

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

Desarrollo de un paquete de enseñanza estadística
básica en R para su posterior aceptación en
CRAN

Autor: Dennis Monheimius Lombardía

Tutor/es: Juan José Cuadrado Gallego

TRIBUNAL:

Presidente: Luis Usero

Vocal 1º: Antonio Navidad

Vocal 2º: Juan José Cuadrado

FECHA: September 21, 2018

Contents

1 RESUMEN

Castellano: En el presente documento se muestra el proceso llevado a cabo para planificar y desarrollar un paquete de enseñanza estadística básica y poder subirlo al repositorio oficial CRAN. Se explican desde los conocimientos teóricos necesarios sobre el contenido de cada archivo de la estructura de un paquete hasta el desarrollo de código en lenguaje R y C para poder aplicar el modelo planteado.

English: In this document, the process carried out to help you developing a bank of basic statistics and upload it to the official repository of CRAN is shown. They are explained from the theoretical knowledge required on the content of each file of the structure of a package to the development of code in R and C language to be able to apply the proposed model.

1.1 Palabras clave:

Paquete R, paquete de funciones estadísticas en R, desarrollo R, desarrollo paquete R, CRAN, estadística básica.

2 INTRODUCCIÓN

Este informe recoge el desarrollo de un paquete de enseñanza estadística orientada a alumnos de enseñanza secundaria obligatoria y bachillerato. La idea de este proyecto surge a raíz de intentar explotar las herramientas y funcionalidades que ofrece el lenguaje R, el cual suele estar orientado al trabajo profesional de estudios estadísticos y matemáticos, para orientarlo a la enseñanza de estadística básica. El uso del paquete tendría como objetivo ser utilizado en las aulas como apoyo a la enseñanza tradicional. De esta manera los alumnos dispondrían de una herramienta profesional muy potente pero ajustada a sus necesidades y nivel.

Se ha elegido el desarrollo del paquete sobre el lenguaje R debido a que, como hemos comentado anteriormente, es una herramienta muy potente que se utiliza ampliamente en el campo profesional de la estadística y el análisis de datos. Debido al auge actual de la DataScience hemos creído conveniente utilizar este lenguaje para acostumbrar a los alumnos a utilizar herramientas profesionales pero que se adapte a sus necesidades.

A continuación se introduce al lector sobre que es el lenguaje R:

Los lenguajes de tratamiento estadístico de datos surgieron en torno a 1976, debido a la necesidad de trabajar con grandes cantidades de datos en diferentes áreas profesionales (medicina, biología, etc.) para transformarlos en información de la cual pueda inferirse cierto conocimiento relacionado. En este año, el tratamiento de los datos se realizaba por medio de subrutinas en Fortran (muy tedioso), llamado “S”. Fue a partir de la década de los noventa cuando aparecieron nuevas versiones de S que se asemejan más a lo que actualmente conocemos dentro del entorno de R.

R es una agrupación de instalaciones software para la manipulación, el cálculo y la representación gráfica de datos que forma parte de GNU y se distribuye bajo licencia GNU GLP (causa más probable de que tenga tanta repercusión, y se pueda encontrar fácilmente información sobre este lenguaje). Destaca por poseer una colección grande, coherente e integrada de herramientas intermedias para el análisis de datos; así como para la representación gráfica y visual de los mismos, usando un lenguaje de programación correctamente desarrollado, simple y efectivo, como es S. Por lo tanto, R puede considerarse como una implementación del lenguaje S.

El objetivo es conseguir desarrollar un paquete que sea aceptado por CRAN e incluido en su repositorio para que sea accesible para todo el mundo.

En los siguientes apartados se explica el proceso llevado a cabo y necesario para el correcto desarrollo del paquete.

3 BASE TEÓRICA

3.1 Primeros pasos y Software necesario para Windows

El principal objetivo de este trabajo es documentar el proceso de creación o construcción de un paquete de R, así como el proceso de subida de dicho paquete al repositorio de CRAN para su utilización como software de libre uso. Las principales razones para decidimos a crear un paquete de R son las siguientes:

En primer lugar, aunque dispongamos de paquetes en el repositorio que pueden contener parte de las funcionalidades que se van a incluir en nuestro paquete, crear un paquete nos obligara a pulir nuestras funciones, datos y código. Además, se deberá documentar todo el trabajo y dar ejemplos claros para que nuestras funciones sean más fáciles de usar y estén incluidos en los comandos de R que ofrecen ayuda con respecto al uso de las funciones.

Nuestro trabajo se verá plasmado en el repositorio de CRAN, donde nuestro paquete estará compartido de manera elegante tanto para su utilización por

parte de los usuarios del mismo como para poder mejorarlo añadiendo más documentación, contribuyendo al crecimiento de R. Concretamente, CRAN es una red del sitio WWW que contiene las distribuciones R y el código contribuido, especialmente paquetes R. Además, se alienta a los usuarios de R a unirse al proyecto de colaboración y enviar sus propios paquetes a CRAN (como pretendemos hacer nosotros).

Para la creación de un paquete de Windows, necesitamos instalar una serie de programas que le faltan. Lo primero a instalar es un conjunto mínimo de utilidades tipo Unix (las llamadas Rtools) de Brian Ripley y que ahora mantiene Duncan Murdoch. A continuación, deberemos instalar algunos compiladores GNU que se encuentran reunidos en MinGW-w64 (necesarios para paquetes que contienen código C, Fortran o C++). En nuestro caso de momento no es necesario. Estos dos primeros componentes se obtienen directamente de la página de CRAN. Entramos en Download R for Windows y Rtools, para seleccionar el ejecutable RtoolsXX.exe más actual. Al seleccionar el instalador y ejecutarlo, se debe mantener los nombres de las carpetas de instalación por si debemos introducirlas en la variable PATH. El programa de instalación nos brinda la posibilidad de modificar automáticamente la variable PATH, lo que se recomienda aceptar. Por último, nos faltará el compilador de la ayuda HTML de Microsoft y una versión del procesador de textos TeX (como MiKTeX para Windows). El compilador HTML de Windows se puede descargar del Download Center, tratándose del “Microsoft HTML Help Workshop” y el archivo se llama htmlhelp.exe. Para la versión del procesador de textos TeX hemos elegido la distribución MikTeX en su versión básica (<http://www.miktex.org/>).

Una vez hemos instalados los anteriores softwares, solo necesitaremos disponer de un editor ASCII como es el propio RStudio. Cuando instalamos las Rtools, el programa de instalación nos ofrece editar la variable PATH de la siguiente manera:

```
PATH=c:/Rtools/bin;c:/Rtools/gcc-4.6.3/bin;...
```

Se debe comprobar que las carpetas son correctas, ya que las rutas a los programas a ejecutar que acabamos de añadir deben ser válidas para la localización de los ejecutables. También es importante que dejemos el PATH en el mismo orden.

A continuación, procedemos a probarlos. Para ello abrimos la línea de comandos de Windows o cmd e introducimos el siguiente comando: Path. Este nos devolverá la lista de carpetas de la variable de entorno PATH, donde podemos observar si se encuentran añadidas las rutas de los nuevos programas de manera correcta o si hay algún tipo de error.

Ahora procedemos a comprobar si la ejecución de estos programas es correcta desde la propia línea de comandos.

Lo primero que probamos es a escribir R y ejecutamos. Con ello abrimos una

```
C:\Users\edube>path
PATH=C:\Rtools\bin;C:\Rtools\gcc-4.6.3\bin;C:\Rtools\mingw_32\bin;C:\Program Files (x86)\Intel\iCLS Client\bin;C:\Program Files\Intel\iCLS Client\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Intel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\;C:\Program Files (x86)\VIDIA Corporation\PhysX\Common;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH\;C:\Rtools\gcc-4.6.3\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\WiXTeX\2.9\miktex\bin\x64\;C:\Program Files\R\R-3.5.1\bin\x64\;C:\Users\edube\AppData\Local\Microsoft\WindowsApps;C:\Users\edube\AppData\Roaming\Dashlane\6.0.2.11187\bin\Firefox\Extensions\{442718d9-475e-452a-b3e1-fb1ee168e9f3}\components;C:\Users\edube\AppData\Roaming\Dashlane\6.0.2.11187\ucrt;C:\Users\edube\AppData\Roaming\Dashlane\6.0.2.11187\bin\Qt;C:\Users\edube\AppData\Roaming\Dashlane\6.0.2.11187\ucrt;C:\Users\edube\AppData\Roaming\Dashlane\6.0.2.11187\bin\Sql
```

Figure 1: Contenido de la variable de entorno PATH

sesión de “Rterm” que es la versión de línea de comandos de R. Es imprescindible si no vamos a trabajar desde RStudio. Al ir todo bien, deberíamos de ver el mensaje de bienvenida de R, por lo que salimos directamente con el comando `q()`:

```
C:\Users\edube>R

R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

> q()
Guardar imagen de área de trabajo? [y/n/c]: n
```

Figure 2: Ejecución del comando R en CMD

Para comprobar si los demás programas se encuentran bien instalados, ejecutamos las siguientes instrucciones:

- Para el compilador GNU de C: gcc -help

```

C:\Users\edube>gcc --help
Usage: gcc [options] file...
Options:
  -pass-exit-codes      Exit with highest error code from a phase
  --help                Display this information
  --target-help         Display target specific command line options
  --help={target|optimizers|warnings|params|[^]{joined|separate|undocumented}}[,...]
                        Display specific types of command line options
  (Use '-v --help' to display command line options of sub-processes)
  --version             Display compiler version information
  -dumpspecs            Display all of the built in spec strings
  -dumpversion          Display the version of the compiler
  -dumpmachine          Display the compiler's target processor
  -print-search-dirs    Display the directories in the compiler's search path
  -print-libgcc-file-name
                        Display the name of the compiler's companion library
  -print-file-name=<lib>
                        Display the full path to library <lib>
  -print-prog-name=<prog>
                        Display the full path to compiler component <prog>
  -print-multi-directory
                        Display the root directory for versions of libgcc
  -print-multi-lib      Display the mapping between command line options and
                        multiple library search directories
  -print-multi-os-directory
                        Display the relative path to OS libraries

```

Figure 3: Ejecución del compilador desde CMD

- Para el programa TeX: tex --help

```

C:\Users\edube>tex --help
Usage: tex [OPTION...] [COMMAND...]
  -alias=APP            Pretend to be APP. This affects both
                        the format used and the search path.
  -aux-directory=DIR    Use DIR as the directory to write
                        auxiliary files to.
  -buf-size=N           Set buf size to N.
  -c-style-errors       Enable file:line:error style messages.
  -disable-8bit-chars   Make only 7-bit characters printable.
  -disable-installer    Disable the package installer. Missing
                        files will not be installed.
  -disable-pipes        Disable input (output) from (to)
                        processes.
  -disable-write18      Disable the \write18{COMMAND} construct.
  -dont-parse-first-line
                        Do not parse the first line of the input
                        line to look for a dump name and/or
                        extra command-line options.
  -enable-8bit-chars    Make all characters printable by default.
  -enable-encTeX        Enable EncTeX extensions such as \mubyte.
  -enable-installer     Enable the package installer. Missing
                        files will be installed.
  -enable-mltex         Enable MLTeX extensions such as
                        \charsubdef.

```

Figure 4: Ejecución del editor de texto desde CMD

- Una de las Rtools: tar --help

```

C:\Users\edube>tar --help
Usage: tar [OPTION...] [FILE]...
GNU 'tar' saves many files together into a single tape or disk archive, and can
restore individual files from the archive.

Examples:
  tar -cf archive.tar foo bar  # Create archive.tar from files foo and bar.
  tar -tvf archive.tar         # List all files in archive.tar verbosely.
  tar -xvf archive.tar         # Extract all files from archive.tar.

Main operation mode:

-A, --catenate, --concatenate  append tar files to an archive
-c, --create                   create a new archive
-d, --diff, --compare          find differences between archive and file system
--delete                       delete from the archive (not on mag tapes!)
-r, --append                   append files to the end of an archive
-t, --list                     list the contents of an archive
--test-label                   test the archive volume label and exit
-u, --update                   only append files newer than copy in archive
-x, --extract, --get           extract files from an archive

Operation modifiers:

--check-device                 check device numbers when creating incremental
                               archives (default)

```

Figure 5: Ejecución de las herramientas de R desde CMD

Al aparecer una lista de opciones, nos indica que los programas están listos para ser utilizados. En nuestro caso, el compilador GNU de C y las Rtools funcionaban correctamente, pero el programa TeX daba fallo. Para corregir el error, habrá que localizar la carpeta que contiene el ejecutable de dicho programa “C:/Program Files/MiKTeX 2.9/miktex/bin/x64” y corregir adecuadamente la variable de entorno PATH. Para ello, podemos ir:

“Panel de control - Sistema y Seguridad - Sistema - Configuración avanzada del sistema”, y seleccionamos el botón “Variables de entorno” y en el menú resultante buscamos la variable de entorno PATH del sistema (no del usuario, para que pueda ser accedido desde todo el sistema y cualquier usuario) y seleccionamos el botón “Editar”. Añadimos la correcta ruta del archivo ejecutable para MikTeX y volvemos a probar, dando como resultado la imagen mostrada anteriormente con la respuesta que indica que su instalación ha sido correcta.

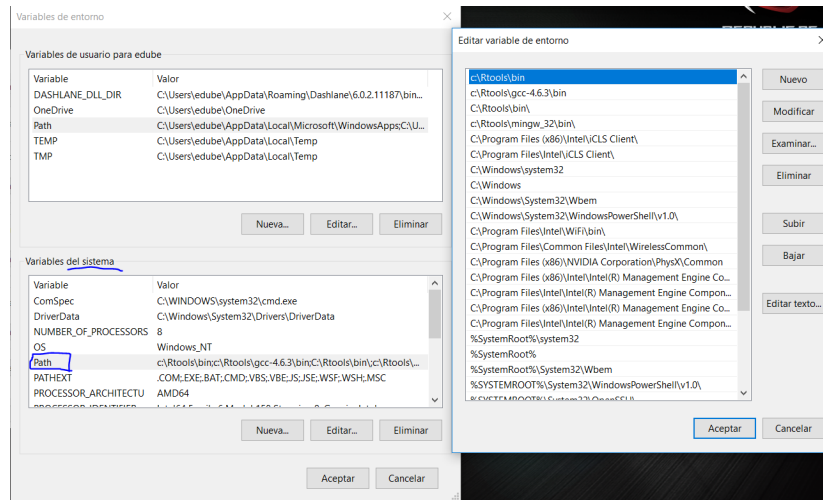


Figure 6: ..

También se pueden añadir nuevas rutas a través de la línea de comandos cmd introduciendo la siguiente instrucción por cada nueva ruta que se quiera añadir al archivo path:

Set path = %path%; C: /MiNuevaRuta

Con este comando se consigue añadir una nueva ruta a la variable de entorno.

3.2 ¿Qué es un paquete de R?

Un paquete es un directorio de archivos que extienden de R; es decir, un paquete fuente con los archivos maestros para la creación de un paquete, o un paquete ya instalado como resultado de ejecutar la instalación para CMD de R sobre un paquete fuente. En ciertas plataformas, como Windows o macOS, también pueden encontrarse paquetes binarios. Estos consisten en un archivo zip o tarball que contiene los archivos de un paquete que puede descomprimirse en cualquier lugar en vez de tener que ser instalado desde las fuentes.

Una vez se ha definido lo que es un paquete, podemos explicar para lo que se utilizan. Se ha mencionado que un paquete es una extensión de R, donde se encontraran los archivos necesarios para aplicar una serie de funciones que podrán ser utilizadas desde cualquier entorno de R. Entonces, los paquetes proporcionan un mecanismo para cargar código, para cargar datos y añadir documentación opcional, según sean las necesidades. Para aquellos paquetes que contienen código en algunos de los lenguajes disponibles y tiene que ser compilado, se deberá de mirar en el manual de Instalación y Administración de R para informarse de las herramientas necesarias para cada sistema operativo. La

propia distribución de R incluye alrededor de 30 paquetes.

También es importante diferenciar los términos de paquete y biblioteca, los cuales a veces tienden a confundirse. El término de biblioteca se utiliza en dos sentidos en la documentación de R.

El primero se refiere al directorio en el cuál los paquetes están instalados; es decir, la ruta donde son instalados los paquetes a veces es llamado directorio de la biblioteca, ya que la biblioteca es un directorio que contiene paquetes como directorios, que a su vez contienen más directorios.

El otro sentido de la palabra biblioteca es el utilizado por el sistema operativo, como una biblioteca compartida, dinámica o estática o una DLL (bibliotecas de enlace dinámico). El concepto de biblioteca compartida (biblioteca dinámica en macOS) se refiere a una colección de código compilado a lo que se puede vincular un paquete, siendo usado para R en algunas plataformas. En la mayoría de las plataformas, los objetos compartidos y las DLL pueden ser ambos cargados en el proceso R y estar vinculados, pero macOS distingue entre objetos compartidos (`extension.so`) y bibliotecas dinámicas (`extension.dylib`).

