between=None, aggregate_func=None)
#En este caso usaríamos 'within' con una variable
  independiente.
#Para estimar el modelo y compilar la tabla
aovrm.fit()
```

En caso de que escojamos dos o más variables independientes, la herramienta volverá a preguntar qué tipo de comparación quiere aplicar el usuario, en este caso son, intra grupos, donde se procede con un ANOVA Bidireccional; inter grupos, en el cual se usa ANOVA Bidireccional de medidas repetidas; o ambas, para la cual aparece un mensaje puesto que no hemos podido implementar el ANOVA de métodos mixtos.

```
#Tomamos typ=2 porque es ANOVA Bidireccional
aova = sm.stats.anova_lm(*vectcol, typ='II')
aova.fit()

aovrm2way = statsmodels.stats.anova.AnovaRM(data, depvar,
  subject, within=None, between=None, aggregate_func=None)
#En este caso usaríamos 'within' con dos variables
  independientes.
aovrm2way.fit()
```

Mencionar que, en esta opción, si el usuario introduce inicialmente menos de dos variables aparece un mensaje de error en pantalla (ver Figura 3.14) y se cierran todas directamente con el comando *.destroy()*.

Los resultados en forma de tabla que obtenemos en las pruebas ANOVA, se insertan en el cuadro de texto de la derecha en la ventana principal. Además, en la comparación de datos continuos también aparecen las gráficas de los datos nombradas en el apartado anterior como resultado.

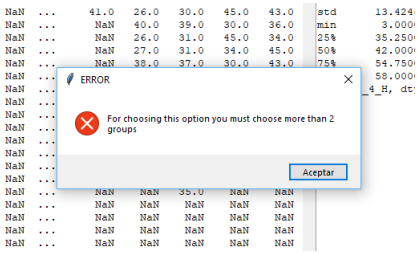


Figura 3.14. Ventana secundaria donde aparece un error por elecciones de variables hechas.

3.6. Predicción

La tercera opción que podemos seleccionar nos lleva a predecir para resumir la relación entre variables. En este caso, se tiene en cuenta si el tamaño de la muestra es menor que 50.

Si esto pasa, se muestra en una ventana que no hay suficientes datos para realizar un estudio estadístico; si no, emerge otra ventana donde el usuario ha de señalar si hay dos o tres o más variables.

Al seleccionar dos, pasamos a la función de correlación; al seleccionar tres o más, entramos en la función de regresión.

3.6.1. Correlación

Si el usuario elige esta opción habiendo señalado más de dos grupos inicialmente, aparece un mensaje de error similar a los ya mostrados.

Lo primero que aparece al tomar este camino es una ventana donde nos preguntan cómo son las variables. En la Figura 3.15 podemos ver los tipos que se aceptan para el estudio de la relación entre ellas. Si se señala la primera, segunda o tercera opción, se realizan los tests de Spearman y Tau de Kendall; si señalamos la cuarta, se hace el test de Pearson. Estos son los comandos pertenecientes a la librería *Stats*.

```
scipy.stats.spearmanr(x, y=None, axis=0)
scipy.stats.kendalltau(x, y, initial_lexsort=None,
  nan_policy='propagate', method='auto')

scipy.stats.pearsonr(x, y)
#x e y son arrays a estudiar
```

Como sólo tenemos dos variables, tomamos como argumentos las listas de posición 0 y 1 en el vector que contiene los datos de las columnas seleccionadas.

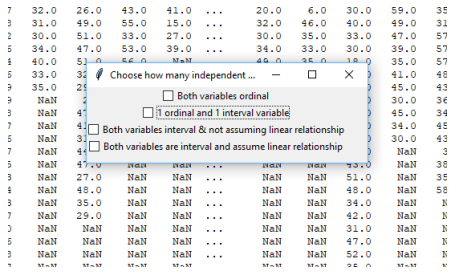


Figura 3.15. Ventana secundaria donde aparecen los tipos de variables a elegir.

Los parámetros que devuelven estas funciones son el coeficiente de correlación y un *p-valor*. Ambos se representan como resultados en el cuadro de texto de la derecha de la ventana principal.

3.6.2. Regresión

Al introducirnos en esta opción, el programa también dará como un error, el ya mencionado, si inicialmente son señaladas menos de tres variables.

Aquí, aparece una ventana donde dividiremos el estudio de regresión. Tendremos que elegir entre variable dependiente ordinal o variable dependiente continua.

Si se escoge la variable dependiente ordinal, se realiza una Regresión Logística. En este modelo obtendremos la bondad de ajuste, con *.score()*, y los valores resultados de aplicarlo a las muestras X e y, que son *arrays* con los datos elegidos para el estudio.

```
model = linear_model.LogisticRegression()
logreg=model.fit(X, y)
#X es la variable independiente e y la variable dependiente
goodness=model.score(X, y)
```

En caso de que se escoja que la variable dependiente sea continua, aparece una ventana donde tenemos que seleccionar si hay una o dos o más variables independientes. Si hay una, se realiza Regresión Lineal; si hay dos o más, Regresión Lineal Múltiple.

Cabe destacar que, en esta opción, aparece un mensaje donde se informa de que las hipótesis de normalidad, linealidad, homocedasticidad e independencia de errores se suponen.

```
#Regresión Lineal

dataX = tab[[var1]]
X = np.array(dataX) #Variable independiente
y = tab[var2].values #Variable dependiente
y[np.isnan(y)] = 0 #Para convertir valores Nan en 0

# Creamos el objeto de Regresión Lineal
regr = linear_model.LinearRegression()
# Entrenamos nuestro modelo
regr.fit(X, y)
# Hacemos las predicciones
y_pred = regr.predict(X)

# Veamos los coeficientes obtenidos, en nuestro caso, serán
  la tangente
print('Coefficients: \n', regr.coef_)
# Este es el valor donde corta el eje Y (en X=0)
print('Independent term: \n', regr.intercept_)
# Error Cuadrado Medio
print('Mean squared error: %.2f' % mean_squared_error(y,
  y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Variance score: %.2f' % r2_score(y, y_pred))
```

Aportamos también al usuario una gráfica de puntos donde los colores señalan los datos de las variables seleccionadas que están por encima y por debajo de la media.

```
colores = ['orange', 'blue']
tamanios = [30, 60]
f1 = tab[var1].values
f2 = tab[var2].values
asignar = []
for index, row in tab.iterrows():
  if (row[var1] > np.mean(tab[var1])):
    asignar.append(colores[0])
  else:
    asignar.append(colores[1])
#Usamos .ion() en vez de .show() para que el programa siga
  ejecutando
plt.ion()
plt.title('Linear Regression')
plt.scatter(f1, f2, c=asignar, s=tamanios[0])
```

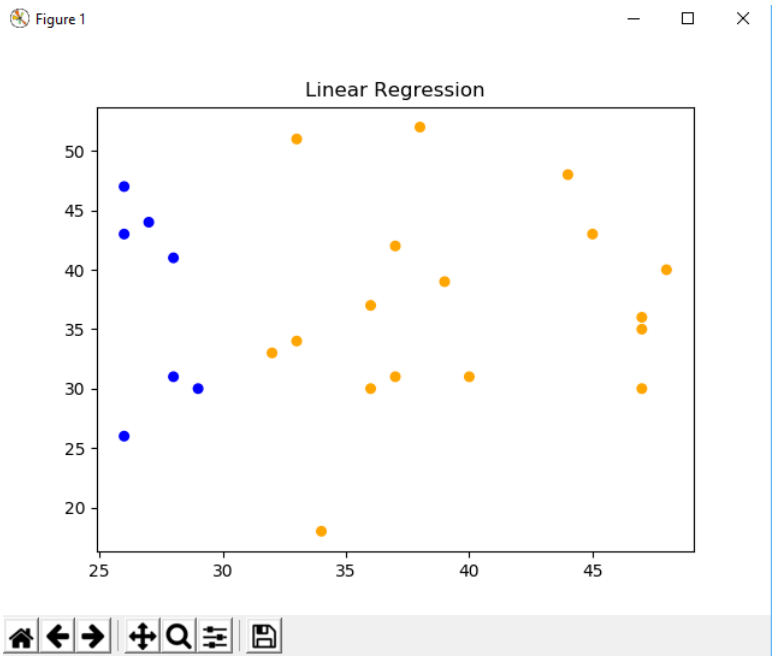


Figura 3.16. Ventana secundaria donde aparece la gráfica de Regresión Lineal.

El resultado que obtenemos es el siguiente representado en la Figura 3.16, en este caso se han seleccionado las variables *F_1_M*, *E_1_H* y *E_4_H*.