Lo más normal es que la instalación de un paquete tome los archivos fuentes necesarios para componer dicho paquete y lo incluya en una biblioteca usando la instalación de R para CMD, o utilizando el comando de R `"install.packages"`. Además de instalar un paquete ya creado en una biblioteca, se puede construir paquetes fuente. Estos paquetes pueden verificarse cuando se realiza y ejecuta una instalación de prueba (incluso ejecutar sus ejemplos) para probar el contenido del paquete logrando consistencia y portabilidad. Construir paquetes fuente o de origen implica tomar un directorio de origen y crear un tarball listo para la distribución, incluida la limpieza y creación de la documentación PDF correspondiente a partir de cualquier viñeta que pueda contener.

Por otro lado, cabe destacar que hablar de la compilación de un paquete fuente o de origen no es del todo correcto cuando nos estamos refiriendo a un paquete. Instalar un paquete fuente que contiene código en diversos lenguajes (como C, C++ o Fortran) implicará la compilación en ese paquete de dicho código. También existe la posibilidad de la compilación de código R a nivel de 'byte' en un paquete, usando las facilidades suministradas por el compilador de paquetes, que desde la versión 3.5.0 de R están habilitadas por defecto para todos los paquetes. Anteriormente, los paquetes base y recomendados eran normalmente compilados en bytes, pero esto debía especificarse explícitamente si se trataba de otros paquetes. Por lo tanto, compilar un paquete puede significar compilar en byte su código R.

También solía ser inequívoco hablar sobre cargar un paquete instalado usando el comando `library()`, pero desde la llegada de los espacios de nombres de paquetes, esto ha sido menos claro debido a que ahora se habla acerca de cargar

el espacio de nombres del paquete y a continuación adjuntar el paquete para que se vuelva visible en la ruta de búsqueda. La biblioteca de funciones realiza ambos pasos, pero el espacio de nombres de un paquete puede ser cargado sin que el paquete esté conectado.

El concepto de carga diferida de código o datos es parte de la instalación, siempre seleccionada para código R pero opcional para los datos (seleccionada por el mantenedor del paquete). Cuando se carga un paquete/espacio de nombres que lo usa, el entorno de este paquete o espacio de nombres se llena con promesas para todos los objetos que han sido nombrados: cuando se evalúan estas promesas, cargan el código real de una base de datos.

Existen bases de datos separadas para el código y los datos, almacenados en los subdirectorios de R y de datos. Cada base de datos consta de dos archivos, `nombre.rdb` y `nombre.rdx`. El archivo `.rdb` es una concatenación de objetos serializados, mientras que el archivo `.rdx` contiene un índice. El nombre del índice, o ‘mapa’ `nombre.rdx`, es un objeto R serializado comprimido para ser leído por `readRDS`, tratándose de una lista con tres elementos variables, referencias y comprimidos. Los dos primeros son listas de vectores enteros que proporcionan el desplazamiento y la longitud del objeto serializado en el archivo `nombre.rdb`. Las variables de elemento tienen una entrada para cada objeto nombrado, las referencias serializan un entorno temporal utilizado cuando los entornos con nombre se agregan a la base de datos, y comprimido es un indicador lógico que indica si los objetos serializados fueron comprimidos (actualmente la compresión siempre se usa). Posteriormente se tienen que agregar los valores comprimidos igual a 2 y 3 para compresión `bzip2` y `xz`.

Los objetos normalmente se almacenan en un formato comprimido en `gzip` con un encabezado de 4 bytes que indica la longitud serializada sin comprimir (en XDR, orden de byte de gran tamaño) y se lee mediante una llamada al primitivo `lazyLoadDBfetch`. Esto hace que la carga diferida no sea adecuada para objetos realmente grandes (la longitud no serializada de un objeto R puede exceder los 4GB). En la compresión `bzip2` y `xz` se agrega un quinto byte al encabezado para el tipo de compresión y almacenan objetos serializados sin comprimir si la compresión se expande de ellos.

El cargador de una base de datos de código o datos lazy-load es la función `lazyLoad` (de carga lenta) en el paquete `base`, pero se debe tener en cuenta que hay una copia separada para cargar la base en el archivo `R_HOME/base/R/base`. Las bases de datos lazy-load se crean mediante el código en `src/library/tools/R/makeLazyLoad.R`.

La herramienta principal es la función no exportada `makeLazyLoadDB` y la inserción de las entradas de la base de datos se realiza mediante llamadas a `.Call` (“`R.lazyLoadDBinsertValue`”,...). Estas bases de datos lazy-load si ocupan menos de 10 MB se guardan en la memoria durante el primer uso, para

evitar las altas latencias en sistemas de archivos propensos (dispositivos extraíbles y sistemas de archivos montados en red en Windows).

Entonces, las bases de datos lazy-load se cargan en las exportaciones de un paquete, pero no en el propio entorno de espacio de nombres, por lo que son visibles cuando el paquete está conectado y también a través del operador ‘::’. El mismo mecanismo de base de datos se utiliza para almacenar los archivos Rd analizados. Uno o todos los objetos analizados se recuperan mediante una llamada a `tools::fetchRdDB`.

3.2.1 ¿Por qué desarrollar paquetes R propios?

Hasta este momento hemos presentado que es R y que son los paquetes de R, pero si disponemos de un repositorio donde podemos encontrar infinidad de paquetes para diversos usos, ¿para qué crear nuevos paquetes, nuestros paquetes? Se puede responder a esta pregunta fácilmente, argumentando que se deberán de crear paquetes para aquellas funciones que no están recogidas en otros paquetes o que presenten una nueva forma de trabajar con los datos para cualquier proceso. El caso es que no solo se trata de eso, sino que también pueden crearse paquetes propios cuyas funciones ya estén implementadas en otros paquetes disponibles en el repositorio.

Estas son las principales razones por las que se deben crear y publicar nuevos paquetes R:

Se trata de una herramienta cómoda para mantener colecciones coherentes de funciones y datos, permitiendo publicar código de forma que pueda ser empleado por otros siguiendo unas estructuras comunes. Entonces, es un modo elegante de compartir nuestro trabajo y los posibles usuarios del paquete agradecerán el esfuerzo de mejora y documentación que requiere. Además, al distribuirlo para que otros puedan usarlo, se obtiene realimentación sobre lo que ha sido publicado, aumentado su robustez y ampliando sus funcionalidades, pudiendo conectarse con otras herramientas y proyectos.

Respecto a la estructura del paquete creado, llevarlo a cabo implica organizar, limpiar y documentar el código, por lo que nos obligará a pulir nuestras propias funciones, datos y código. Además, también deberemos de documentar todo el trabajo y dar ejemplos claros. Las funciones serán más fáciles de usar y se podrá utilizar el comando ‘?’ para ver detalles de los parámetros, los resultados y ejemplos.

Por último y más importante, es la mejor manera de contribuir al crecimiento de R. (Ampliar con mas documentación de por que es importante contribuir a R y CRAN).

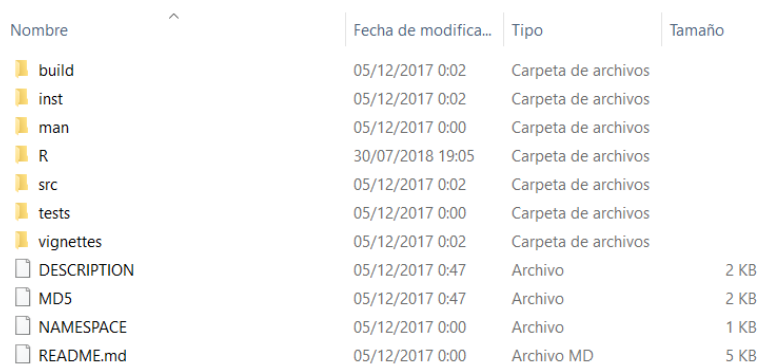
3.3 Estructura de un paquete R

3.3.1 Estructura general de un paquete

Una vez hemos explicado lo que es un paquete R, algunos conceptos relacionados y por qué es muy conveniente desarrollar paquetes propios, vamos a explicar cómo es la estructura de dichos paquetes.

Un paquete R está compuesto por un subdirectorio que contienen dos archivos y una serie de subdirectorios. Los dos archivos son los llamados DESCRIPTION y NAMESPACE, y los subdirectorios son R, data, demo, exec, inst, man, po, src, test, tools y vignettes (pueden faltar algunos de estos subdirectorios, pero nunca deben aparecer vacíos). Además de esto, el subdirectorio del paquete puede contener los archivos INDEX, configure, cleanup, LICENSE, LICENCE y NEWS. También puede haber archivos como INSTALL, para contener las instrucciones de instalación no estándar si se diera el caso, o README/README.md2 o ChangeLog, que es muy útil para los usuarios finales, aunque sea ignorado por R.

Se incluye como ejemplo la imagen del subdirectorio del paquete llamado parallelDist, utilizada para calcular matrices de distancia para matrices de entrada continuas, binarias y multidimensionales, con una amplia variedad de funciones de distancia predefinidas de los paquetes R llamados ‘stats’, ‘proxy’ y ‘dtw’, así como funciones definidas por el usuario escritas en C++:



Nombre	Fecha de modifica...	Tipo	Tamaño
build	05/12/2017 0:02	Carpeta de archivos	
inst	05/12/2017 0:02	Carpeta de archivos	
man	05/12/2017 0:00	Carpeta de archivos	
R	30/07/2018 19:05	Carpeta de archivos	
src	05/12/2017 0:02	Carpeta de archivos	
tests	05/12/2017 0:00	Carpeta de archivos	
vignettes	05/12/2017 0:02	Carpeta de archivos	
DESCRIPTION	05/12/2017 0:47	Archivo	2 KB
MD5	05/12/2017 0:47	Archivo	2 KB
NAMESPACE	05/12/2017 0:00	Archivo	1 KB
README.md	05/12/2017 0:00	Archivo MD	5 KB

Figure 7: ..

En ella podemos ver que se encuentran gran parte de los subdirectorios y archivos mencionados anteriormente.

La utilidad de R para CMD build permite ir incorporando archivos en un directorio ya construido, por lo que es bastante interesante ir incorporando archivos y viendo el resultado para asegurarse de una correcta construcción del paquete.

Se debe considerar que, excepto donde se mencione específicamente, los paquetes no deben contener ningún tipo de archivos/directorios Unix ‘ocultos’; es decir, aquellos cuyo nombre comienza en un punto.

Además, el subdirectorio del paquete debe poseer el mismo nombre que el propio paquete. Se debe tener especial consideración de que algunos archivos del sistema no distinguen entre las letras mayúsculas y las minúsculas, por lo que para mantener la portabilidad se recomienda que las distinciones entre mayúsculas y minúsculas no sirvan para diferenciar dos paquetes distintos, ya que podría dar lugar a errores. Esto se refiere a que si ya existe un paquete llamado `Lstatics`, no se debe crear un paquete que se llame `lStatics`, `lStatics`, etc. Para garantizar que los nombres de los archivos sean válidos en todos los sistemas de archivos y compatibles para todos los sistemas operativos, tanto los caracteres de control ASCII como los caracteres especiales: (`""`, `""`, `:"`, `/"`, `l`, `;`, `?`, `"`) no están permitidos en los nombres de archivo. Por último, tampoco están permitidos los archivos con los nombres `con`, `prn`, `aux`, `clock$`, `null`, `com1` hasta `com9`, y de `lpt1` a `lpt9` tras la conversión a minúsculas y el desmantelamiento de las diferentes extensiones que pudiera tener.

Por otro lado, los nombres de archivo en el mismo directorio no deben diferir solo por caso. También, los nombres base de los archivos `.Rd` pueden ser usados en las URL, por lo que deben de estar nombrados con caracteres ASCII y no contener el carácter del porcentaje. La portabilidad máxima se conseguirá nombrando archivos que solo contengan aquellos caracteres que no han sido excluidos ya. Para aquellos caracteres que no sean del alfabeto inglés no puede garantizarse que sean compatibles en todas las configuraciones regionales. Cabe añadir que sería una buena práctica evitar los meta-caracteres propios de la Shell o consola (paréntesis, corchetes, símbolo del dólar, comilla simple, ...).

Como hemos mencionado anteriormente, los paquetes se pueden distribuir también como tarballs, los cuales tienen un límite en la duración de la ruta: para una portabilidad máxima de 100 bytes. Debido a que no es aconsejable que un paquete fuente contenga archivos ejecutables binarios, debido a que no son portables, la comprobación R de CMD (check) advertirá sobre ellos a menos que estén listados en un archivo llamado `BinaryFiles` en el nivel superior del paquete. Hay que tener en cuenta que CRAN no aceptará envíos que contengan archivos binarios, incluso si están listados en dicho archivo, por lo que en nuestra práctica no incluiremos ningún archivo de tales características.

3.3.2 Construcción de la estructura del paquete con RStudio

Primero, puesto que es la primera vez que hablamos de RStudio, vamos a explicar brevemente lo que es. RStudio es un entorno de desarrollo integrado (IDE) para el lenguaje de programación R, y por lo tanto dedicado a la computación estadística y la representación de datos mediante diferentes tipos de

gráficos. Tiene como misión proporcionar el entorno informático estadístico R para trabajar con datos.

Se compone de un editor de sintaxis que apoya la ejecución del código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo, además de la propia consola de trabajo. Esta disponible para Windows, macOS y Linux, o para navegadores conectados a RStudio Server o RStudio Server Pro.

Una vez hemos explicado que es RStudio, vamos a proceder a mostrar los diferentes pasos necesarios para crear un nuevo paquete a través de este entorno. Para crear un nuevo paquete con RStudio, usaremos la instrucción ‘Create Project’ desde el menú Project o desde la barra de herramientas. Se recomienda iniciar una sesión en el entorno y crear el proyecto desde cero, ya que al introducir el comando anterior mientras estamos trabajando, eso cierra la sesión y abre una nueva.

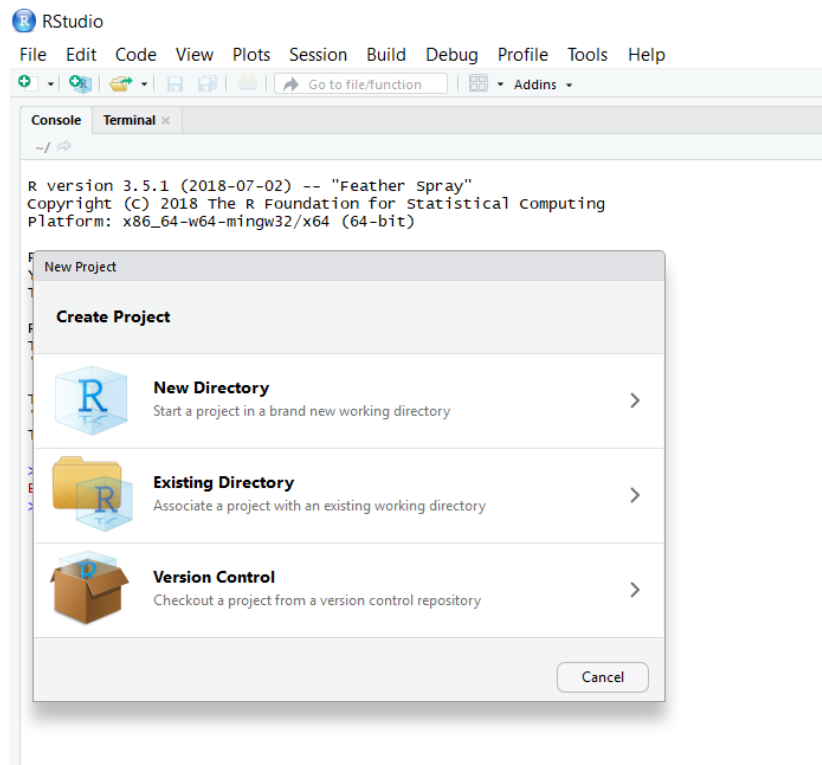


Figure 8: ..

A continuación, debemos seleccionar que queremos crear un nuevo proyecto en un nuevo directorio de trabajo. También debemos seleccionar que nuestro proyecto que vamos a crear sea del tipo paquete de R.