```
#Regresión Lineal Múltiple

regr2 = linear_model.LinearRegression()
# Entrenamos el modelo, esta vez, con 2 dimensiones
  obtendremos 2 coeficientes
regr2.fit(XY, z)
# Hacemos la predicción con la que tendremos puntos sobre el
   plano hallado
z_pred = regr2.predict(XY)

# Los coeficientes
print('Coefficients: \n', regr2.coef_)
# Error cuadrático medio
print('Mean squared error: %.2f' % mean_squared_error(z,
  z_pred))
# Evaluamos el puntaje de varianza (siendo 1.0 el mejor
  posible)
```

```
print('Variance score: %.2f' % r2_score(z, z_pred))
```

Donde XY y z son *arrays* formados a partir de los datos que queramos analizar.

3.7. Exportación de resultados

A lo largo de todo el código, vamos obteniendo resultados ya sean en forma de tabla, coeficientes o gráficas. En este apartado vamos a hablar de las distintas formas de guardar esos resultados en un fichero y exportarlos a una carpeta del directorio que deseemos.

Como hemos podido observar en las gráficas que han aparecido (ver Figura 3.12, Figura 3.16), en las mismas ventanas existe un botón para acceder al directorio y poderlas exportar.

Mientras, para guardar las tablas y coeficientes al principio del código se ha abierto un fichero usando *.open()* y, a medida que se van exponiendo en el cuadro de texto de la ventana principal los resultados obtenidos, se van escribiendo en ese fichero también. De esta forma, el usuario tendrá un fichero con todos los resultados de los estudios que ha realizado.

Para que este fichero se guarde donde y con el nombre que el usuario desee, al seleccionar la opción de *Export* en la ventana principal, aparece una ventana de directorio para guardar y nombrar el fichero. Usamos lo siguiente,

```
file =filedialog.asksaveasfile(title='Save file', filetypes
  =(('Text files', '*.txt'), ('All files', '*.*')))
```

De aquí, se toma la ruta en formato *TextIOWrapper* que va a tener el documento con el nombre que le haya asignado el usuario, y de ella, con el atributo *.name* obtenemos justo la ruta deseada.

Luego, como tenemos un fichero con los datos que se ha ido escribiendo mientras se hacía uso de la aplicación y uno nuevo donde el usuario quiere que se guarden los resultados, se procede a renombrar el primer fichero con la ruta del segundo. Para ello, se hace uso de la clase *.rename()* del paquete *os*, a la cual le otorgamos como argumento primero la ruta del fichero a copiar, y luego la ruta del fichero elegido por el usuario.

Para finalizar, se cierran todas las ventanas usando el comando *.destroy()*.

4

Conclusiones, trabajos futuros y valoración personal

Para finalizar este escrito, abordaremos en este capítulo los logros alcanzados con el trabajo, las dificultades que nos hemos encontrado en el desarrollo de este, del mismo modo que haremos mención a líneas de trabajo futuras y una breve valoración personal.

4.1. Conclusiones

Hemos conseguido diseñar una herramienta visual que es capaz de realizar análisis estadístico de datos cuantitativos haciendo uso del lenguaje de programación *Python* y diversas librerías. Las principales son: *Tkinter* para la visualización de las ventanas gráficas, y *Scipy*, *Statsmodels* y algunos comandos específicos de *Pandas* para realizar los tests y análisis estadísticos.

La aplicación que hemos desarrollado es apta para realizar comparaciones de datos cuantitativos discretos y continuos y predicciones para resumir la relación entre variables. En ella hemos aplicado los tests estadísticos pertinentes a cada caso, siendo éste elegido por las características que los datos presentan y lo que el usuario haya escogido en las ventanas emergentes a lo largo del uso de la herramienta. Además, tanto de las tablas que provienen de las pruebas ANOVA, como de las gráficas que aparecen en la comparación de datos y los modelos de regresión, como de los coeficientes o las ventanas de información se obtienen resultados certeros que el usuario puede visualizar en el mismo momento que utiliza la aplicación.

4.2. Trabajos futuros

Partiendo del trabajo que se ha realizado, caben mencionar algunas propuestas para su desarrollo futuro. Se detectan posibles líneas de trabajo en sentido

de extender, progresar y mejorar las tareas que han quedado pendientes. Estas son:

- Ofrecer más opciones para la realización de gráficas, por ejemplo boxplots.
- Poder exportar en formato de tabla (*.csv*) los resultados obtenidos.
- Mejorar el diseño de las ventanas y *widgets*, que actualmente es estándar.
- Añadir los tests estadísticos que no hemos podido desarrollar, como es el caso de las transformaciones y análisis no paramétrico.
- De la misma manera, poder ampliar la aplicación al uso de datos cualitativos.

4.3. Valoración personal

Haciendo un balance general, la aplicación *T-Qube* ha logrado cumplir con los principales requisitos que pretendíamos con la misma: realizar, a grandes rasgos, un estudio estadístico de datos proporcionados por el usuario y que ésta fuera versátil y los resultados obtenidos verosímiles.

Para alcanzar estos objetivos nos hemos encontrado durante el desarrollo del trabajo con varias dificultades:

- Para la creación de las ventanas y sus características hemos tenido que aprender a usar la librería *Tkinter*, para lo que me he valido de los documentos y páginas web citadas en la bibliografía acerca de este módulo.
- La mayoría de tests estadísticos utilizados en el apartado de *Análisis de datos* no han sido aprendidos a lo largo del Grado, por lo que se ha tenido que buscar información para poder aplicarlos con eficacia estadística.
- Del mismo modo que con la librería nombrada en el primer punto, se ha tenido que profundizar en las demás librerías utilizadas en la elaboración de la aplicación, debido a que el nivel de conocimiento previo sobre el lenguaje de programación utilizado, *Python*, no era suficiente para llevar a cabo algunas tareas.

Por último, destacar que la elección del lenguaje de programación *Python* en el trabajo facilita de cara a estudios futuros la comprensión del mismo para poder utilizarlo en las diferentes tareas y asignaturas que se lleven a cabo en los estudios de máster.

Bibliografía

[1] Mokhtar, E. (2018) *Ejemplos De La GUI De Python (Tutorial De Tkinter).* Consultado en https://likegeeks.com/es/ejemplos-de-la-gui-de-python/ Accedido por última vez 07/07/2019.

[2] (2019) *SciPy.org* Consultado en https://www.scipy.org/ Accedido por última vez 06/07/2019.

[3] (2019) *Statistical functions (scipy.stats).* Consultado en https://docs.scipy.org/doc/scipy/reference/stats.html Accedido por última vez 06/07/2019.

[4] Fincher, J. (2018) *Python IDEs and Code Editors (Guide).* Consultado en https://realpython.com/python-ides-code-editors-guide/ Accedido por última vez 18/03/2019.

[5] (2017) *Pandas cookbook.* Consultado en https://github.com/jvns/pandas-cookbook Accedido por última vez 25/05/2019

[6] (2019) *Python 3.7.3 documentation.* Consultado en https://docs.python.org/3.7/ Accedido por última vez 03/07/2019.

[7] (2019) *Tkinter - Python interface to Tcl/Tk.* Consultado en https://docs.python.org/2/library/tkinter.html# Accedido por última vez 06/07/2019.

[8] W. Shipman, J. *Tkinter8.5reference:aGUIfor Python.* En *New Mexico Tech Computer Center* a 31/12/2013.

[9] Love, D. *Python Tkinter By Example.* Publicado el 18/01/2018.

[10] (2015) *Quantitative Data Analysis: Choosing a statistical test.* Consultado en https://www.researchgate.net/profile/Mohamed_Hammad11/post/Which_statistical_analysis_technique_can_be_applied_for_comparing_an_outcome_before_and_after_using_a_technology/attachment/5a18e89b4cde267c3e6e78e5/AS%3A564429348708352%401511581851013/download/Quantitative+Data+Analysis-+Choosing+a+statistical+test.pdf Accedido por última vez 02/07/2019.