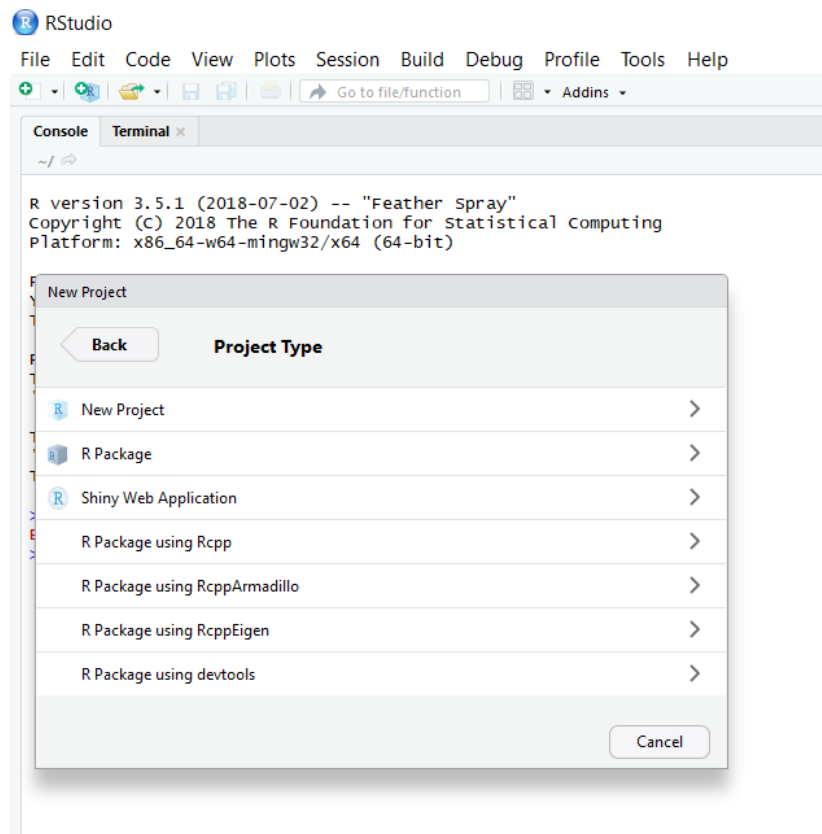


Figure 9: ..

Para concluir, nos aparecerá una última pantalla del menú de creación de un nuevo proyecto donde se debe especificar el tipo (que ya lo habíamos elegido pero aparece por mera información), el nombre del paquete (donde debemos tener en cuenta las consideraciones anteriores sobre los caracteres válidos para dicho nombre, así como los nombres que están reservados y tampoco son válidos), la ruta del archivo o los archivos de origen/fuente a partir de los cuales se va a crear el nuevo paquete (contendrá las funciones) y finalmente indicar el subdirectorio donde queremos crear el nuevo paquete.

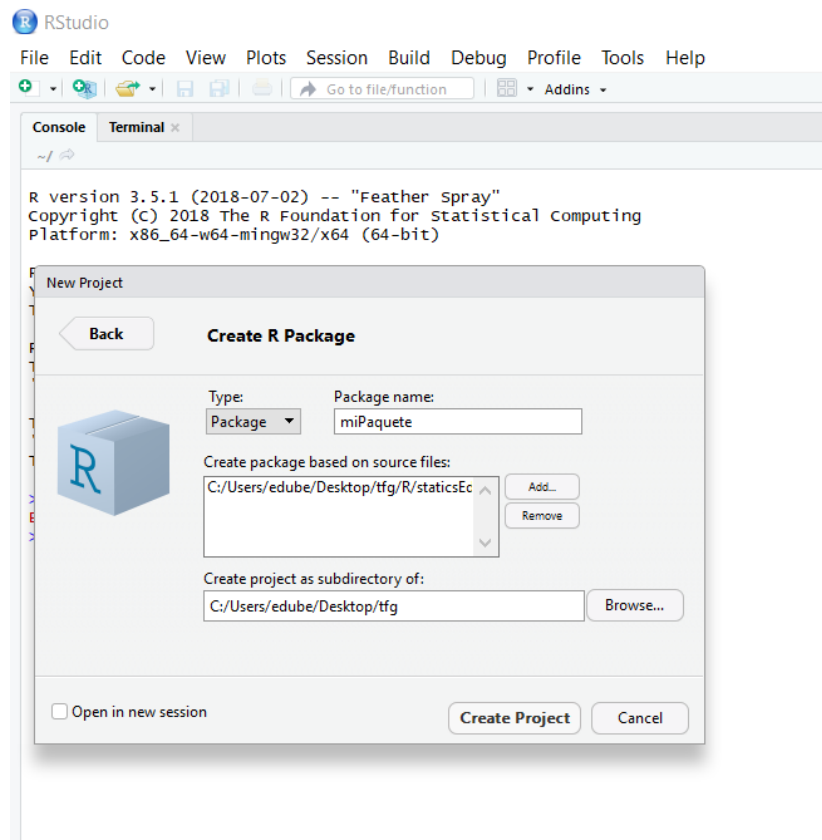


Figure 10: ..

Como vemos en la imagen, hemos indicado el nombre del paquete “miPaquete” y que se cree en el subdirectorio “C:/Users/edube/Desktop/tfg”, por lo que vamos a ver en ese subdirectorio si nos ha creado correctamente el nuevo paquete:

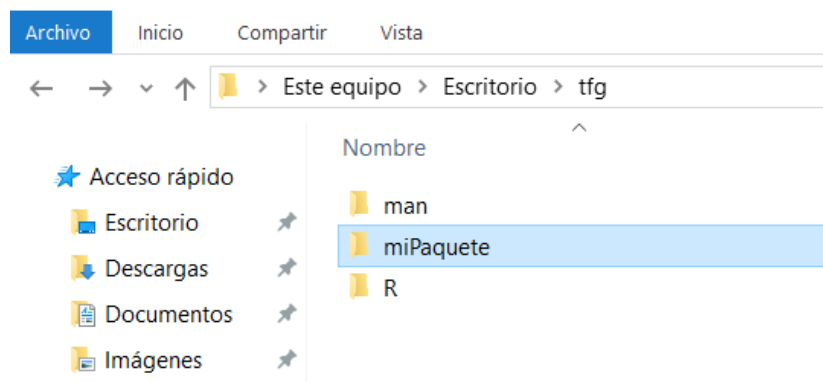


Figure 11: ..

Ya hemos visto que esta creada la carpeta del paquete en el lugar correspondiente, ahora vamos a mostrar el interior de dicha carpeta (ver las carpetas y archivos que crea por defecto):

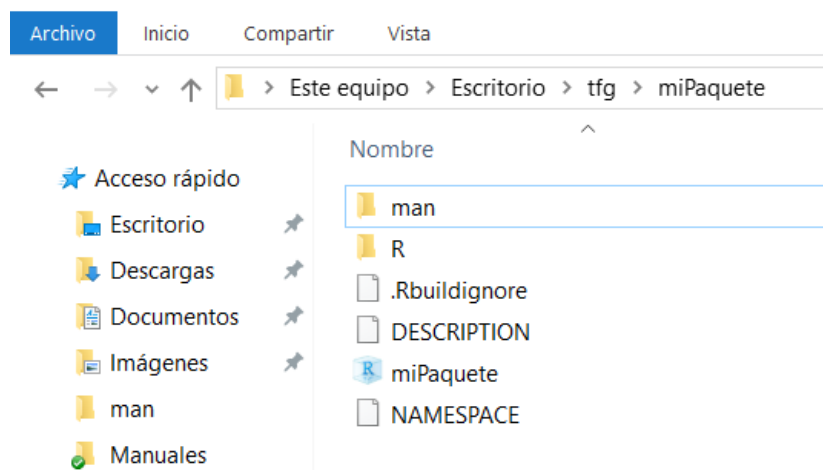


Figure 12: ..

3.3.3 Construcción de la estructura del paquete de manera manual

Como ya hemos explicado anteriormente, este paso se puede realizar tanto a través de RStudio y la opción de creación de un nuevo proyecto, como de manera manual mediante la función `package.skeleton()`. Lo más sencillo es utilizar la función con sus mínimos parámetros:

```

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> rm(list = ls())
> package.skeleton("miPaquete", path="C:/Users/edube/Desktop/tfg")
Creating directories ...
Error in package.skeleton("miPaquete", path = "C:/Users/edube/Desktop/tfg") :
  directory 'C:/Users/edube/Desktop/tfg/miPaquete' already exists
> rm(list = ls())
> package.skeleton("miPaqueteManual", path="C:/Users/edube/Desktop/tfg")
Creating directories ...
Creating DESCRIPTION ...
Creating NAMESPACE ...
Creating Read-and-delete-me ...
Saving functions and data ...
Making help files ...
Done.
Further steps are described in 'C:/Users/edube/Desktop/tfg/miPaqueteManual/Read-and-delete-me'.
> |

```

Figure 13: ..

Lo primero que realizamos es limpiar el espacio de trabajo (aunque en nuestro caso no era necesario). Para crear la estructura en la carpeta con nombre “miPaqueteManual” y en la ruta seleccionada se debe introducir el comando o instrucción `package.skeleton()` de la manera que se muestra en la imagen. Como nos hemos creado previamente el paquete “miPaquete” con RStudio en el mismo subdirectorio, podemos observar como no nos deja crearnos otro paquete con el mismo nombre (primera ejecución de la instrucción `package.skeleton` que da error). Cabe mencionar que se deben de cargar en R los objetos (funciones y datos) del futuro paquete antes de llevar a cabo la ejecución de la instrucción.

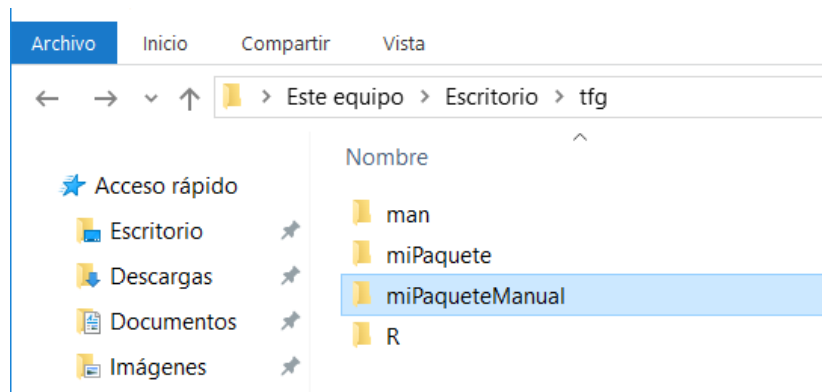


Figure 14: ..

Podemos ver que se ha creado la estructura del paquete correspondiente en el subdirectorio indicado, ahora vamos a observar el contenido de dicho paquete:

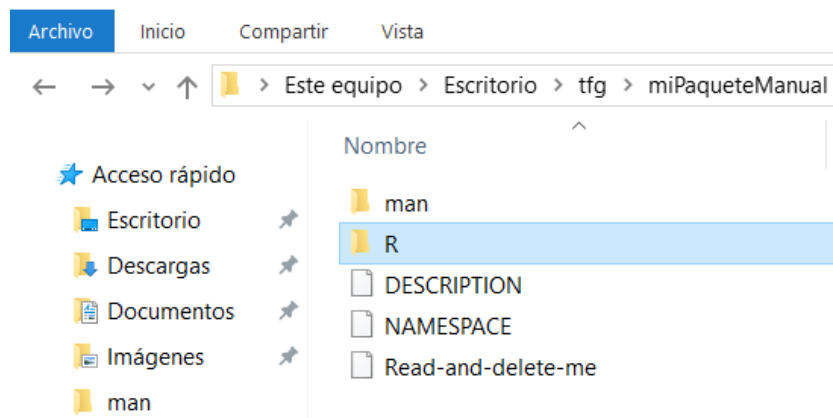


Figure 15: ..

Podemos observar que se ha generado una estructura del paquete muy similar o prácticamente igual a la generada por RStudio, donde solamente se encuentra el archivo fuente .R en el subdirectorío R, y a partir de la cual se irán añadiendo el resto de archivos y subdirectoríos según veamos necesario.

3.3.4 El archivo DESCRIPTION

El archivo DESCRIPTION contiene información básica sobre el paquete. Esta información se encuentra dispuesta con un formato concreto, que es el de una versión de un “Archivo de Control Debian”, donde se estructura en distintos campos. Cada uno de estos campos comienzan con un nombre ASCII seguido de dos puntos y un espacio, y el valor del propio campo.

```
Package: pkgname
Version: 0.5-1
Date: 2015-01-01
Title: My First Collection of Functions
Authors@R: c(person("Joe", "Developer", role = c("aut", "cre"),
  email = "Joe.Developer@some.domain.net"),
  person("Pat", "Developer", role = "aut"),
  person("A.", "User", role = "ctb",
    email = "A.User@whereever.net"))
Author: Joe Developer [aut, cre],
  Pat Developer [aut],
  A. User [ctb]
Maintainer: Joe Developer <Joe.Developer@some.domain.net>
Depends: R (>= 3.1.0), nlme
Suggests: MASS
Description: A (one paragraph) description of what
  the package does and why it may be useful.
License: GPL (>= 2)
URL: https://www.r-project.org, http://www.another.url
BugReports: https://pkgname.bugtracker.url
```

Figure 16: ..

Como podemos observar en la imagen, puede haber campos que ocupen mas de una línea, por lo que se debe empezar cada una de las líneas de continuación con un espacio o tabulador. Además, en los nombres de campos se distingue entra mayúsculas y minúsculas: todos los utilizados por R deben escribirse en mayúscula. Para lograr la máxima portabilidad, este archivo debe estar escrito por completo en ASCII, o en su defecto, contener un campo ‘Codification’. Algunos campos opcionales toman valores lógicos (“si”, “true”, “no”, “false”).

En este archivo, encontramos una serie de campos que son de carácter obligatorio, entre los que se encuentran:

- ‘Package’: este campo da el nombre del paquete. Debe contener solo letras, números y puntos, debe contener mínimo dos caracteres y empezar por una letra, además de no acabar en punto. Se ha de recordar que, si se requiere de una explicación, esto debe hacerse en el campo ‘Description’, que explicaremos a continuación.
- ‘Version’: este campo proporciona la versión actual del paquete. Se trata de una secuencia de al menos dos (y usualmente tres) enteros no negativos separados por puntos o guiones individuales.
- ‘License’: se trata de un campo fundamental, ya que es muy importante incluir información de licencia en el paquete, ya que puede que ni siquiera sea legalmente correcto para que otros distribuyan copias del paquete y lo usen si no la tiene. Las herramientas de administración de paquetes usan licencias de “software de fuente abierta o libre”, para que no haya restricciones en el uso o compartición del paquete. En este campo se debe especificar la licencia del paquete de forma estandarizada, y las alternativas se indican mediante barras verticales. Concretamente, las especificaciones individuales debes ser una de:
 - una especificación corta ‘estandar’ de las siguientes: GPL-2, GPL-3, LGPL-2, LGPL-2.1, AGPL-3, Artistic-2.0, BSD_2_clause, BSD_3_clause MIT. Está contenido en el subdirectorio share/licenses de la fuente R o en el directorio de inicio.
 - nombres o abreviaturas de otras licencias contenidas en la base de datos de la licencia en el archivo share/licenses/license.db en el directorio R de origen o de inicio. Para licencias versionadas, también se puede especificar el nombre seguido de la versión o combinar una abreviatura existente y la versión con un guion. Las abreviaturas GPL y LGPL son ambiguas y, en general, se refieren a cualquier versión de la licencia, pero es mejor no usarlos.
 - una de las cadenas ‘file LICENSE’ o ‘file LICENCE’ referido a un archivo llamado LICENSE O LICENCE en el paquete.

- la cadena ‘Unlimited’, lo que significa que no hay restricciones en la distribución o uso de otra que aquellos impuestos por leyes relevantes (incluyendo los derechos de autor).

Hay que asegurarse de que la licencia que se ha escogido también cubra todas las dependencias (incluidas las propias dependencias del sistema) del paquete. Es muy importante que cualquier restricción en el uso de tales dependencias sean evidentes para las personas que leen el archivo DESCRIPTION. El archivo opcional LICENSE/LICENCE contiene una copia de la licencia del paquete. Para evitar cualquier confusión solo incluye dicho archivo si se menciona en el campo ‘License’ del archivo DESCRIPTION. Por último, los campos ‘License_is.FOSS’ y ‘License_restricts_use’ puede tener valores ‘Yes’ o ‘No’ si se sabe que el archivo LICENSE restringe los usuarios o el propio uso del paquete, o se sabe que no. Además, algunas licencias “estándar” son más bien plantillas de licencia que necesitan información adicional que se completa añadiendo ‘+ file LICENSE’.

- ‘Description’: este campo debe dar una descripción completa de lo que hace el paquete. Se pueden usar varias oraciones, pero todas en un solo párrafo. Esta descripción debe estar escrita para ser entendida por todos los lectores previstos. Además, es buena práctica no comenzar con el nombre del paquete, “Este paquete” o algo parecido. Cuando queramos incluir alguna cita, se deben incluir entre comillas dobles (incluidos los títulos de libros y artículos), y solo citas para uso no inglés, incluidos nombres de otros paquetes y software externo. Como hemos mencionado anteriormente, este campo también se puede usar para explicar el nombre del paquete si fuera necesario.
- ‘Title’: este campo debe dar una breve descripción de paquete. Debe usar el título del caso (usando mayúsculas para las palabras importantes), no se tiene que usar ningún marcado, no tiene ninguna línea de continuación, y no termina en un periodo. No debe repetirse el nombre del paquete: a menudo se usa con el prefijo del nombre.
- ‘Author’: describe quien escribió el paquete. Se trata de un campo de texto plano destinado para lectores humanos, pero que no debe ser apta para el procesamiento automático. Se debe tener en cuenta que todos los contribuyentes importantes deben ser incluidos.
- ‘Maintainer’: este campo obligatorio debe dar un nombre único seguido de una dirección de correo electrónico válida (RFC 2822) en corchetes angulares. En ningún caso debe terminar en un punto o en una coma. Además, este campo es lo que es informado por la función maintainer y utilizado por bug.report. Para un paquete CRAN debería ser una persona, no una entidad corporativa o lista de correo: se debe asegurar que sea una dirección de correo válida y lo siga siendo durante el tiempo de uso o vida del paquete.

Los campos ‘Author’ y ‘Maintainer’ se pueden omitir si se proporciona el campo ‘Authors@R’ adecuado. En este caso, se proporciona una descripción refinada y legible por la máquina del paquete ‘Authors’ a través del código R adecuado (especificando las funciones precisas), al contrario que en los campos que sustituye que debían contener texto legible por humanos. El resultado es la creación de un objeto de la clase ‘person’ ya sea por la llamada de una persona o una serie de llamadas, cuando son varios los autores, concatenado por ‘c()’ como se puede ver en la imagen del comienzo de la sección. Los roles pueden incluir:

- ‘aut’ (autor): para autores completos,
- ‘cre’ (creador): para el mantenimiento del propio paquete,
- ‘ctb’ (colaborador): para otros contribuyentes, y
- ‘cph’ (propietario de los derechos de autor) entre otros

No se asume ningún rol por defecto. La información de citas de paquetes generados aprovecha esta especificación. Los campos ‘Author’ y ‘Maintainer’ se generan automáticamente desde allí si es necesario al construir o instalar. A continuación, explicamos los demás campos que pueden aparecer en el archivo DESCRIPTION, pero que ya no son obligatorios, entre los que destacan:

- ‘Copyright’: campo opcional en el que los titulares de los derechos de autor no son los autores. Si es necesario, puede estar referido a un archivo de instalación: lo normal es usar el archivo inst/COPYRIGHTS.
- ‘Date’: este campo opcional indica la fecha de lanzamiento de la versión actual del paquete. Es recomendable utilizar el formato “aaaa-mm-dd” conforme es recogido en el modelo ISO 8601.
- ‘Depends’: dicho campo proporciona una lista de nombres de paquetes separados por comas de los que depende este paquete; es decir, que para parte de la funcionalidad del paquete construido existe un paquete o una serie de paquetes que deben ser cargados para que puedan utilizarse correctamente. Estos paquetes se adjuntarán antes del paquete actual, cuando la biblioteca sea llamada o sea requerida. Cada nombre de paquete puede estar seguido opcionalmente por un comentario entre paréntesis que especifique el requisito de la versión. Este comentario debe contener un operador de comparación, espacios en blanco y un número de versión válida. (p.ej.: MASS (\geq 3.1-2.0)). Este campo también puede especificar una dependencia de una determinada versión de R; es decir, cuando el paquete solo funciona con una cierta versión de R. No tiene sentido declarar una dependencia de R sin una especificación de versión, ni en la base del paquete. Un paquete o ‘R’ puede aparecer más de una vez en este campo, para dar los límites superiores e inferiores en versiones aceptables. Además, no es recomendable usar una dependencia de R con el tercer dígito de la versión que no sea igual a cero. Si se hace con paquetes de

los que otros dependen, los otros paquetes se volverán inutilizables en versiones anteriores de la serie. Las instalaciones de R INSTALL verifican si la versión de R usada es lo suficientemente reciente para el paquete que se ha instalado, y se adjuntará la lista de paquetes que se especifica antes del paquete actual. Este campo debería utilizarse en la actualidad con poca frecuencia, solo para aquellos paquetes destinados a ponerse en la ruta de búsqueda para que sus instalaciones estén disponibles para el usuario final, y no para el paquete en sí.

- ‘Imports’: enumera los paquetes cuyos espacios de nombres se importan (como es especificado en el archivo NAMESPACE) pero que no es necesario adjuntar. Concretamente, este campo incluirá todos los paquetes estándar que se utilizan. En este campo no deben estar los paquetes que ya han sido nombrados en el campo ‘Depends’. Los requisitos de versión se pueden especificar y son verificados cuando el espacio de nombres está cargado. Aquí deberían aparecer aquellos paquetes cuyo espacio de nombres solo es necesario para cargar el paquete utilizando library(pkgname). No debe contener paquetes que no se importen, ya que todos los paquetes enumerados en este campo deben ser instalados para que se instale el paquete actual.
- ‘Suggests’: este campo enumera los paquetes que no son estrictamente necesarios, usando la misma sintaxis que en el campo ‘Depends’. Esto incluye tanto los paquetes utilizados en ejemplos, pruebas y viñetas, como en los paquetes cargados en el cuerpo de funciones. Los requisitos de versión pueden especificarse, pero deben ser verificados por el código que usa el paquete.
- ‘Enhances’: enumero los paquetes “mejorados” por el paquete en cuestión; es decir, por proporcionar métodos para las clases de estos paquetes o, por así decirlo, formas de manejar objetos a partir de estos paquetes. En este campo, los requisitos de versión se pueden especificar, pero actualmente no se utilizan. Tales paquetes no se pueden requerir para verificar el paquete: cualquier prueba que los use debe ser condicional en la presencia del paquete.
- ‘LinkingTo’: este campo se utiliza cuando un paquete desea hacer uso de los archivos de encabezado situados en otros paquetes. Se debe declarar una lista separada por comas y puede contener un requisito de versión que se verifica durante la instalación. Especificar en este campo un paquete es suficiente si se trata de encabezados de C++ que contienen el código fuente o la vinculación estática se realiza en la instalación. Además de registrar las rutinas C para ser llamadas por R, a veces puede ser bastante útil para un paquete para hacer que algunas de sus rutinas C estén disponibles para ser llamadas por código C en otro paquete.
- ‘Additional_repositories’: se trata de una lista separada por comas de URL de repositorio donde los paquetes nombrados en los otros campos se

pueden encontrar. Actualmente, lo usa la verificación de R CMD (check) para comprobar que los paquetes se pueden encontrar, al menos como paquetes fuente que se pueden instalar en cualquier plataforma.

- ‘SystemRequirements’: en este campo se deben incluir las dependencias externas al sistema R, que posiblemente se encuentre ampliación en un archivo README separado.
- ‘BugReports’: este puede contener una URL a la cual se deben presentar los informes de errores del paquete. Esta URL será usada por bug.report, en vez de enviar un correo electrónico al mantenimiento, ya que intentará extraer una dirección de correo electrónico abriendo la URL en un navegador (a partir de la versión de R 3.4.0).
- ‘Priority’: para paquetes base y recomendados contenidos en la distribución fuente de R o disponible de CRAN y que se recomiendan que se incluyan en cada distribución binaria de R, deben poseer este campo con los valores ‘base’ o ‘recommended’, respectivamente.
- ‘Collate’: este campo se utiliza para controlar el orden de intercalación de los archivos de código R en un paquete cuando estos se procesan para la instalación del paquete.
- ‘LazyData’: este campo lógico controla si los datasets R usan la carga lenta o diferida. El campo ‘LazyLoad’ era usado en versiones anteriores, ahora se ignora.
- ‘KeepSource’: es un campo lógico que controla si el código del paquete se obtiene utilizando keep.resource = TRUE/FALSE.
- ‘ByteCompile’: este campo lógico controla si el código del paquete debe compilarse en bytes durante la instalación. Su valor predeterminado es byte-compile.
- ‘Biarch’: este campo se utiliza en Windows para seleccionar la opción INSTALL – force-biarch para este paquete.
- ‘BuildVignettes’: se trata de otro campo lógico donde se puede establecer un valor falso para detener la construcción de R para CMD intentando construir las viñetas, así como para evitar que la comprobación de R para CMD pruebe esto. Solo puede usarse excepcionalmente si los archivos PDF incluyen cifras grandes que no son parte de las fuentes del paquete, por ejemplo.
- ‘VignetteBuilder’: estos nombres de campo (en una lista separada por comas) son para paquetes que proporcionan un motor para la construcción de viñetas, incluyendo el paquete actual o los enumerados en ‘Depends’, ‘Suggests’ o ‘Imports’. Más adelante explicaremos lo que son las viñetas.

- ‘Encoding’: este campo especifica una codificación cuando el archivo DESCRIPTION no está completamente escrito con caracteres ASCII. Además de usarse como la codificación del archivo DESCRIPTION en sí, se hace para los archivos R y NAMESPACE, y como la codificación predeterminada de los archivos .rd. Especificar una codificación distinta supone que el paquete sea menos portable, ya que debe ejecutar el constructor de R para CMD (build) en una configuración regional usando dicha codificación.
- ‘NeedsCompilation’: este campo especifica si el paquete contiene código que deba compilarse o no (valores “si” o “no”). Esto es utilizado por install.packages (type = “both”) en las versiones de R superiores a la 2.15.2 en plataformas donde los paquetes binarios son la norma: establecido normalmente por R CMD build o el repositorio suponiendo que la compilación es necesaria sí y solo si el paquete tiene un directorio src.
- ‘OS_type’: es un campo que especifica los sistemas operativos a los que está destinado el paquete.
- ‘Type’: especifica el tipo de paquete. Si este campo falta, se supone que es ‘Package’, pero actualmente se reconoce otro tipo y antiguamente también solía haber un tipo ‘Translation’. ‘Frontend’ es un mecanismo bastante general para agregar nuevos front-ends. Si se encuentra un archivo de configuración en el directorio del nivel superior del paquete que se ejecuta, y a continuación un archivo Makefile, se llama a make. Si R CMD INSTALL –clean se utiliza, se llama a make clean y no se toma ninguna otra acción. Además, R CMD build puede empaquetar este tipo de extensión, pero R CMD check verificara el tipo y lo salta. Bastantes paquetes de este tipo necesitan permiso de escritura para el directorio de instalación de R.
- Se pueden agregar también clasificaciones de materias para el contenido del paquete utilizando los campos ‘Classification/ACM’ o ‘Classification/ACM-2012’ (clasificación de computación), ‘Classification/JEL’ (el Diario de Sistema de Clasificación de Literatura Económica), o ‘Classification/MSA’ o ‘Classification/MSA-2010’ (Clasificación de la American Mathematical Society). Estas clasificaciones deber ser listas separadas por comas de los códigos de clasificación respectivos.
- ‘Language’: este campo adicional puede utilizarse para indicar si la documentación del paquete se encuentra en otro idioma que no sea el inglés. Debería tratarse de una lista separada por comas de las etiquetas de idioma tal y como están definidas actualmente por el RFC 5646.
- ‘RdMacros’: se suele utilizar este campo para contener una lista de paquetes separados por comas de la cual el paquete actual importará definiciones de macro Rd. Estos paquetes deben estar nombrados también en los campos ‘Imports’, ‘Suggests’ o ‘Depends’. Las macros en estos paquetes se importarán después de las macros del sistema, en el orden enumerado en

el propio campo que estamos tratando, antes de cualquier definición de macro que este cargado en el paquete actual.

- ‘Built’ o ‘Packaged’: estos campos no deben aparecer, ya que son agregados por herramientas de gestión de paquetes.
- Por último, algunos repositorios (incluidos CRAN y R-forge) agregan sus propios campos.

3.3.5 El archivo NAMESPACE

Este archivo especifica el espacio de nombres para un paquete en el directorio de nivel superior del paquete, ya que R posee un sistema de gestión del espacio de nombres para el código en paquetes. Este sistema permite al escritor del paquete especificar qué variables del paquete deben exportarse para que estén disponibles para usuarios del propio paquete, y qué variables deberían importarse de otros paquetes.

Por lo tanto, este archivo contiene directivas del espacio de nombres que describen las importaciones y exportaciones del espacio de nombres. Las directivas adicionales registran los objetos compartidos que se cargaran, y cualquier método del estilo S3 que se proporcionan. El contenido del archivo es parecido al código R, aunque no se procesa como código R. Solo el procesamiento condicional muy simple de estas declaraciones son implementadas.

Los paquetes se cargan y se adjuntan a la ruta de búsqueda llamando a `library` o `require`. Solo las variables exportadas son colocadas en el marco adjunto. Cargar un paquete que importe variables de otros paquetes causara que estos otros paquetes tengan que ser cargados también si no han sido cargados previamente, pero estas cargas implícitas no harán que sean colocados en la ruta de búsqueda. Entonces, el código utilizado en un paquete solo puede depender de objetos que se encuentren en su mismo espacio de nombres y sus importaciones (incluido el espacio de nombres base) siendo visibles.

Los espacios de nombres se sellan una vez que se cargan, esto quiere decir que las importaciones y exportaciones no pueden cambiarse, y que los enlaces de variables internas no se pueden cambiar. Esto permite una estrategia simple de implementación para el mecanismo del espacio de nombres, además del análisis del código y herramientas de compilación para poder identificar con precisión la definición correspondiente a una referencia de la variable global en el cuerpo de una función.

Por último, cabe mencionar que el espacio de nombres controla la estrategia de búsqueda para las variables usadas por funciones en el paquete. Si no es encontrada localmente, R busca primero el espacio de nombres del paquete, después las importaciones, a continuación el espacio de nombres base y finalmente la ruta de búsqueda normal.

¿Cómo se especifican las importaciones y exportaciones? Las exportaciones son especificadas utilizando la directiva de exportación en el archivo `NAMESPACE`. La directiva tiene la forma `"export(var1,var2)"`, que especifica que las variables `"var1"` y `"var2"` tienen que exportarse. Si hay muchas variables que exportar, será mejor especificar los nombres de las variables para exportar con una expresión regular usando `"exportPattern"`, que exporta todas las variables que no comienzan en un punto. En cuanto a las importaciones, los paquetes importan implícitamente el espacio de nombres base. Las variables exportadas desde otros paquetes con espacios de nombres deben ser importados explícitamente usando las directivas `"import"` e `"importFrom"`, con la directiva `"import(paquete1, paquete2)"` que indica que deben importarse todas las variables exportadas en los paquetes `"paquete1"` y `"paquete2"`. Cuando solo sea necesario importar algunas de las variables exportadas por cierto paquete, se usará la directiva `"importFrom(paquete1,var1,var2)"` que especifica que las variables exportadas `"var1"` y `"var2"` del paquete `"paquete1"` deben ser importadas. Por último, también es posible exportar variables desde un espacio de nombres que ha sido importado desde otros espacios de nombres, lo cual tiene que hacerse explícitamente y no a través de `"exportPattern"`.

¿Cómo se registran métodos S3? El método estándar para el envío del estilo S3 `"UseMethod"` puede fallar a la hora de localizar los métodos definidos en un paquete que ha sido importado pero no se adjunta a la ruta de búsqueda. Para asegurarse de que estos métodos estén disponibles, los paquetes que definen los métodos deben asegurar que los genéricos sean importados y registrar los métodos utilizando las directivas `"S3method"`. Si un paquete define una función `"print.foo"` destinada a ser utilizada como un método de impresión para la clase `"foo"`, entonces la directiva `"S3method(print,foo)"` asegura que el método está registrado y disponible para el envío `"UseMethod"`.

Load hooks: Son una serie de `"hooks"` o ganchos que son llamados a medida que los paquetes se cargan, se adjuntan, se desconectan y se descargan. Al ser distintas la carga de la unión, se proporcionan ganchos separados para cada una, `".onLoad"` y `".onAttach"` respectivamente. Ambos toman como argumentos `"libname"` y `"pkgname"`, que deberían estar definidos en el espacio de nombres pero no exportados.

La directiva `useDynLib`. Las directivas `"useDynLib"` permiten objetos compartidos que necesitan ser cargados. El archivo `NAMESPACE` puede contener una o más de estas directivas. Se utiliza de la forma `"useDynLib(foo)"`, lo cual registra el objeto compartido `"foo"` para ser cargado con `"library.dynam"`. La carga del objeto registrado se produce después de que se haya cargado el código del paquete, y antes de ejecutar la función de `"load hook"`. Esta directiva se puede utilizar cuando el paquete solo necesita cargar una función `"load hook"` para cargar un objeto compartido. Esta directiva también acepta los nombres de las rutinas nativas que se usarán en R a través de las funciones de interfaz

.C, .Call, .Forran y .External, dándose como argumentos adicionales a la hora de llamar a la directiva ("useDynLib(foo,miRutina,otraRutina)"). Los símbolos nativos se resuelven cuando se carga el paquete y las variables R que identifican estos símbolos se agregan al espacio de nombres del paquete con estos nombres. Estos pueden usarse en las llamadas .C, .Call, .Forran y .External en lugar del nombre de la rutina y el argumento del PACKAGE.