[11] Marsja, E. (2019) *Python and R as tools of data analysis.* Consultado en https://www.marsja.se/ Accedido por última vez 04/07/2019.
[12] (2019) *Welcome to Statsmodels Documentation* Consultado en https://www.statsmodels.org/stable/index.html Accedido última vez 06/07/2019.
[13] (2019) *scikit-learn: Machine Learning in Python* Consultado en https://scikit-learn.org/stable/ Accedido por última vez 06/07/2019.
[14] McKinney, W. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.* En *Estados Unidos* a 30/12/2011
[15] Lutz, M. *Learning Python 5TH Edition.* En *Estados Unidos* a 07/06/2013.

Tool for analysis and experimentation with quantitative data

Sección de Matemáticas
Universidad de La Laguna

Ana María Godoy Orozco
Facultad de Ciencias · Sección de Matemáticas
Universidad de La Laguna
alu0100993639@ull.edu.es

1. Abstract

THE assignment consists mainly of designing a visual tool that is able of performing statistical analysis of quantitative data using the programming language Python and some libraries. We have worked on several tasks simultaneously. First, we created the necessary graphic windows using the package *Tkinter* and, in relation to the statistical analysis, we looked for the application to carry out the following studies: compare continuous and discrete quantitative data and predict. The user enters the data that he wants to study and chooses the columns and the analysis to be performed. In the tool, this is implemented through statistical tests and functions for which we have used the modules *Stats* from *Scipy*, *Statsmodels* and some commands from *Pandas*. The application that we have been able to develop, which we have called *T-Qube*, is apt to make the statistical analysis we were looking for. The results we provide are credible, however, some aspects to improve and ideas to continue with the assignment are also mentioned.

2. Introduction

FROM the idea in the mathematics of making complicated things simpler and the need to innovate and advance in technologies appears the fact of making programs or applications that facilitate the process to the user. This is where the main objective of the work was born: design an application of simple and visual use for the statistical data analysis that facilitates the user to carry out a statistical study on the sample that he proposes. To achieve this, as a main tool we will make use of the Python programming language.
Python is a programming language that is known for its multiplatform and multiparadigm versatility and, in addition, the open source license allows its use without the need to pay for it. Its main objective is the automation of processes to save both complications and time. In the same way, its functionality and the great impact it has on its possible use in the future, are some of the causes that make us to use this language.

3. Quantitative data analysis

ONE of the most important parts of this work is the use of statistical tests on data. When choosing the analysis that we want to use we will have to take into account the type of data, the size of the sample, the distribution of these and more properties of the variables and the observations that will be named in the required case. The choice is divided according to what we want to know about the data: textbf describe them, textbf compare them or textbf predict to summarize the relationship between variables.

4. Application development with Pyhton

WHEN using Python, we have different ways of doing it. We can use the console or the interpreters where we can make our own modules and functions. In this case, we have opted to use a IDE because of the complexity and purpose of the work. For this, we choose PyCharm, which is one of the best known IDEs and one of the best dedicated with full functions for Python.

Figure 1: *PyCharm's IDE*

At the time of writing the code in Python, we may need to use some libraries that are not available in the standard version of the language. We will make use of two of them mainly: Tkinter and Scipy, the latter contains packages known as Pandas, Stats, Matplotlib or Numpy. We will also use the Statsmodels and scikit-learn libraries for some statistical tests.

5. T-Qube

IN this chapter, we will present the features and results we have obtained when compiling the code generated by our tool: **T-Qube**. We will focus on explaining its general operation and will be presented as a user guide.

Figure 2: *Logo of the tool designed: T-Qube.*

In order to achieve the main objective, the tool will follow the following workflow:

- We will obtain the data from a file *.csv*.
- We will take the columns that the user wants to analyze and we will pass the columns of the table to *arrays*.
- We will perform the statistical study that the user wants.
- From the obtained results, we can extract graphs and the tables belonging to the statistical tests carried out.
- Finally, we will allow the user to save the results of the study.

In the following image we can see an example of the operation of the tool.

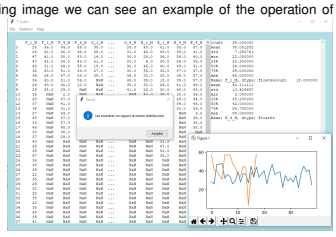


Figure 3: *The main window of the application and some results that we can obtain.*

6. Conclusions, futures works and personal assessment

TO complete this work, in this chapter we will deal with the achievements of the work, the difficulties we have encountered in the development of it, in the same way that we will mention future lines of work and a brief personal assessment.

References

[1] MOKHTAR, E. (2018) *Ejemplos De La GUI De Python (Tutorial De Tkinter)*. Consulted in https://likegeeks.com/es/ejemplos-de-la-gui-de-python/

[2] (2019) *SciPy.org* Consulted in https://www.scipy.org/

[3] (2015) *Quantitative Data Analysis: Choosing a statistical test*. Consulted in https://www.researchgate.net/profile/Mohamed[...]download/Quantitative+Data+Analysis-+Choosing+a+statistical+test.pdf

[4] MCKINNEY, W. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. In *Estados Unidos* at 30/12/2011

[5] LUTZ, M. *Learning Python 5TH Edition*. In *Estados Unidos* at 07/06/2013.