3.3.6 El subdirectorio 'R'

Este archivo contiene sólo los archivos de código R. Estos archivos de código deben empezar con una letra o un dígito ASCII (en mayúsculas o minúsculas), y tiene una de las siguientes extensiones: .R, .S, .q, .r o .s. De todas estas, la extensión más utilizada es la .R ya que parece no ser utilizada por ningún otro software. Los objetos R deben crearse mediante asignaciones, ya que debería de ser posible leer en dichos archivos utilizando "source()", por lo que no debe haber conexión entre el nombre del archivo y los objetos R creados. Los archivos de código R solo deben asignar directamente objetos R. Si se necesitan cálculos para crear estos objetos, estos pueden usar el código "earlier" en el paquete, más las funciones en los paquetes de "Depends", siempre que los objetos creados no dependan de esos paquetes excepto a través de las importaciones del espacio de nombres.

Aparte de las extensiones de archivos mencionadas anteriormente, también se permiten los archivos que terminan en .in. Esto se puede hacer para permitir que un script de configuración genere los archivos adecuados.

Cabe mencionar que aunque en los archivos de código solo se deben usar caracteres ASCII, se aceptan otros caracteres en los comentarios, pero es posible que no sean legibles en una configuración regional, por ejemplo. Aquellos nombres de objetos que contienen caracteres que no son ASCII normalmente fallan cuando el paquete está instalado.

3.3.7 El subdirectorio 'src'

Las fuentes y las cabeceras del código compilado están en este subdirectorio, más opcionalmente un archivo "Makevars" o "Makefile". Cuando se lleva a cabo el R CMD INSTALL, make se utiliza para controlar la compilación y vincular a un objeto compartido para cargar en R. Se recomienda utilizar la extensión .h para las cabeceras. Los archivos en este directorio no deben ocultarse, ya que en algunas versiones de R se ignorarán.

Si nuestro código necesita depender de la plataforma, existen ciertas definiciones que se pueden utilizar en C o C++: en todas las compilaciones de Windows se definirá '_WIN32', en Windows de 64 bits también se crea '_WIN64', y en macOS se define '__APPLE__'. Se pueden ajustar las reglas predeterminadas haciendo cambios en "macros" en un archivo en src/Makefile.

Algunos paquetes utilizan el directorio 'src' para fines distintos de crear un objeto compartido, como crear ejecutables. Estos paquetes deben tener los archivos src/Makefile y src/Makefile.win. En casos muy excepcionales, los paquetes pueden crear archivos binarios distintos de los objetos compartidos /DLL en el subdirectorio 'src'. Si un paquete desea instalar otros binarios, debe proporcionar un script R src/install.libs.R que se ejecutara como parte de la instalación en el directorio de compilación 'src' en lugar de copiar los objetos compartidos /DLL.

3.3.8 El subdirectorio 'demo'

El subdirectorio 'demo' se utiliza para scripts R que demuestran algunas de las funcionalidades del paquete. Las demostraciones pueden ser interactivas y no se verifican automáticamente; si se desean hacer pruebas, se debe usar el código en el directorio 'tests' para lograr esto. Los archivos de script deben comenzar con una letra mayúscula o minúscula y tener la extensión .R o .r.

Este subdirectorio también debe tener un archivo "00Index" con una línea para cada demostración, incluyendo su nombre y una breve descripción separada por un espacio de tabulador o al menos tres espacios. Cabe decir que este archivo de índice no se genera automáticamente. También se debe tener en cuenta que una demostración no tiene una codificación específica, por lo que debe ser un archivo ASCII. La función "demo()" usará la codificación del paquete si hay una, pero es tremendamente útil para comentarios que no sean ASCII.

3.3.9 El subdirectorio 'inst'

El contenido de este subdirectorio se copiará recursivamente en el directorio de instalación. Los subdirectorios de 'inst' no deberían interferir con los usados por R (R, data, demo, exec, libs, man, help, html y Meta). La copia de la que hemos hablado anteriormente, ocurre nada mas construirse el subdirectorio 'src', por lo que su Makefile puede crear archivos para su instalación. Se puede especificar una lista de patrones de exclusión en el archivo, que deberían ser expresiones regulares parecidas a Perl.

hay que tener en cuenta que con las excepciones de INDEX, LICENCE/ LICENSE y NEWS, los archivos de información en el nivel superior del paquete no se instalará y, por lo tanto, no serán vistos por los usuarios de Windows y los paquetes compilados de macOS. Entonces, cualquier archivo de información que se desea que llegue hasta el usuario final, se debe incluir en el subdirectorio 'inst'.

Los principales archivos que se suele agregar a 'inst' son: 'CITATION' para usar con la función de cita, 'NEWS.Rd' para ser utilizado por la función de noticias, y 'AUTHORS' o 'COPYRIGHTS' para especificar los autores o los titulares de derechos de autor cuando esto es demasiado complejo para ponerlo

en el archivo 'DESCRIPTION'.

3.4 Revisando y construyendo paquetes

En este apartado se va a explicar como utilizar las herramientas R CMD check y R CMD build, pero antes de utilizarlas se debe comprobar que el paquete que queremos construir se pueda instalar, lo que se hace probando a cargarlo.

Si el paquete especifica una codificación en el archivo 'DESCRIPTION', se deben ejecutar estas herramientas en un lugar que haga uso de esa codificación, ya que es posible que no funcionen o funcionen incorrectamente en otras configuraciones regionales.

Para los usuarios de Windows, R CMD build puede hacer uso del conjunto de herramientas de Windows si está presente y en su misma ruta, y es requerido para paquetes que lo necesitan para instalar y por ejemplo necesitan construir viñetas.

3.4.1 Construyendo tarballs de paquetes

Como hemos comentado en secciones anteriores, los paquetes se pueden distribuir en forma de fuente como "tarballs", (archivos .tar.gz) o en formato binario. El formato fuente se puede instalar en todas las plataformas con las herramientas adecuadas y es la forma habitual de los sistemas tipo Unix; la forma binaria, en cambio, es específica de la plataforma, por lo que la distribución más común es la de "tarball" para las plataformas Windows y macOS.

Con R CMD build, el constructor de paquetes de R, se pueden construir "tarballs" de paquetes desde sus fuentes. Es recomendable construir los paquetes acordes a la versión actual de R para su lanzamiento. Se recomienda antes de crear el paquete en el formato tar estándar gzipped comprobar si el paquete esta construido en principio de forma correcta, esto se hace con R CMD check.

A no ser que se invoque el R CMD build con la opción `-no-build-vignettes`, intentará crear o recrear si ya habian sido creadas las viñetas que se hayan especificado en el paquete.

Por otro lado, una de las comprobaciones que ejecuta R CMD build es para los directorios fuente que se encuentren vacíos.

Si se desea conocer el resto de opciones de R CMD build introducir la opción `-help` para obtener más información sobre el constructor de R.

3.4.2 Construyendo binarios de paquetes

Los binarios son copias comprimidas de versiones instaladas de los paquetes, los cuales contienen compilados bibliotecas compartidas en lugar de código fuente C, C++ o Fortran, y se incluyen las funciones R en su forma instalada.

El problema de estos binarios de paquetes consiste en que el formato y el nombre de archivo son específicos de cada plataforma (para Windows .zip, para macOS .tgz, etc).

El método recomendado para construir estos binarios es: `R CMD INSTALL -build paquete`, donde `paquete` es el nombre de una fuente tarball o la ubicación del directorio de origen del paquete que se construirá. Esto se lleva a cabo instalando primero el paquete, y luego empaquetando los binarios instalados en el paquete archivo binario de paquete correspondiente para la plataforma particular.

Para evitar que `R CMD INSTALL -build` intente instalar el paquete en el árbol de biblioteca predeterminado para la instalación local de R, y este no tenga permiso de escritura (lo que generaría que el paquete no se instalará y no se creará el binario), se puede añadir la opción: `R CMD INSTALL -l ubicación -build paquete`, donde `ubicación` es el directorio elegido con acceso de escritura. Para más información sobre las diversas opciones para `R CMD INSTALL` se puede añadir la opción `-help` a la herramienta.

3.5 Viñetas

Las viñetas del paquete son documentos en PDF o HTML obtenidos de archivos fuente de texto literario planos (sin formatos) de los cuales R sabe cómo extraer el código R y crear la salida. Este trabajo lo hacen los motores de las viñetas, usando las funciones “tangle” (enredar) y “weave” (tejer) respectivamente. Sweave es el motor predeterminado proporcionado por la distribución R, aunque actualmente otros motores de viñetas son compatibles. Se recomienda utilizar el formato PDF, aunque el formato puede ser arbitrario, para que todos los usuarios de las diversas plataformas puedan leerlos. La ubicación estándar para estos documentos es el subdirectorio `inst/doc` de un paquete fuente, los documentos se copiarán en los documentos del subdirectorio cuando se instale el paquete. Los punteros desde los índices de ayuda del paquete hasta los documentos instalados se crean automáticamente. También, los nombres de los archivos deben comenzar por un carácter ASCII para ser accesible desde un navegador.

Entonces, las viñetas de paquete tienen sus orígenes en el subdirectorio ‘`vi-
gnettes`’ del paquete fuente. Estas fuentes de viñetas, normalmente reciben la extensión de archivo `.Rnw` o `.Rtex`, pero también son reconocidas `.Snw` y `.Stex`. Además, Sweave permite la integración de documentos LATEX.

Las viñetas de paquete se prueban mediante `R CMD check`, ejecutando todos los fragmentos de código R que contienen (excepto los marcados para la no evaluación). El directorio de trabajo de R para todas las pruebas de viñetas en `R CMD check` es una copia del directorio fuente de la viñeta. Todos los archivos necesarios para ejecutar el código R en la viñeta deben estar accesibles en la jerarquía `inst/doc` del paquete fuente, o usando llamadas a `'system.file()'`. Todos los demás archivos necesarios para volver a hacer las viñetas deben estar en el directorio fuente `'vignettes'`. Esta comprobación verificará que la producción de viñetas ha tenido éxito.

`R CMD build` creará automáticamente las viñetas en `inst/doc` para su distribución con las fuentes del paquete. El usuario puede utilizar paquetes R privados, instantáneas de pantalla y extensiones LATEX que solo estén disponibles en su máquina. Por defecto, `R CMD build` ejecutará Sweave en todos los archivos fuente de viñetas Sweave en `'vignettes'`. Si Makefile se encuentra en el directorio fuente de la viñeta, el constructor de paquetes de R intentará runear el make después de que se ejecute Sweave; de lo contrario, `texi2pdf` se ejecutará en cada archivo `.tex` producido. El primer objetivo de Makefile es la creación de archivos PDF/HTML y después limpiar o eliminar aquellos archivos que no deben aparecer en el archivo final del paquete (debe tener un objeto clean para ello).

En el momento de la instalación, se crea automáticamente un índice HTML para todas las viñetas del paquete a partir de las sentencias `VignetteIndexEntry`, a menos que exista un archivo `'index.html'` en el directorio `inst/doc`. Este índice está vinculado desde el índice de ayuda de HTML para el paquete. Si se proporciona el archivo `inst/doc/index.html`, este debe contener enlaces relativos solo a archivos en el directorio del documento instalado (o a archivos de ayuda HTML o DESCRIPTION), y sea válido el HTML según lo confirmado a través del servicio de validación de W3C.

Por último, para instalar cualquier otro archivo del directorio de viñetas, incluya un archivo de `vignettes/.install.extras` que especifica estas como expresiones regulares tipo Perl en una o más líneas.

La viñeta que se ha generado para el paquete ha sido la siguiente:

learningRlab

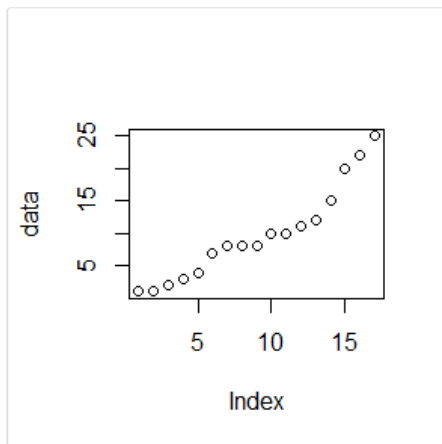
There are three families of fuctions in LearningRlab:

1. Main functions: these functions return the result of performing the process represented with the function.
2. Explained fuctions: these fuctions returns the process itself to get the result, with the result.
3. User Interactive Functions: these functions maintain an interactive contact with the user to guide him in the resolution of the represented function.

Main Functions:

To explain the use of each function, we present a dataset to work with them:

```
data <- c(1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25)
plot(data);
```



The arithmetic mean calculus function:

```
mean_(data)
#> [1] 9.823529
```

The geometric mean calculus function:

```
geometricMean_(data)
#> [1] 6.911414
```

The mode calculus function:

```
mode_(data)
#> Factor appears 3 times in the vector.
#> Unique mode
#> [1] 8
```

The median calculus function:

```
median_(data)
#>
#> Sorted vector: 1 1 2 3 4 7 8 8 8 10 10 11 12 15 20 22 25
#> [1] 8
```

33

The standard deviation calculus function:

```
standardDeviation_(data)
#> [1] 6.989364
```

```
variance_(data)
#> [1] 48.85121
```

The quartiles calculus function:

```
quartile_(data)
#>   Q1   Q2   Q3
#> 4.25 8.50 12.75
```

The percentile calculus function:

```
percentile_(data)
#> Percentile[ 10 ] = 1
#> Percentile[ 20 ] = 3
#> Percentile[ 30 ] = 7
#> Percentile[ 40 ] = 8
#> Percentile[ 50 ] = 8
#> Percentile[ 60 ] = 10
#> Percentile[ 70 ] = 11
#> Percentile[ 80 ] = 15
#> Percentile[ 90 ] = 22
#> Percentile[ 100 ] = 25
```

The absolute frequency calculus function:

```
frequency_abs(data,1)
#> [1] 2
```

The relative frequency calculus function:

```
frequency_relative(data,20)
#> [1] 0.05882353
```

The absolute acumulated frequency calculus function:

```
frequency_absolute_acum(data,1)
#> [1] 2
```

The relative acumulated frequency calculus function:

```
frequency_relative_acum(data,20)
#> [1] 0.8823529
```

Explained Functions:

For each main function, there are an explained function to see the calculus process:

- arithmetic mean:

```

explain.mean(data)
#>
#> __MEAN CALCULUS__
#>
#> Formula ->  $(x_1 + x_2 + \dots + x_n) / \text{num\_elements}$ 
#>
#> The mean of a dataset is calculated by adding each element of the dataset and dividing the result by the number of
elements in the dataset. We'll give the user an example for better comprehension.
#>
#> __Use Example__
#>
#> First of all, we need to know the content of the dataset/vector of numbers
#>
#> The content of the vector is: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> Now we need to add each element of the vector/dataset
#> The sum of the elements is: 167
#>
#> Next step, get the number of elements that we've examined
#> The length of the vector is 17 elements
#>
#> Formula applied ->  $167 / 17 = 9.82352941176471$ 
#> Now try by your own! :D
#>
#> Use interactive.mean function to practice.

```

- **geometric mean:**

```

explain.geometricMean(data)
#>
#> __GEOMETRIC MEAN CALCULUS__
#>
#> Formula ->  $(x_1 * x_2 * \dots * x_n)^{(1 / \text{num\_elements})}$ 
#>
#> The geometric mean of a dataset is calculated by multiplying each element of the dataset and raising the result to 1 divided
by the number of elements in the dataset. We'll give the user an example for better comprehension.
#>
#> __Use Example__
#>
#> First of all, we need to know the content of the dataset/vector of numbers
#>
#> The content of the vector is: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> Now we need to multiply each element of the vector/dataset
#> The product of the elements is: 1.87342848e+14
#>
#> Next step, get the number of elements that we've examined
#> The length of the vector is 17 elements
#>
#> Formula applied ->  $(1.87342848e+14)^{(1 / 17)} = 6.91141369632174$ 
#> Now try by your own! :D
#>
#> Use interactive.geometricMean function to practice.

```

- **mode:**

explain.mode(data)

```
#>
#> __MODE CALCULUS__
#>
#> Formula -> Most repeated value of [Data]
#>
#> The mode of a dataset is calculated by looking for the most repeated value in the dataset. If in a group there are two or
several scores with the same frequency and that frequency is the maximum, the distribution is bimodal or multimodal, that is, it
has several modes.
#>
#> __Use Example__
#>
#> First step : search the most repeated value
#>
#> The content of the vector is: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> Factor 8 appears 3 times in the vector.
#>
#> Second step : check the dataset looking for a value with the same maximum frequency
#>
#> If there are only 1 unique most repeated value, it is the mode.
#> If there are 2 values repeated with the same maximum frequency each value represents the mode. Bimodal dataset
#> If there are more than 2 values repeated with the same maximum frequency, it is a Multimodal dataset
#>
#> Now try by your own! :D
#>
#> Use interactive.mode function to practice.
```

- median:

explain.median(data)

```
#>
#> __MEDIAN CALCULUS__
#>
#> Formula ->  $1/2(n+1)$  where  $n$  -> vector size
#>
#> The median of a dataset is the value in the middle of the sorted data. It's important to know that the data must be sorted. If
the dataset has a pair number of elements, we should select both in the middle to add each other and get divided by two. If the
dataset has a no pair number of elements, we should select the one in the middle.
#>
#> __Use Example__
#>
#> First step : identify if the vector has a pair number of elements
#>
#> The content of the vector is: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#>
#> Second step: depending of the number of elements
#>
#> It has a ODD number of elements ( 17 )
#>
#> We take the 'n/2' approaching up element
#> The result is : 8
#> Now try by your own! :D
#>
#> Use interactive.median function to practice.
```

- standard deviation:

explain.standardDeviation(data)

```
#>
#> __STANDARD DEVIATION CALCULUS__
#>
#> Formula -> square_root ((Summation(each_element - mean)^2) / num_elements)
#>
#> The standard deviation of a dataset is calculated by adding the square of the difference between each element and the mean of the dataset. This sum will be dividing by the number of elements in the dataset and finally making the square root on the result. We'll give the user an example for better comprehension.
#>
#> __Use Example__
#>
#> First of all, we need to know the content of the dataset/vector of numbers
#>
#> The content of the vector is: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> The mean of dataset is... 9.82352941176471
#> The square of the difference between each number and the mean of dataset is: 77.85467 ,77.85467 ,61.20761 ,46.56055 ,33.91349 ,7.972318 ,3.32526 ,3.32526 ,3.32526 ,0.03114187 ,0.03114187 ,1.384083 ,4.737024 ,26.79585 ,103.5606 ,148.2664 ,230.3253
#> Now we need to add each element of the vector/dataset
#> The sum of the squares is: 830.470588235294
#>
#> Next step, get the number of elements that we've examined
#> The length of the vector is 17 elements
#>
#> Formula applied -> ( 830.4706 / 17 ) ^ (1/2) = 6.98936413936664
#> Now try by your own! :D
#>
#> Use interactive.standardDeviation function to practice.
```

- average absolute deviation:

explain.averageDeviation(data)

```
#>
#> __AVERAGE DEVIATION CALCULUS__
#>
#> Formula -> (Summation(abs(each_element - mean))) / num_elements
#>
#> The average deviation of a dataset is calculated by adding the absolute value of the difference between each element and the mean of the dataset. This sum will be dividing by the number of elements in the dataset. We'll give the user an example for better comprehension.
#>
#> __Use Example__
#>
#> First of all, we need to know the content of the dataset/vector of numbers
#>
#> The content of the vector is: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> The mean of dataset is... 9.82352941176471
#> The absolute value of the difference between each number and the mean of dataset is: 8.823529 ,8.823529 ,7.823529 ,6.823529 ,5.823529 ,2.823529 ,1.823529 ,1.823529 ,1.823529 ,0.1764706 ,0.1764706 ,1.176471 ,2.176471 ,5.176471 ,10.17647 ,12.17647 ,15.17647
#> Now we need to add each element of the vector/dataset
#> The sum of the squares is: 92.8235294117647
#>
#> Next step, get the number of elements that we've examined
#> The length of the vector is 17 elements
#>
#> Formula applied -> 92.82353 / 17 = 5.46020761245675
#> Now try by your own! :D
#>
#> Use interactive.averageDeviation function to practice.
```

- variance:

explain.variance(data)

```
#>
#> __ VARIANCE CALCULUS __
#>
#> Formula -> (Summation(each_element - mean)^2) / num_elements
#>
#> The variance of a dataset is calculated by adding the square of the difference between each element and the mean of the dataset. This sum will be dividing by the number of elements in the dataset. We'll give the user an example for better comprehension.
#>
#> __ Use Example __
#>
#> First of all, we need to know the content of the dataset/vector of numbers
#>
#> The content of the vector is: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> The mean of dataset is... 9.82352941176471
#> The square of the difference between each number and the mean of dataset is: 77.85467 ,77.85467 ,61.20761 ,46.56055 ,33.91349 ,7.972318 ,3.32526 ,3.32526 ,3.32526 ,0.03114187 ,0.03114187 ,1.384083 ,4.737024 ,26.79585 ,103.5606 ,148.2664 ,230.3253
#> Now we need to add each element of the vector/dataset
#> The sum of the squares is: 830.470588235294
#>
#> Next step, get the number of elements that we've examined
#> The length of the vector is 17 elements
#>
#> Formula applied -> 830.4706 / 17 = 48.8512110726644
#> Now try by your own! :D
#>
#> Use interactive.variance function to practice.
```

- **quartile:**

explain.quartile(data)

```
#>
#> __ QUARTILES CALCULUS __
#>
#> Formula -> (k * N) / 4 where k -> 1,2,3 and N -> vector size
#>
#> The quartile divides the dataset in 4 parts as equal as possible.
#>
#> __ Use Example __
#>
#> Step 1: The vector must be sorted.
#> 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#>
#> Step 2: Apply the formula (k * N) / 4 where 'k' is [1-3]
#>
#> Q1 -> (1 * 17) / 4 = 4.25
#> Q2 -> (2 * 17) / 4 = 8.5
#> Q3 -> (3 * 17) / 4 = 12.75
#>
#> Visualization with colors:
#> 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#>
#> Q1 -> 4.25 || Q2 -> 8.5 || Q3 -> 12.75 || Q4 -> onwards
#> Now try by your own! :D
#>
#> Use interactive.quartile function to practice.
```

- **percentile:**

explain.percentile(data)

```
#>
#> __PERCENTILES CALCULUS__
#>
#> Formula ->  $(k * N) / 100$  where  $k \rightarrow [1-100]$  and  $N \rightarrow$  vector size
#>
#> The percentile divides the dataset in 100 parts.
#> The percentile indicates, once the data is ordered from least to greatest, the value of the variable below which a given percentage is located on the data
#>
#> __Use Example__
#>
#> Step 1: The vector must be sorted.
#> 1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25
#>
#> Step 2: Apply the formula  $(k * N) / 100$  where 'k' is [1-100]
#>
#> We will calculate the percentiles 1,25,37,50,92 in this example
#>
#> Percentile 1 ->  $(1 * 17) / 100 = 0.17$ 
#> .Round up the value to locate it in the vector ->  $0.17 \sim 1$ 
#> ..In our data, the value is = 1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25
#>
#> Percentile 25 ->  $(25 * 17) / 100 = 4.25$ 
#> .Round up the value to locate it in the vector ->  $4.25 \sim 5$ 
#> ..In our data, the value is = 1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25
#>
#> Percentile 37 ->  $(37 * 17) / 100 = 6.29$ 
#> .Round up the value to locate it in the vector ->  $6.29 \sim 7$ 
#> ..In our data, the value is = 1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25
#>
#> Percentile 50 ->  $(50 * 17) / 100 = 8.5$ 
#> .Round up the value to locate it in the vector ->  $8.5 \sim 9$ 
#> ..In our data, the value is = 1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25
#>
#> Percentile 92 ->  $(92 * 17) / 100 = 15.64$ 
#> .Round up the value to locate it in the vector ->  $15.64 \sim 16$ 
#> ..In our data, the value is = 1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25
#>
#> Now try by your own! :D
#>
#> Use interactive.percentile function to practice.
```

- absolute frequency:

explain.absolute_frequency(data,10)

```
#>
#> __ABSOLUTE FREQUENCY CALCULUS__
#>
#> Formula ->  $N1 + N2 + N3 + \dots + Nk \rightarrow Nk = X$  (Where 'X' is the element we want to examine)
#>
#> The absolute frequency ( $N_i$ ) of a value  $X_i$  is the number of times the value is in the set ( $X_1, X_2, \dots, X_N$ )
#>
#> __Use Example__
#>
#> All we need to do is count the number of times that the element 10 appears in our data set
#>
#> Our data set: 1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25
#>
#> Now count the number of times that the element 10 appears: 2
#> 1,1,2,3,4,7,8,8,8,10,10,11,12,15,20,22,25
#> Now try by your own! :D
#>
#> Use interactive.absolute_frequency function to practice.
```

- relative frequency:


```

explain.relative_frequency(data,8)
#>
#> __RELATIVE FREQUENCY CALCULUS__
#>
#> Formula -> (Abs_freq(X) / N ) -> Where 'X' is the element we want to examine
#>
#> The relative frequency is the quotient between the absolute frequency of a certain value and the total number of data
#>
#> __Use Example__
#>
#> Step 1: count the number of times that the element 8 appears in our data set
#>
#> Our data set: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#>
#> Now count the number of times that the element 8 appears: 3
#> 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> Step 2: divide it by the length of the data set
#>
#> Solution --> relative_frequency = (absolute_frequency(x) / length(data)) = 3 / 17 = 0.176470588235294 .
#>
#> Now try by your own! :D
#>
#> Use interactive.relative_frequency function to practice.

```

- absolute acumulated frequency:

```

explain.absolute_acum_frequency(data,10)
#>
#> __ABSOLUTE ACUMULATED FREQUENCY CALCULUS__
#>
#> Formula -> Summation(abs_frequency <= X ) -> Where 'X' is the element we want to examine
#>
#> The absolute acumulated frequency is the sum of the absolute frequency of the values minors or equals than the value we want to examine
#>
#> __Use Example__
#>
#> Step 1: count the number of times that the elements minors or equals than 10 appears in our data set
#>
#> Our data set: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#>
#> Number of times that elements minors or equals to 10 appears = 11
#> 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> Solution --> absolute_frequency_acum = Summation(abs_frequency <= X) = 11 .
#>
#> Now try by your own! :D
#>
#> Use interactive.absolute_acum_frequency function to practice.

```

- relative acumulated frequency:

```

explain.relative_acum_frecuency(data,8)
#>
#> __RELATIVE ACUMULATED FREQUENCY CALCULUS__
#>
#> Formula -> (Summation(abs_frecuency <= X) / N ) -> Where 'X' is the element we want to examine
#>
#> The relative acumulated frequency is the quotient between the sum of the absolute frequency of the values minors or equals than the value we want to examine, and the total number of data
#>
#> __Use Example__
#>
#> Step 1: count the number of times that the elements minors or equals than 8 appears in our data set
#>
#> Our data set: 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#>
#> Number of times that elements minors or equals to 8 appears = 9
#> 1 ,1 ,2 ,3 ,4 ,7 ,8 ,8 ,8 ,10 ,10 ,11 ,12 ,15 ,20 ,22 ,25
#> Step 2: divide it by the length of the data set
#>
#> Solution --> relative_frecuency_acum = (Summation(abs_frecuency <= X) / length(data)) = 9 / 17 = 0.529411764705882
.
#>
#> Now try by your own! :D
#>
#> Use interactive.relative_acum_frecuency function to practice.

```

User Interactive Functions:

These functions are designed for the user to practice with them, and they are the following:

- interactive.mean()
- interactive.geometricMean()
- interactive.mode()
- interactive.median()
- interactive.standardDeviation()
- interactive.averageDeviation()
- interactive.variance()
- interactive.quartile()
- interactive.percentile()
- interactive.absolute_frecuency()
- interactive.relative_frecuency()
- interactive.absolute_acum_frecuency()
- interactive.relative_acum_frecuency()

4 DESCRIPCIÓN EXPERIMENTAL

En este apartado se va a abordar una explicación de la metodología y de los pasos seguidos a la hora de desarrollar el paquete ‘LearningRlab’. Nos centraremos en el desarrollo de la funcionalidad de ficheros .R y .c.

4.1 Desarrollo de ficheros R

La funcionalidad principal de este paquete es ayudar al usuario a introducirse en el mundo de la enseñanza estadística básica a nivel de enseñanza obligatoria secundaria, y al mismo tiempo familiarizarle con el uso de una herramienta tan potente como lo es el lenguaje R y su framework de trabajo RStudio.

Las funciones que se han desarrollado pueden dividirse en 4 pequeños grupos, y todas ellas se encuentran definidas en el archivo ‘Statistics.R’:

Funciones principales de cálculo

Funciones didácticas

Funciones interactivas

Funciones auxiliares

4.1.1 Funciones principales de cálculo

Dentro de este grupo, se encuentran las funciones que simplemente reciben los parámetros por parte del usuario y se realiza el cálculo deseado, pudiendo guardarse el valor dentro de una variable para su posterior uso o por si deseamos examinar el valor de forma individual. Las funciones que forman parte de este grupo son las siguientes:

mean_
mode_
median_
percentile_
quartile_
variance_
standardDeviation_
averageDeviation_
frequency_abs
frequency_relative
frequency_absolute_acum
frequency_relative_acum

4.1.2 Funciones didácticas

Dentro de este grupo, se encuentran las funciones diseñadas explícitamente para enseñar al usuario el proceso que se lleva a cabo para realizar el cálculo. Se han personalizado los ejemplos, haciendo que se adapte cada uno a los parámetros que introduzca el usuario a la función, de modo que podrá ver paso por paso como se calcula el valor deseado. Se han incluido efectos visuales como salida con colores o ‘plots’ de formulas. Las funciones que forman parte de este grupo son las siguientes:

```
explain.mean  
explain.mode  
explain.median  
explain.percentile  
explain.quartile  
explain.variance  
explain.standardDeviation  
explain.averageDeviation  
explain.frecuency_abs  
explain.frecuency_relative  
explain.frecuency_absolute_acum  
explain.frecuency_relative_acum
```

4.1.3 Funciones interactivas

Dentro de este grupo, se encuentran las funciones diseñadas para interactuar con el usuario y comprobar que se ha entendido el proceso que hay que llevar a cabo para obtener el calculo del valor necesario. Se pide una serie de datos al usuario y sobre ellos, se pide la solución correcta. Las funciones que forman parte de este grupo son las siguientes:

```
interactive.mean  
interactive.mode  
interactive.median  
interactive.percentile  
interactive.quartile  
interactive.variance  
interactive.standardDeviation  
interactive.averageDeviation  
interactive.frecuency_abs  
interactive.frecuency_relative  
interactive.frecuency_absolute_acum  
interactive.frecuency_relative_acum
```

4.1.4 Funciones auxiliares

Se ha creado este pequeño grupo para juntar todas aquellas funciones que proveen funcionalidad auxiliar al resto de funciones, pero que no encajan dentro de los grupos anteriores, por lo que se han separado en un módulo diferente. Las funciones que forman parte de este grupo son las siguientes:

```
drawVector  
getUserAction  
initImages
```

Se procede a dar una pequeña explicación del cometido de cada una de ellas. La función ‘drawVector’ es la encargada de pintar el buffer de datos introducido por el usuario o como argumento, de esta manera se permite llevar un control en todo momento de los datos con los que estamos trabajando haciendo que el proceso sea mas cómodo.

La función ‘getUserAction’ se encarga de controlar la entrada de datos por parte del usuario. Esta función se encuentra dentro del cuerpo de las funciones interactive y se encarga de guardar en un buffer la entrada de datos del usuario.

La función ‘initImages’ se encarga de hacer una llamada a la función load_image del paquete magick, el cual nos permite realizar plots de las formulas para ayudar al usuario en el proceso.

4.2 Desarrollo de ficheros C

Con el desarrollo de las funciones en R se cubre por completo la funcionalidad referente al paquete, sin embargo, se ha creído conveniente apoyar dicho desarrollo con un lenguaje de programación más implementado hoy en día que R, y ese es el lenguaje C. Se han creado una serie de funciones correspondientes al grupo de ‘funciones principales de calculo’ las cuales se encargan de obtener el resultado del método correspondiente, pero con el desarrollo en lenguaje C.

Para poder ejecutar estas funciones es necesario modificar el archivo NAMESPACE dentro de la estructura del paquete construido de la siguiente manera:

```
exportPattern(".*")  
useDynLib(LearningRlab)
```

Mediante el ‘exportPattern’ establecemos de que manera vamos a llamar a las funciones desarrolladas. En nuestro caso, hemos optado por utilizar el carácter ‘.’.

El problema de las funciones desarrolladas en C es que el formato del resultado de la ejecución no posee un diseño tan personalizable como el de R, de modo que como se ha comentado anteriormente, simplemente se ha utilizado el desarrollo en C para el cálculo de las funciones principales, pero no para aquellas que requieren de una trazabilidad más personalizada para ayudar con el aprendizaje.

4.3 Consideraciones y Pre-requisitos

El objetivo del paquete ‘LearningRlab’ es conseguir aplicar un modelo de enseñanza estadística básica interactiva, por lo que para su desarrollo se ha necesitado de funcionalidad auxiliar proporcionada por otros paquetes. El paquete debe servir como intermediario entre el usuario y los métodos de enseñanza que conlleva, y para ello se ha necesitado incrustar imágenes que ayuden en la asimilación del contenido explicado así como salida por pantalla de gráficas o colores que ayuden al usuario a comprender mejor el proceso llevado a cabo.

En primer lugar es necesario la instalación y posterior carga al entorno de trabajo del paquete ‘magick’. Este paquete es el encargado de manejar la carga de imágenes desde nuestro repositorio al framework de trabajo del usuario.

4.4 Paquete ‘Magick’

El paquete ‘magick’ es el resultado de un ambicioso esfuerzo llevado a cabo con el objetivo de simplificar la manera de procesamiento de imágenes de alta calidad dentro de R. La versión actual de ‘Magick’ contiene una gran variedad de funcionalidad que nos permite jugar con las imágenes. En lo referente al desarrollo del paquete LearningRlab, no entraremos en profundidad en el uso de esta librería, sino que simplemente la utilizaremos para cargar las imágenes al entorno de trabajo del usuario y poco más.

La principal razón por la que se ha optado por la utilización del paquete ‘magick’ es debido a que convierte y renderiza de manera automática todos los tipos de imágenes más comunes. El paquete soporta docenas de formatos y detecta el tipo de manera automática. Podemos ver el tipo de imágenes soportadas ejecutando el comando:

```
- magick::magick_config()
```

4.4.1 Instalación en Windows / OS-X

En los sistemas Windows o OS-X, la instalación del paquete se realiza de manera muy sencilla. La manera de hacerlo es acceder al repositorio de CRAN, ya sea mediante la interfaz gráfica de nuestro propio RStudio o introduciendo el comando:

```
- install.packages("magick")
```

Los paquetes binarios CRAN tienen habilitadas las funciones más importantes. Podemos utilizar ‘magick_config’ para ver qué características y formatos son compatibles con la versión de ImageMagick que tenemos instalada. Por último, para cargarlo a nuestro entorno de trabajo, debemos aplicar el siguiente comando:

```
- library(magick)
```

4.4.2 Instalación en Linux

En Linux, el proceso de instalación es diferente ya que es necesario instalarlo desde la fuente. Se necesita instalar la librería ‘ImageMagick++’ la cual en Debian/Ubuntu recibe el nombre de libmagick++-dev. El comando para su instalación sería:

```
- sudo apt-get install libmagick++-dev
```

Por otro lado, para su instalación sobre sistemas Fedora o CentOS/RHEL necesitamos ejecutar una pequeña modificación del comando anterior:

```
- sudo yum install ImageMagick-c++-devel
```

4.5 Paquete ‘Crayon’

El desarrollo de este paquete se ha inspirado en el proyecto ‘chalk JavaScript’. El objetivo es mostrar la salida de la terminal con colores que soportan la codificación ‘ANSI’. Es un paquete muy útil a la hora de ayudar al usuario con la visualización de los datos ya que resulta más fácil identificar los diferentes resultados codificados por colores.

Para instalar este paquete, basta con utilizar la interfaz gráfica ofrecida por RStudio y realizar la instalación desde CRAN, o bien ejecutar el siguiente comando en la consola de nuestro framework:

```
- install.packages("crayon")
```

5 CONCLUSIONES

Tras la realización del proyecto, se han consolidado una serie de conocimientos y de bases tanto teóricas como prácticas que han permitido al alumno perfeccionar sus habilidades y las competencias adquiridas durante la realización del

grado.

Las principales capacidades que se han visto respaldadas han sido las siguientes:

- Trabajo en equipo y comunicación
- Planificación de tareas y organización
- Desarrollo en lenguaje R
- Desarrollo de documentación y control de versiones

Tras la planificación y posterior desarrollo del proyecto propuesto el alumno ha sido capaz de plantear los requisitos necesarios para el desarrollo y posterior subida de un paquete al repositorio oficial de la comunidad de R, CRAN.

Antes del desarrollo del paquete, no se ha encontrado en el repositorio algún otro que cubra las necesidades, o este enfocado desde el punto de vista que hemos planteado. Debido a esto, se considera que se ha aportado a CRAN un paquete diferente, desde el punto de vista de la enseñanza estadística básica, lo cual pretende fomentar y diversificar los métodos de aprendizaje tradicionales.

Dado el estado del arte actual con respecto al repositorio oficial, se ha creído que se aporta una metodología diferente con un enfoque didáctico puesto que tras la revisión del contenido del repositorio CRAN, no se han encontrado paquetes que estén enfocados a usuarios de niveles básicos, sino que se suele utilizar en proyectos de investigación y cursos superiores.

Se considera que el alumno ha puesto en práctica diversas metodologías y conocimientos adquiridos durante que el curso los cuales le han permitido realizar un desarrollo profesional de un paquete el cual cumple con los requerimientos necesarios para su subida al repositorio oficial.

6 MANUAL DE USUARIO

Documentación Rd

MANUAL DE USUARIO

Cómo usar el paquete

Package ‘LearningRlab’

September 26, 2018

Version 1.2

Date 2018-09-23

Title Statistical Learning Functions

Author Dennis Monheimius [aut],
Eduardo Benito [aut, cre],
Juan Jose Cuadrado [ctb],
Universidad de Alcala de Henares [ctb]

Maintainer Eduardo Benito <eduardo.benito@edu.uah.es>

Depends magick, crayon

Suggests knitr

Description The LearningRlab package pretends to serve the user as a method of learning basic statistical functions at secondary and baccalaureate courses. The content of the package incorporates a serie of statistical functions like the calculus of the arithmetic mean or the frequencies. There is no only calculus functions, further more, there are incorporated interactive and explicative functions in order to help and guide the user in the learning process.

License Unlimited

VignetteBuilder knitr

NeedsCompilation yes

Type Package

R topics documented:

averageDeviation_	2
drawVector	3
explain.absolute_acum_frecuency	4
explain.absolute_frecuency	5
explain.averageDeviation	6
explain.geometricMean	7
explain.mean	8
explain.median	9
explain.mode	10
explain.percentile	11
explain.quartile	12
explain.relative_acum_frecuency	13
explain.relative_frecuency	14
explain.standardDeviation	15

explain.variance	16
frecuency_abs	17
frecuency_absolute_acum	18
frecuency_relative	19
frecuency_relative_acum	20
geometricMean_	21
getUserAction	22
initImages	23
interactive.absolute_acum_frecuency	23
interactive.absolute_frecuency	24
interactive.average_deviation	25
interactive.geometricMean	26
interactive.mean	26
interactive.median	27
interactive.mode	28
interactive.percentile	29
interactive.quartile	29
interactive.relative_acum_frecuency	30
interactive.relative_frecuency	31
interactive.standard_deviation	32
interactive.variance	32
LearningRlab	33
meanC	34
mean_	35
median_	36
mode_	37
percentile_	38
quartile_	39
standardDeviation_	40
variance_	41
Index	43

averageDeviation_	<i>Average Absolute Deviation Function</i>
-----------------------------	--

Description

This function calculates the average absolute deviation of a numbers vector.

Usage

```
averageDeviation_(x)
```

Arguments

x	Should be a numbers vector
---	----------------------------

Details

To calculate the average deviation, the user should give a numbers vector. The result is the sum of the differences in absolute value between each vector element and the mean, divided by the number of elemets. The average absolute deviation formule is the following:

Value

Numeric, the average absolute deviation of the numbers vector.

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"averageDeviation_(1:20)
```

```
mean = 10.5
```

```
ad = (|1-10.5| + |2-10.5|...+|20-10.5|) / 20
```

```
This example returns this result -> 5
```

```
standardDeviation_(c(9,2,1,8,11))
```

```
mean = 6.2
```

```
ad = (|9-6.2| + |2-6.2| + |1-6.2| + |8-6.2| + |11-6.2|) / 5)
```

```
This example returns this result -> 3.76"
```

drawVector	<i>Draw Vector Function</i>
------------	-----------------------------

Description

This function prints all the elements of a vector

Usage

```
drawVector(buffer)
```

Arguments

buffer	A vector of elements
--------	----------------------

Value

There isn't return value, prints on screen

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5 or `c(true,false,false)` creates a vector with the booleans: true, false, true

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
{  
  "drawVector(1:10)  
  
  This example prints: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10"  
}
```

explain.absolute_acum_frecuency

Absolute Accumulated Frecuency Calculus Explained

Description

Step by step demonstration of the absolute accumulated frecuency calculus

Usage

`explain.absolute_acum_frecuency(v,x)`

Arguments

v	Should be a vector
x	Should be a number

Details

To calculate the absolute accumulated frecuency, the user should give a vector and a number. We can saw the absolute accumulated frecuency formule in the `frecuency_acum_absolute` help document.

Value

A demonstration of the calculus process

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
{  
  data <- c(1,2,2,5,10,4,2)  
  value = 2  
  explain.absolute_acum_frecuency(data, value)  
}
```

explain.absolute_frecuency

Absolute Frecuency Calculus Explained

Description

Step by step demonstration of the absolute frecuency calculus

Usage

```
explain.absolute_frecuency(v,x)
```

Arguments

v	Should be a vector
x	Should be a number

Details

To calculate the absolute frecuency, the user should give a vector and a number. We can saw the absolute frecuency formule in the frecuency_abs help document.

Value

A demonstration of the calculus process

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
{  
  data <- c(1,2,2,5,10,4,2)  
  value = 2  
  explain.absolute_frecuency(data, value)  
}
```

explain.averageDeviation

Average Absolute Deviation Function Explained

Description

Step by step demonstration of the average absolute deviation calculus.

Usage

```
explain.averageDeviation(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the average absolute deviation, the user should give a numbers vector. The result is the explained process to calculate the average absolute deviation, with the data of the dataset provided like argument. We can saw the average absolute deviation formule in the averageDeviation_ help document.

Value

Numeric, the average absolute deviation of the numbers vector.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
data <- c(7,2,5,7,1,4,12)
explain.averageDeviation(data)

explain.averageDeviation(c(5,2,1,13,9,11,6))

"The process is explained in the function..."
```

explain.geometricMean *Geometric Mean Function Explained*

Description

Step by step demonstration of the geometric mean calculus.

Usage

```
explain.geometricMean(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the geometric mean of a dataset, the user should give a vector. The result is the explained process to calculate the geometric mean, with the data of the dataset provided like argument. We can saw the geometric mean formule in the `geometricMean_` help document.

Value

Numeric result and the process of this calculus explained.

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
{
  "data" <- c(5,21,12,7,3,9,1)
  explain.geometricMean(data)

  explain.geometricMean(1:20)

  The process is explained in the function..."
}
```

explain.mean	<i>Mean Function Explained</i>
--------------	--------------------------------

Description

Step by step demonstration of the arithmetic mean calculus.

Usage

```
explain.mean(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the arithmetic mean of a dataset, the user should give a vector. The result is the explained process to calculate the arithmetic mean, with the data of the dataset provided like argument. We can saw the arithmetic mean formule in the mean_ help document.

Value

Numeric result and the process of this calculus explained.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  data <- c(1,2,2,5,10,4,2)
  explain.mean(data)

  explain.mean(1:20)

  "The process is explained in the function..."
}
```

explain.median	<i>Median Function Explained</i>
----------------	----------------------------------

Description

Step by step demonstration of the median calculus.

Usage

```
explain.median(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the median, the user should give a numbers vector. The result is the explained process to calculate the median, with the data of the dataset provided like argument. We can saw the median formule in the median_ help document.

Value

Numeric result and the process of this calculus explained.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  data <- c(1,2,2,5,10,4,2)
  explain.median(data)

  explain.median(c(10,3,4,13,9,1))

  "The process is explained in the function..."
}
```

explain.mode	<i>Mode Function Explained</i>
--------------	--------------------------------

Description

Step by step demonstration of the mode calculus.

Usage

```
explain.mode(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the mode, the user should give a numbers vector. The result is the explained process to calculate the mode, with the data of the dataset provided like argument. We can saw the mode formule in the mode_ help document.

Value

Numeric result and the process of this calculus explained.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  data <- c(1,1,2,5,2,3,1,4,1)
  explain.mode(data)

  explain.mode(c(1,2,5,5,7,10))

  "The process is explained in the function..."
}
```

explain.percentile	<i>Percentiles Calculus Explained</i>
--------------------	---------------------------------------

Description

Step by step demonstration of the percentiles calculus

Usage

```
explain.percentile(x)
```

Arguments

x Should be a vector

Details

To calculate the percentiles, the user should give a vector. We can saw the percentile formule in the percentile_ help document.

Value

A demonstration of the calculus process

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "data <- c(1,2,2,5,10,4,2)"
  explain.percentile(data)"
}
```

`explain.quartile`*Quartiles Calculus Explained*

Description

Step by step demonstration of the quartiles calculus

Usage

```
explain.quartile(x)
```

Arguments

`x` Should be a vector

Details

To calculate the quartiles, the user should give a vector. We can saw the quartile formule in the `quartile_` help document.

Value

A demonstration of the calculus process

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjc@uah.es>
Universidad de Alcala de Henares

Examples

```
{  
  " data <- c(1,2,2,5,10,4,2)  
  explain.quartile(data)"  
}
```

`explain.relative_acum_frecuency`*Relative Accumulated Frequency Calculus Explained*

Description

Step by step demonstration of the relative accumulated frequency calculus

Usage

```
explain.relative_acum_frecuency(v,x)
```

Arguments

<code>v</code>	Should be a vector
<code>x</code>	Should be a numebr of the vector

Details

To calculate the relative accumulated frequency, the user should give a vector and a number. We can saw the relative accumulated frequency formule in the `frecuency_acum_relative` help document.

Value

A demonstration of the calculus process

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjc@uah.es>
Universidad de Alcala de Henares

Examples

```
{  
  " data <- c(1,2,2,5,10,4,2)  
  value = 2  
  explain.relative_acum_frecuency(data, value)"  
}
```

`explain.relative_frequency`*Relative Frequency Calculus Explained*

Description

Step by step demonstration of the relative frequency calculus

Usage

```
explain.relative_frequency(v,x)
```

Arguments

<code>v</code>	Should be a vector
<code>x</code>	Should be a number

Details

To calculate the relative frequency, the user should give a vector and a number. We can see the relative frequency formula in the `frequency_relative` help document.

Value

A demonstration of the calculus process

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monheimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
{  
  data <- c(1,2,2,5,10,4,2)  
  value = 2  
  explain.relative_frequency(data, value)  
}
```

`explain.standardDeviation`*Standard Deviation Function Explained*

Description

Step by step demonstration of the standard deviation calculus.

Usage

```
explain.standardDeviation(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the standard deviation, the user should give a numbers vector. The result is the explained process to calculate the standard deviation, with the data of the dataset provided like argument. We can saw the standard deviation formule in the standardDeviation_ help document.

Value

Numeric result and the process of this calculus explained.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>

Eduardo Benito, <eduardo.benito@edu.uah.es>

Juan Jose Cuadrado, <jjcg@uah.es>

Universidad de Alcala de Henares

Examples

```
data <- c(1,5,3,7,10,4,2)
explain.standardDeviation(data)

explain.standardDeviation(c(15,12,4,13,9,1,6))

"The process is explained in the function..."
```

explain.variance	<i>Variance Function Explained</i>
------------------	------------------------------------

Description

Step by step demonstration of the variance calculus.

Usage

```
explain.variance(x)
```

Arguments

x	Should be a numbers vector
---	----------------------------

Details

To calculate the variance, the user should give a numbers vector. The result is the explained process to calculate the variance, with the data of the dataset provided like argument. We can saw the variance formule in the variance_ help document.

Value

Numeric result and the process of this calculus explained.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
data <- c(10,4,5,7,3,4,1)
explain.averageDeviation(data)

explain.averageDeviation(c(20,12,9,13,9,1,6))

"The process is explained in the function..."
```

frecuency_abs	<i>Absolute Frecuency Calculus</i>
---------------	------------------------------------

Description

This function calculate the number of times that a specific number appears in the data set.

Usage

```
frecuency_abs(v,x)
```

Arguments

v	Should be a vector
x	Should be a number

Details

The absolute frecuency formula is the following:

Absolute frequency = number of aparitions of
the examined element

LearningRlab ®

Value

An integer that represents the number of times that the value appears in the vector

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "frecuency_abs(c(1,4,3,3,2,5,7,12,1,2,3,12), 12)

  This will create a vector with the elements especificed
  and count the number of times that the element 12 appears."}
```

frequency_absolute_acum

Accumulated Absolute Frequency Calculus

Description

This function calculate the number of times that a specific number appears in the data set. The value depends on the elements that are lower than itself

Usage

```
frequency_absolute_acum(v,x)
```

Arguments

v	Should be a vector
x	Should be a number

Details

The accumulated absolute frequency formula is the following:

$$\text{Absolute accumulated frequency (X)} = \sum F_i \text{ where } i \leq X$$

LearningRlab ®

Value

A double that represents the number of times that the value appears in the vector regarding the total of elements

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "frequency_absolute_acum(c(1,4,3,3,2,5,7,12,1,2,3,12), 12)

  This will create a vector with the elements especificied
  and count the number of times that the element 12 appears
  regarding the total of elements."}
```

frequency_relative	<i>Relative Frequency Calculus</i>
--------------------	------------------------------------

Description

This function calculate the number of times that a specific number appears in the data set divided by the total length of the vector.

Usage

```
frequency_relative(v,x)
```

Arguments

v	Should be a vector
x	Should be a number

Details

The relative frequency formula is the following:

$$\text{Relative frequency} = \frac{\text{absolute frequency}}{\sum \text{all frequencies}}$$

LearningRlab ®

Value

A double that represents the number of times that the value appears in the vector regarding the total of elements

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "frequency_relative(c(1,4,3,3,2,5,7,12,1,2,3,12), 12)
```

This will create a vector with the elements especificied
 and count the number of times that the element 12 appears
 regarding the total of elements."}

frequency_relative_acum

Accumulated Relative Frequency Calculus

Description

This function calculate the number of times that a specific number appears in the data set divided by the total length of the vector. The value depends on the elements that are lower than itself

Usage

```
frequency_relative_acum(v,x)
```

Arguments

v	Should be a vector
x	Should be a number

Details

The accumulated relative frequency formula is the following:

$$\text{Relative accumulated frequency (X)} = \sum f_i \text{ where } i \leq X$$

LearningRlab ®

Value

A double that represents the number of times that the value appears in the vector regarding the total of elements

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "frequency_relative_acum(c(1,4,3,3,2,5,7,12,1,2,3,12), 12)

  This will create a vector with the elements especificed
  and count the number of times that the element 12 appears
  regarding the total of elements."}
```

 geometricMean_

Geometric Mean Function

Description

This function calculates the geometric mean of a numbers vector.

Usage

```
geometricMean_(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the geometric mean of a dataset, the user should give a numbers vector. The result is the product of all vector elements raise to 1 divided by the number of elements. The arithmetic mean formule is the following:

Value

A numeric, the geometric mean of the numbers vector.

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
"geometricMean_(1:20)
```

```
geometricMean = (1*2*3*4*5*6*7*8*9*10*11*12*13*14*15*16*17*18*19*20)^(1/20)
```

This example returns this result -> 8.304

```
geometricMean_(c(2,4,6,8,10,12,14,16,18))
```

```
geometricMean = (2*4*6*8*10*12*14*16*18)^(1/9)
```

This example returns this result -> 8.294"

getUserAction

Get User Action Funcion

Description

This function get the buffer introduced by the user. Typically a numerical vector.

Usage

```
getUserAction()
```

Arguments

void, not argument

Value

A vector

Note

The process is interactive with the user

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>

Eduardo Benito, <eduardo.benito@edu.uah.es>

Juan Jose Cuadrado, <jjcg@uah.es>

Universidad de Alcala de Henares

Examples

```
{
  "vector <- getUserAction()"
}
```

`initImages`*Init Images Function*

Description

This function is used to display an image.

Usage

```
initImages(path)
```

Arguments

path	An url of an image
------	--------------------

Value

There isn't return value

Note

The path should be toward an image

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
{  
  "initImages(https://i.imgur.com/8237YhzJ.png)  
  
  The path in double quotes."  
}
```

`interactive.absolute_acum_frecuency`*User Interactive Absolute Accumulated Frequency Calculus*

Description

Interactive function for absolute accumulated frequency calculus.

Usage

```
interactive.absolute_acum_frecuency()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the formula.

Value

An interactive process to calculate the absolute accumulated frequency

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjc@uah.es>
 Universidad de Alcala de Henares

Examples

```
"interactive.absolute_acum_frecuency()"
```

```
interactive.absolute_frecuency
```

User Interactive Absolute Frequency Calculus

Description

Interactive function for absolute frequency calculus.

Usage

```
interactive.absolute_frecuency()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the formula.

Value

An interactive process to calculate the absolute frequency

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjc@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.absolute_frecuency()"
```

```
interactive.average_deviation
```

User Interactive Average Absolute Deviation Calculus

Description

Interactive function for average absolute deviation calculus.

Usage

```
interactive.average_deviation()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the average absolute deviation formule.

Value

An interactive process to calculate the average absolute deviation

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjc@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.average_deviation()"
```

interactive.geometricMean

User Interactive Geometric Mean Calculus

Description

Interactive function for geometric mean calculus.

Usage

```
interactive.geometricMean()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the geometric mean formule, apart from the geometricMean_ help document.

Value

An interactive process to calculate the geometric mean.

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.geometricMean()"
```

interactive.mean

User Interactive Mean Calculus

Description

Interactive function for arithmetic mean calculus.

Usage

```
interactive.mean()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the arithmetic mean formule, apart from the mean_ help document.

Value

An interactive process to calculate the arithmetic mean.

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "interactive.mean()"
}
```

interactive.median	<i>User Interactive Median Calculus</i>
--------------------	---

Description

Interactive function for median calculus.

Usage

```
interactive.median()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the median formule, apart from the median_ help document.

Value

An interactive process to calculate the median

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.median()"
```

interactive.mode	<i>User Interactive Mode Calculus</i>
------------------	---------------------------------------

Description

Interactive function for mode calculus.

Usage

```
interactive.mode()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset.

Value

An interactive process to calculate the mode.

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.mode()"
```

`interactive.percentile`*User Interactive Percentile Calculus*

Description

Interactive function for percentiles calculus.

Usage

```
interactive.percentile()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the formula.

Value

An interactive process to calculate the percentiles

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.percentile()"
```

`interactive.quartile`*User Interactive Quartiles Calculus*

Description

Interactive function for quartiles calculus.

Usage

```
interactive.quartile()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the formula.

Value

An interactive process to calculate the quartiles

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.quartile()"
```

```
interactive.relative_acum_frecuency
```

User Interactive Relative Accumulated Frequency Calculus

Description

Interactive function for relative accumulated frequency calculus.

Usage

```
interactive.relative_acum_frecuency()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the formula.

Value

An interactive process to calculate the relative accumulated frequency

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.relative_acum_frecuency()"
```

```
interactive.relative_frecuency
```

User Interactive Relative Frecuency Calculus

Description

Interactive function for relative frecuency calculus.

Usage

```
interactive.relative_frecuency()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the formula.

Value

An interactive process to calculate the relative frecuency

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.relative_frecuency()"
```

`interactive.standard_deviation`*User Interactive Standard Deviation Calculus*

Description

Interactive function for standard deviation calculus.

Usage

```
interactive.standard_deviation()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the standard deviation formule, apart from the `standardDeviation_` help document.

Value

An interactive process to calculate the standard deviation

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjc@uah.es>
Universidad de Alcala de Henares

Examples

```
"interactive.standard_deviation()"
```

`interactive.variance` *User Interactive Variance Calculus*

Description

Interactive function for variance calculus.

Usage

```
interactive.variance()
```

Arguments

void, not argument

Details

The user provides the dataset when the function needs it. After that, the function will ask what is the correct result for this dataset. The function itself will provide the variance formule, apart from the variance_ help document.

Value

An interactive process to calculate the average absolute deviation

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
"interactive.variance()"
```

 LearningRlab

Statistical Learning Functions

Description

Package used to teach basic statistics to students.

Details

The LearningRlab package pretends to serve the user as a method of learning basic statistical functions at secondary and baccalaureate courses. The content of the package incorporate a serie of statistical functions like the calculus of the arithmetic mean or the calculus of the frequencies. There is no only calculus functions, further more, there are incorporated interactive and explicative functions to help and guide the user in the learning process.

Package: LearningRlab
 Type: Package
 Version: 1.2
 Date: 2018-09-19
 License: Unlimited

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>

Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares
 Maintainer: Eduardo Benito <eduardo.benito@edu.uah.com>

 meanC

Mean Function Developed in C

Description

This function calculates the arithmetic mean of a numbers vector.

Usage

meanC(x)

Arguments

x Should be a numbers vector

Details

To calculate the arithmetic mean of a dataset, the user should give a numbers vector. The result is the addition of all vector elements divided by the number of elements. The arithmetic mean formule is the following:

$$\bar{X} = \frac{\sum_{i=1}^n X_i}{n} = \frac{X_1 + X_2 + \dots + X_n}{n}$$

@LearningRlab

Value

A numeric, the arithmetic mean of the numbers vector.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
"meanC(1:20)
```

```
mean = (1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20)/20
```

This example returns this result -> 10,5

```
meanC(c(2,4,6,8,10,12,14,16,18))
```

```
mean = (2+4+6+8+10+12+14+16+18)/9
```

This example returns this result -> 10"

mean_	<i>Mean Calculus Function</i>
-------	-------------------------------

Description

This function calculates the arithmetic mean of a numbers vector.

Usage

```
mean_(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the arithmetic mean of a dataset, the user should give a numbers vector. The result is the addition of all vector elements divided by the number of elements. The arithmetic mean formule is the following:

$$\bar{X} = \frac{\sum_{i=1}^n x_i}{n} = \frac{x_1 + x_2 + \dots + x_n}{n}$$

@LearningRlab

Value

A numeric, the arithmetic mean of the numbers vector.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjc@uah.es>
 Universidad de Alcala de Henares

Examples

```
"mean_(1:20)
```

```
mean = (1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20)/20
```

This example returns this result -> 10,5

```
mean_(c(2,4,6,8,10,12,14,16,18))
```

```
mean = (2+4+6+8+10+12+14+16+18)/9
```

This example returns this result -> 10"

median_	<i>Median Calculus Function</i>
---------	---------------------------------

Description

This function calculates the median of a numbers vector.

Usage

```
median_(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the median, the user should give a numbers vector. The result is the value separating the higher half from the lower half of the dataset, it may be thought of as the middle value. The median formule is the following:

$$\text{Median} = \frac{1}{2}(n + 1)\text{th value}$$

® LearningRlab

Value

A numeric, the median of the numbers vector.

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "median_(c(1,3,2,5,12,4,4,2,9)) -> median_(c(1,2,2,3,4,4,5,9,12))

  median = 5 th value = 4

  median_(c(1,20,3,17,8,4,21,3)) -> median_(c(1,3,3,7,8,17,20,21))

  median = 4.5 = 4 th value = 7"
}
```

mode_

*Mode Calculus Function***Description**

This function calculates the mode of a numbers vector.

Usage

```
mode_(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the mode of a dataset, the user should give a numbers vector. The result is the numeric value that appears most often. In other words, it's the value that is most likely to be sampled. The mode formule is the following:

Value

Numeric, the mode of the numbers vector.

Note

A vector is created by `c()`, like `c(1,2,3,4,5)` creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "mode_(c(1,2,2,3,4))

    mode = 2 (appears two times)

  mode_(c(1,1,2,5,2,3,1,4,1))

    mode = 1 (appears four times)"
}
```

percentile_

*Percentile Calculus Function***Description**

This function calculate the percentiles of a vector of numbers

Usage

```
percentile_(x)
```

Arguments

x Should be a vector

Details

To calculate the percentiles, the user should give a vector. This function divide the dataset in 100 parts as equal as possible. The formula is the following:

Value

A vector sorted with the elements divided by 100 parts

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  " percentile_(c(1,4,3,3,2,5,7,12,1,2,3,12))
    This will create a vector with the elements especificied"
}
```

quartile_

*Quartiles Calculus***Description**

Calculates the 3 Quartiles of a vector of data

Usage

```
quartile_(x)
```

Arguments

x Should be a vector

Details

To calculate the quartiles, the user should give a vector. This function divide the dataset in 4 parts as equal as possible. The formula is the following:

Value

A vector sorted with the elements divided by 4 parts

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
{
  "quartile_(1:20)
    This will create a vector with the elements (1,2,3...,20)
    The elements will be separated in 4 parts as equals as possible"
}
```

standardDeviation_	<i>Standard Deviation Calculus Function</i>
--------------------	---

Description

This function calculates the standard deviation of a numbers vector.

Usage

```
standardDeviation_(x)
```

Arguments

x	Should be a numbers vector
---	----------------------------

Details

To calculate the standard deviation, the user should give a numbers vector. The result is the square root of the sum of the differences between each vector element and the mean squared divided by the number of elemets. The standard deviation formule is the following:

Value

Numeric, the standard deviation of the numbers vector.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
 Eduardo Benito, <eduardo.benito@edu.uah.es>
 Juan Jose Cuadrado, <jjcg@uah.es>
 Universidad de Alcala de Henares

Examples

```
"standardDeviation_(1:20)

mean = 10.5

sd = sqrt( ( (1-10.5)^2 + (2-10.5)^2....+(20-10.5)^2) / 20)

This example returns this result -> 5.766

standardDeviation_(c(1,2,5,8,10))

mean = 5.2

sd = sqrt( ( (1-5.2)^2 + (2-5.2)^2 + (5-5.2)^2 + (8-5.2)^2 + (10-5.2)^2) / 5)
```

This example returns this result -> 3.429"

variance_

Variance Calculus Function

Description

This function calculates the variance of a numbers vector.

Usage

```
variance_(x)
```

Arguments

x Should be a numbers vector

Details

To calculate the variance, the user should give a numbers vector. The result is the expectation of the squared deviation of all numbers vector from its mean. The variance formule is the following:

Value

Numeric, the variance of the numbers vector.

Note

A vector is created by c(), like c(1,2,3,4,5) creates a vector with the numbers: 1,2,3,4,5

Author(s)

Dennis Monheimius, <dennis.monhemimius@edu.uah.es>
Eduardo Benito, <eduardo.benito@edu.uah.es>
Juan Jose Cuadrado, <jjcg@uah.es>
Universidad de Alcala de Henares

Examples

```
"variance_(1:20)

mean = 10.5

var = ( (1-10.5)^2 + (2-10.5)^2...+ (20-10.5)^2 ) / 20

This example returns this result -> 33.25

variance_(c(5,2,3,8,7))

mean = 5
```

```
var = ( (5-5)^2 + (2-5)^2 + (3-5)^2 + (8-5)^2 + (7-5)^2 ) / 5
```

This example returns this result -> 5.2"

Index

- *Topic **\textasciitildeabsoluta**
 - averageDeviation_, 2
 - explain.averageDeviation, 6
 - frequency_abs, 17
 - frequency_absolute_acum, 18
 - interactive.average_deviation, 25
- *Topic **\textasciitildeabsolute**
 - averageDeviation_, 2
 - explain.absolute_acum_frequency, 4
 - explain.absolute_frequency, 5
 - explain.averageDeviation, 6
 - frequency_abs, 17
 - frequency_absolute_acum, 18
 - interactive.absolute_acum_frequency, 23
 - interactive.absolute_frequency, 24
 - interactive.average_deviation, 25
- *Topic **\textasciitildeaccumulated**
 - explain.absolute_acum_frequency, 4
 - explain.relative_acum_frequency, 13
 - frequency_absolute_acum, 18
 - frequency_relative_acum, 20
 - interactive.absolute_acum_frequency, 23
 - interactive.relative_acum_frequency, 30
- *Topic **\textasciitildeacumulada**
 - explain.absolute_acum_frequency, 4
 - explain.relative_acum_frequency, 13
 - frequency_absolute_acum, 18
 - frequency_relative_acum, 20
- *Topic **\textasciitildearithmetic.mean**
 - mean_, 35
 - meanC, 34
- *Topic **\textasciitildeaverage**
 - averageDeviation_, 2
 - explain.averageDeviation, 6
 - interactive.average_deviation, 25
- *Topic **\textasciitildecogger**
 - getUserAction, 22
- *Topic **\textasciitildecuartiles**
 - interactive.quartile, 29
 - quartile_, 39
- *Topic **\textasciitildedesviacion**
 - averageDeviation_, 2
 - explain.averageDeviation, 6
 - explain.standardDeviation, 15
 - interactive.average_deviation, 25
 - interactive.standard_deviation, 32
 - standardDeviation_, 40
- *Topic **\textasciitildedevelopment**
 - averageDeviation_, 2
 - explain.averageDeviation, 6
 - explain.standardDeviation, 15
 - interactive.average_deviation, 25
 - interactive.standard_deviation, 32
 - standardDeviation_, 40
- *Topic **\textasciitildedraw**
 - drawVector, 3
- *Topic **\textasciitildeexplained**
 - explain.averageDeviation, 6
 - explain.standardDeviation, 15
- *Topic **\textasciitildeexplain**
 - explain.absolute_acum_frequency, 4
 - explain.absolute_frequency, 5
 - explain.geometricMean, 7
 - explain.mean, 8
 - explain.median, 9
 - explain.mode, 10
 - explain.percentile, 11
 - explain.quartile, 12
 - explain.relative_acum_frequency, 13
 - explain.relative_frequency, 14
 - explain.variance, 16
- *Topic **\textasciitildeexplicada**
 - explain.averageDeviation, 6
 - explain.geometricMean, 7
 - explain.mean, 8
 - explain.median, 9
 - explain.mode, 10
 - explain.standardDeviation, 15
 - explain.variance, 16
- *Topic **\textasciitildedefrecuencia**

7 BIBLIOGRAFÍA

- [1] R Core Team. (2018, Jul 02). Writing R Extensions (versión 3.5.1) [Online]. Available:
<https://cran.r-project.org/doc/manuals/r-release/R-exts.pdf>
- [2] The R Project for Statistical Computing [Online]. Available:
<https://www.r-project.org/>
- [3] W. N. Venables, D. M. Smith, The R Core Team. (2018, Jul 02). An Introduction to R (versión 3.5.1) [Online]. Available:
<https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf>
- [4] F. Leisch. (2009, Sept 14). Creating R Packages: A Tutorial (1st ed.) [Online]. Available:
<https://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf>