UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa

Titulación:

Master en Ingeniería Industrial

Alumno:

Gerard Pueyo Vinader

Titulo del TFM:

Proyecto para la creación de un paquete en R para el análisis de datos basados en el paquete RQDA

Director/a del TFM:

Vicenç Fernandez Alarcon

Convocatoria de entrega del TFM:

Septiembre de 2016

Contenidos de este volumen:

DOCUMENTO 1.- MEMORIA

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

La introducción está formada por los siguientes apartados.

1.1 Objeto

El objeto de este proyecto es el desarrollo de un paquete de R basado en la creación de una interfaz gráfica de usuario para el análisis de datos tratados con el paquete RQDA.

1.2 Alcance

El proyecto tiene previsto abarcar las siguientes tareas:

- Identificar los análisis que debe realizar la interfaz.
- Diseñar y programar las funciones.
- Diseñar e implementar la interfaz gráfica de usuario.
- Escribir la documentación y el manual del paquete.
- Crear el paquete de R bajo los estándares de CRAN.

Queda fuera del alcance enviar el paquete a la base de datos de CRAN para conocer si el paquete queda finalmente alojado allí.

1.3 Especificaciones básicas

Para la consecución del objeto, el proyecto se debe realizar en base a las siguientes especificaciones técnicas:

- Realizar la interfaz con el lenguaje de programación R.
- Basar la interfaz en uno de los siguientes paquetes: *gWidgetsRGtk2*, *gWidgets* o *RGtk2*, con el objetivo de tener un diseño parecido a RQDA.
- Crear una interfaz con un mínimo de 4 tipos de análisis diferentes, que aparte, se incluyan opciones gráficas con *igraph* y tablas de concurrencias entre códigos.
- Realizar la documentación del paquete con *LaTeX*.
- Licenciar el software para que todo el mundo pueda estudiar, modificar y mejorar su diseño (software libre).
- Cumplir con los estándares de formalización y con las políticas que exige CRAN para poder enviar un paquete a su repositorio.
- Realizar la interfaz para que funcione con la versión de R 3.2.2 en adelante y para cualquier sistema operativo (*Linux*, *Windows* y *Mac OS*).

2. Justificación

En el mundo de la investigación son muchos los investigadores que dan preferencia al análisis cualitativo de datos (QDA) versus al análisis cuantitativo de datos.

Este tipo de análisis consta de 3 partes. Una primera que consiste en el tratamiento de la información que se dispone, normalmente transcripciones de entrevistas o conversaciones. En ésta, lo que se intenta es reducir dicha información en unidades significativas para que sean más fáciles de procesar. A cada de las unidades de información se le asigna un código para identificarlas y tenerlas clasificadas por temáticas y categorías. Una segunda parte que consiste en el tratamiento de la información que se ha reducido para su análisis detallado y para hacer lo que sería desarrollo del trabajo. Se busca presentar los datos en forma de tablas o gráficos para que, en la tercera parte, se extraigan las conclusiones de la investigación.

Tiempo atrás este análisis se hacía a mano. Se transcribían en papel las entrevistas que se habían hecho y con subrayadores de colores se hacía la codificación de los fragmentos de texto para que después, a mano también, se llevarán a cabo los conteos, las concurrencias de códigos, etc. No es de extrañar que esto ha evolucionado con el paso del tiempo. Hoy en día, para hacer este tipo de análisis existen programas informáticos como *MaxQDA*, *Atlas.ti* o *Nvivo9* que permiten sobretodo ahorrar tiempo. Todos ellos disponen de muchas opciones de análisis, pero no son gratuitos, requieren de la obtención de licencias de uso que pueden rondar los mil euros anuales.

Como alternativa a los investigadores que no desean o no puedan recurrir a éstos programas de pago existe un paquete de funciones de software libre bastante conocido llamado RQDA creado por Rounggui Huang (2014) [1] que usa el lenguaje de programación R. Dicho paquete ofrece una interfaz gráfica de usuario (GUI) para realizar la primera parte del análisis: codificación de las unidades de datos y su clasificación y categorización.

Para la segunda, que vendría a ser la del tratamiento de las unidades de datos para la extracción de conclusiones, RQDA ofrecería una serie de funciones implementadas con el mismo R. Las funciones pueden servir para para realizar el trabajo satisfactoriamente, pero, así como para la primera parte se dispone de una interfaz muy clara, de fácil manejo y muy bien conseguida, para la segunda el trabajo es más farragoso.

Primeramente, porque se requiere tener nociones básicas de R. Muchos investigadores no tienen conocimientos de programación de ningún tipo, por lo que está última parte les supondría un esfuerzo adicional. Y segundo, porque la tarea de análisis se vuelve en un tema reiterativo de llamar a las funciones una y

otra vez para obtener los números de forma poco clara. Otras cosas más a destacar son los argumentos varían de una función a otra (en unos piden # de identificador y en otros el nombre), se buscan las concurrencias de una forma más lenta, no se pueden hacer gráficos, no se pueden hacer tablas, etc. Sí, ahorran tiempo respecto a hacerlo manualmente, pero los investigadores coinciden en que estaría bien disponer de algo más cómodo.

Son estas razones por las que en este proyecto se pretende diseñar un paquete de R que se utilice como complemento de RQDA de Rounggui Huang (2014) [1] y que añada todo aquello de lo que carece. Se buscará que el paquete sea igual de efectivo, por lo que se implementará una interfaz gráfica de las mismas características, para que los investigadores que utilicen RQDA por su interfaz lo tenga igual de fácil.

No se va a buscar que la interfaz tenga todas las funcionalidades que pueda haber en uno de pago ni todas aquellas existentes en el QDA, si no que se buscará aquellas funcionalidades que, como dice Pareto, pueda solucionar el 80% de las tareas más comunes para un investigador con un 20% de esfuerzo. En caso, de querer alguna tarea adicional se deberá complementar con la programación.

Destacar que la intención del proyecto es que el paquete resultado quede colgado en un repositorio libre llamada CRAN para que los investigadores que quieran pueda beneficiarse de él. Sin embargo, tal y como se indica en el alcance, queda fuera el subirlo dado que requiere que un grupo de voluntarios te lo acepte o te diga que antes de colgarlo se modifiquen ciertas cosas, y no se puede contemplar la demora que esto ocasiona (contando además que tendrán otros paquetes en cola para inspeccionar). Por ello, el paquete se considerará acabado cuando siga los estándares de formalización y las políticas CRAN. Después se mirará de subir, pero fuera de este proyecto.

3. Antecedentes

3.1 QDA

El *Qualitative Data Analysis* (Análisis Cualitativo de Datos), conocido mayormente por sus siglas QDA, es un método de análisis de base lingüística que se emplea en investigación y estudios científicos donde se busca evaluar la información que se dispone de forma cualitativa.

Mientras la investigación cuantitativa asigna valores numéricos a las declaraciones u observaciones, con el propósito de estudiar con métodos estadísticos posibles relaciones entre las variables, la investigación cualitativa recoge los discursos completos de los sujetos, para proceder luego a su interpretación, analizando las relaciones de significado que se producen en una determinada cultura o ideología.

Éste tipo de análisis se emplea principalmente por cualquier persona que necesite codificar texto o imágenes, anotar, buscar, explorar y extraer información de pequeñas o grandes colecciones de documentos e imágenes. Sus principales usuarios serían los investigadores en sociológica, ciencia y psicológica, sin embargo, también lo pueden utilizar personas como:

- Politólogos y etnógrafos.
- Investigadores de mercado, encuestadores y analistas de negocios.
- Analistas del crimen, expertos antifraudes, abogados y expertos de asistencia legal.
- Periodistas e historiadores.
- Especialistas en gestión de documentos y bibliotecarios.

La característica principal de los datos cualitativos es que están en forma de palabras o texto (incluso imágenes) por lo que no pueden ser representados como un número y, por lo tanto, estos datos son difícilmente medibles, no traducibles a términos matemáticos y no sujetos a la inferencia estadística.

Se suele considerar que los datos cualitativos son todos aquellos que se extraen de fuentes distintas a la encuesta y al experimento, como por ejemplo entrevistas abiertas, grupos de discusión o técnicas de observación y observación participante. A modo de resumen estas serían las todas las fuentes:

- Resultados de entrevistas
- Comentarios de reuniones
- Contacto interpersonal
- Documentos escritos
- Conductas o sucesos recogidos en notas de campo
- Fotografías

- Filmaciones
- Grabaciones sobre contextos investigados

El proceso general de un análisis cualitativo consta de 3 tareas principales: reducción de datos; disposición y transformación de datos; y obtención de resultados y verificación de conclusiones.

La reducción de datos responde a la necesidad de procesar grandes cantidades de información reduciéndolas a unidades elementales, fácilmente analizables, comprensivas, relevantes y significativamente densas. En esta fase se engloba 3 actividades principales:

- Separación de unidades. Consiste en dividir la información en unidades relevantes y significativas en base a una serie de criterios físicos (espaciales y temporales), temáticos, gramaticales, conversacionales o sociales.
- Identificación y clasificación de elementos. Consiste en examinar las unidades de datos para encontrar los componentes que nos permitan clasificar dichas unidades en categorías de contenido. Las actividades de categorización y codificación son las más significativas. La categorización hace referencia a agrupar conceptualmente las unidades que son cubiertas bajo un mismo tópico, este proceso se realiza conjuntamente a la división de unidades cuando se hace atendiendo criterios temáticos. La codificación hace referencia a asignar a cada unidad un indicativo o código propio de la categoría en la que se incluye, siendo el proceso físico, manipulativo, mediante el cual se realiza la categorización.
  Los sistemas de categorías pueden elaborarse inductivamente a partir de los propios datos, es decir, al examinar los datos se identifica el tópico capaz de cubrir cada unidad generando así una categoría.
- Síntesis y agrupamiento. Cuando categorizamos, estamos sintetizando diferentes unidades de datos en un mismo tópico o concepto.

La segunda tarea es la de disposición y transformación de los datos. Este paso permite presentar los datos de manera abarcable y operativa de cara a resolver las cuestiones de investigación. Engloba 2 actividades:

- Disposición de datos. Supone organizar los datos, presentándolos en alguna forma espacial ordenada, de tal modo que se simplifique la información y se posibilite su procesamiento posterior. Como por ejemplo tablas de frecuencias, concurrencias, porcentajes, etc.
- Transformación de datos. Supone pasar de una expresión verbal de los datos a una expresión numérica o gráfica al contar códigos, palabras, segmentos de palabras, etc.

La tarea final es la de obtención de resultados y verificación de conclusiones.

Consta de 3 procesos:

- Proceso para obtener conclusiones. Cuando se analizan los datos cualitativos, los resultados se obtienen en función del modo en que se dispone y organiza la información, a partir del recuento o de la concurrencia de códigos. Los resultados pueden surgir de la comparación con otros escenarios, casos, situaciones, etc similares al estudiado. También la comparación se puede realizar respecto a un criterio particular (una meta u objetico) o a una norma, que sirven de base para el establecimiento de un juicio de valor.
- Proceso para alcanzar conclusiones. La interpretación de los resultados nos lleva a conclusiones que pueden conducir a la creación y explicación de generalizaciones. Estas estrategias, que suponen integrar los resultados en marcos teóricos y de investigación más amplios son: consolidación teórica, aplicación de otras teorías, uso de metáforas y analogías y síntesis de los resultados con los obtenidos por otros investigadores.
- Verificación de conclusiones. Una vez alcanzadas las conclusiones de un estudio, es necesario verificarlas. Para favorecer o incrementar la obtención de conclusiones validas desde esta posición se hacen esfuerzos por conseguir que el proceso de investigación no se vea afectado por diferentes fuentes de error y que los resultados puedan ser generalizado. Para ello, debe ser comprobada la coherencia estructural para verifica que entre los datos e interpretaciones no se dan contradicciones o incoherencias mediante la búsqueda de datos o evidencias que se opongan con las conclusiones.

Figura 1. Subrayadores de colores [10]

Anteriormente, este procedimiento se realizaba completamente a mano. La codificación, por ejemplo, se realizaba mediante rotuladores de colores, subrayando los fragmentos de texto de un mismo código con el mismo color. Obviamente esto implicaba dificultades a la hora de distinguir códigos de color parecido y sobretodo implicaba horas y horas de dedicación en el conteo y la codificación en sí. No es de extrañar que hoy en día, existan programas informáticos específicos para hacer este análisis.

3.2 R

R es un sistema para análisis estadísticos y gráficos creado por Ross Ihaka y Robert Gentleman. Tiene una naturaleza doble, de programa y de lenguaje de programación, y es considerado como un dialecto del lenguaje S, creado por los Laboratorios AT&T Bell. Forma parte del sistema GNU y se distribuye gratuitamente bajo licencia GNU GPL (Licencia Pública General de GNU), estando disponible para los sistemas operativos *Windows*, *Macintosh*, *Unix* y *GNU/Linux*. Está dentro de los lenguajes interpretados (como *Java*) y no compilados (como *C*, *C++, Fortran*, *Pascal*…), lo cual significa que los comandos escritos en el teclado son ejecutados directamente sin necesidad de construir ejecutables.

Se trataría de uno de los lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy popular en el campo de la minería de datos, la investigación biomédica, la bioinformática y las matemáticas financieras, entre otros campos.

Una de las características más sobresaliente de R es su enorme flexibilidad. Mientras que programas más clásicos muestran directamente los resultados de un análisis, R sería un lenguaje orientado a objetos, de modo que se guardan estos resultados pudiéndose hacer un análisis sin necesidad de mostrar su resultado inmediatamente. A parte, que sea orientado a objetos significa que las variables, datos, funciones, resultados, etc., se almacenan en la memoria activa del computador en forma de objetos con un nombre específico. El usuario puede modificar o manipular estos objetos con operadores (aritméticos, lógicos, y comparativos) y funciones. Véase Figura 2.

R posee muchas funciones para análisis estadísticos y gráficos; estos últimos pueden ser visualizados de manera inmediata en su propia ventana y ser guardados en varios formatos (*jpg*, *png*, *bmp*, *ps*, *pdf*, *emf*, *pictex*, *xfig*…). Los resultados de análisis estadísticos se muestran en la pantalla, y algunos resultados intermedios (como valores *P*-, coeficientes de regresión, residuales…) se pueden guardar, exportar a un archivo, o ser utilizados en análisis posteriores. A parte, contribuye en la posibilidad de cargar diferentes bibliotecas o paquetes con funcionalidades de cálculo o graficación adicionales y para casos específicos. Una función clásica en R se puede delinear de la siguiente manera:

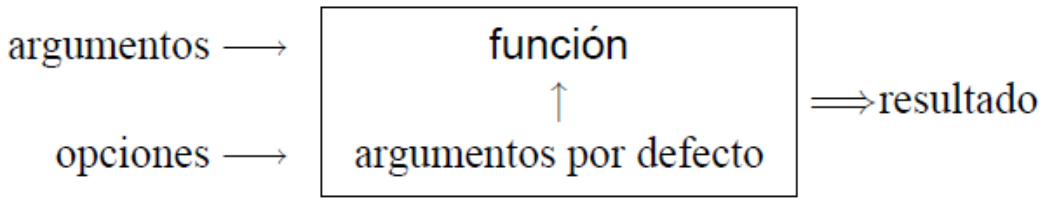


Figura 2. Esquema del funcionamiento de una función de R [2]

Los argumentos pueden ser objetos ("datos", fórmulas, expresiones…), algunos de los cuales pueden ser definidos por defecto en la función. Una función en R puede carecer totalmente de argumentos, ya sea porque todos están definidos por defecto, o porque la función realmente no tiene argumentos.

Tal y como se ha dicho anteriormente, todas las acciones en R se realizan con objetos que son guardados en la memoria activa del ordenador, sin usar archivos temporales. La lectura y escritura de archivos solo se realiza para la entrada y salida de datos y resultados (gráficas…). El usuario ejecuta las funciones con la ayuda de comandos definidos. Los resultados se pueden visualizar directamente en la pantalla, guardar en un objeto o escribir directamente en el disco (particularmente para gráficos). Debido a que los resultados mismos son objetos, pueden ser considerados como datos y analizados como tal. Archivos que contengan datos pueden ser leídos directamente desde el disco local o en un servido remoto a través de la red. Véase Figura 3.

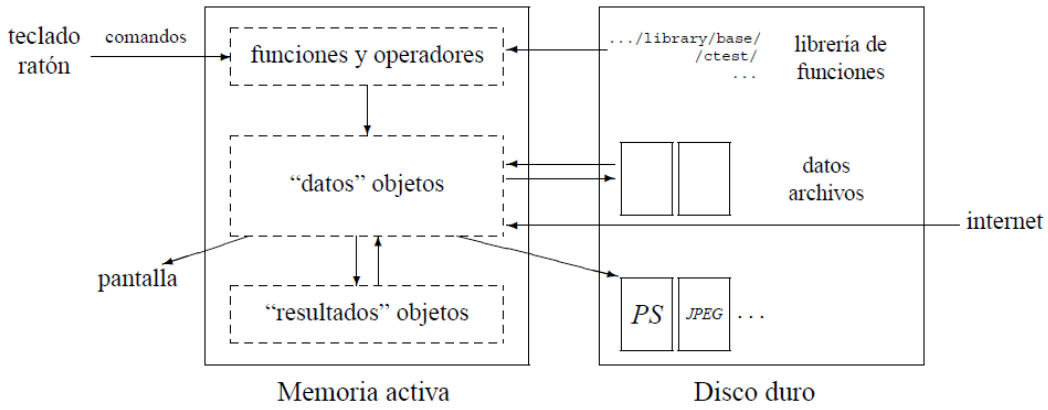


Figura 3. Esquema del funcionamiento de R [2]

Otra de las posibilidades que ofrece R es la de programar bucles (*loops*) para analizar conjuntos sucesivos de datos y la de combinar en un solo programa diferentes funciones estadísticas para realizar análisis más complejos.

A modo de resumen, los grandes atractivos que provee R son:

- El amplio abanico de herramientas estadísticas (modelos lineales y no lineales, pruebas estadísticas, series temporales, algoritmos de clasificación y agrupamiento, etc.) y gráficas.
- La capacidad de combinar, sin fisuras, análisis "preempaquetados" (ej., una regresión logística) con análisis ad-hoc, específicos para una situación.
- La gran capacidad gráfica. R permite generar gráficos con alta calidad, además posee su propio formato para la documentación basado en *LaTeX* por lo que es muy frecuente su uso en la generación de gráficos para *papers*.
- La capacidad de integración con distintas bases de datos. Además, existen bibliotecas que facilitan su utilización desde lenguajes interpretados como *Perl* y *Python*.
- La comunidad de usuarios y programadores. R cuenta con una comunidad de usuarios muy dinámica e integra estadísticas de gran de gran renombre. Esto hace que el número de paquetes y herramientas esté en constante crecimiento.
- Es un lenguaje orientado a objetos. Esto significa que las variables, datos, funciones, resultados, etc, se guardan en la memoria activa del ordenador en forma de objetos con un nombre específico. De esta forma el usuario puede modificar o manipular estos objetos con operadores (aritméticos, lógicos y comparativos) y funciones (que a su vez son objetos).
- Puede usarse como herramienta de cálculo numérico, por lo que puede ser tan eficaz como *Matlab* y *GNU Octave,* con una sintaxis que recuerda a C/C++.
- Se trata de un lenguaje de programación, lo que permite a los usuarios lo extiendan definiendo sus propias funciones. Gran parte de las funciones de R están escritas con el mismo R, aunque para algoritmos computacionalmente exigentes es posible desarrollar bibliotecas en *C*, *C++* o *Fortran* que se cargan dinámicamente. Los usuarios más avanzados pueden también manipular los objetos de R directamente desde código desarrollado en C. R también puede extenderse a través de paquetes desarrollados por su comunidad de usuarios.
- Se distribuye bajo licencia GNU GPL (Licencia Pública General de GNU). Ésta licencia garantiza a los usuarios finales (personas, organizaciones y compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software. Su propósito es declarar que el software cubierto por esta

licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios

3.3 Paquete de R

Un paquete de R es una forma de mantener colecciones de funciones y conjuntos de datos. Estos pueden ser cargados y descargados dinámicamente durante el tiempo de ejecución y por lo tanto sólo ocuparan memoria cuando se estén utilizando.

Los paquetes instalados están guardados en una librería localizada en el directorio de R que, a su vez, están estructurados en otros directorios. El paquete denominado "base" constituye el núcleo de R y contiene las funciones básicas del lenguaje para leer y manipular datos, algunas funciones gráficas y algunas funciones estadísticas (regresión lineal y análisis de varianza). El resto son meras formas de extender las posibilidades que ofrece R a través de nuevas funciones pero que no dependen de su núcleo principal.

R forma parte de un proyecto colaborativo y abierto. Sus usuarios pueden publicar paquetes que extienden su configuración básica, añadiendo o mejorando funciones. Algunos de esos implementan métodos estadísticos especializados, otros dan acceso a datos o a hardware, otros son complemento de libros de texto, etc.

Existe un repositorio oficial llamado *CRAN* (https://CRAN.R-project.org/) donde hay miles de paquetes publicados y listos para ser descargados por otros usuarios. Existirían otros como *Bioconductor* (https://www.bioconductor.org/) y *Omegahat* (http://www.omegahat.org/).

Dado el enorme número de nuevos paquetes, éstos se han organizado en vistas (o temas), que permiten agruparlos según su naturaleza y función. Por ejemplo, hay grupos relacionados con estadística bayesiana, econometría, series temporales, etc.

Las principales razones para su creación son:

- Nos obliga a pulir nuestras funciones, datos y código y, sobretodo, a documentar todo el trabajo y a dar ejemplos claros.
- Es un modo más elegante de compartir nuestro trabajo. Los posibles usuarios del paquete agradecerán el esfuerzo de mejora y documentación que requiere la creación de un paquete. Además, el intercambio con los usuarios también favorece la mejora del código.
- Es la forma establecida para contribuir al crecimiento de R.

Todo paquete de R tiene que tener una estructura mínima formada por:

- Un archivo R/ con el código.

- Un archivo DESCRIPTION con la información ("Metadatos") sobre el paquete.

Adicionalmente pueden existir otros muchos directorios con informaciones diversas:

- man/ es el directorio donde está la documentación de las funciones.
- Archivo inst/CITATION es el archivo que describe como citar a tu paquete.
- inst/doc/ es el directorio usado para la documentación de gran tamaño.
- Archivo NAMESPACE es el archivo que describe que funciones son parte de la API del paquete y están permitidas para otros usos.
- test/ y inst/test son directorios que contiene las unidades de prueba.
- data/ es el directorio que contiene archivos .rqda usados para muestras de datos.

3.4 RQDA

RQDA (*R-based Qualitative Data Analysis*) es un paquete de R implementado por HUANG Rounggui el 27-10-2014 y que va por la versión 0.2-7 [1]. Se puede encontrar gratuitamente en el repositorio CRAN (http://rqda.r-forge.r-project.org/) bajo licencia BSD-3.

El paquete RQDA requiere de los siguientes paquetes para funcionar:

- *Base* (vers. >=2.8.0). Es el paquete que contiene las funciones básicas de R.
- *RSQLite* (vers. >=1.0.0) y *DBI*. Son los paquetes que contienen las funciones para crear y acceder a la base de datos a través del lenguaje SQL.
- *gWidgetsRGtk2* (vers. >=0.0-36), *RGtk2*(vers. >=2.20) y *gWidgets* (vers. >=0.0-31). Son los paquetes que contienen las funciones para poder crear las GUI.
- *Igraph*. Es el paquete que contiene las funciones para crear gráficos de redes.

Es el único paquete conocido de R que contiene funciones para realizar un análisis cualitativo de datos. Una de ellas es la que llama a la GUI del programa con la que, a través de sus funcionalidades, se puede realizar la codificación de archivos de texto plano (.txt). Además, podemos clasificar los archivos en casos, categorías y atributos y los códigos en categorías. Con todo esto se irá construyendo las base de datos del proyecto en un archivo específico (.rqda).

El resto de funciones que contiene vendría a dar maneras de poder encontrar concurrencias entre códigos, concurrencias entre archivos y códigos, recuentos de las codificaciones, fragmentos codificados, etc.

4. Análisis del RQDA

4.1 Funcionalidades

Como se ha dicho en apartados anteriores, RQDA es un paquete de funciones de R del cual se destaca una de ellas que es *RQDA().* Ésta es la que llama a la interfaz gráfica del programa que está diseñada para realizar las primeras actividades de un análisis QDA, es decir, aquellas referentes a la reducción de información, mediante su codificación, para facilitar su procesamiento.

La GUI consta de una ventana que ocupa aproximadamente un tercio de la pantalla y dentro de esta contiene la libreta con 9 pestañas a través de las cuales se puede mover uno por las diferentes opciones del programa. Las pestañas son las siguientes:

- *Project.* Es la pestaña principal del programa y es la primera que aparece activa al ejecutarlo. Tiene 9 botones 7 de los cuales están bloqueados al iniciar. Con los dos únicos activos se puede o crear un nuevo proyecto o abrir uno ya existente. Una vez se ha hecho una de las dos opciones, el programa desbloquea el resto y permite la navegación por las otras páginas de la libreta.
  Los otros 7 botones son los de: cerrar proyecto, hacer un memo del proyecto, hacer un *backup* del proyecto, guardar el proyecto, limpiar el proyecto (borrar) y cerrar todas las codificaciones.
  Además, debajo se nos muestra el directorio del proyecto actual e información referente al paquete como su autor, el mail de contacto, el tipo de licencia y la versión del paquete.
- *Files.* Esta es la pestaña donde se añaden los diferentes archivos que contendrán la información sobre la que se quiere hacer el análisis cualitativo. Se puede hacer añadiendo el texto directamente o importándolo desde un documento *.txt*. Los archivos añadidos se muestran en forma de lista y siempre tenemos las opciones de eliminarlos, abrirlos, renómbralos, hacerles un memo o mostrar sus atributos.
- *Codes.* En esta pestaña es donde se crean los códigos que utilizaremos para codificar la información añadida en la pestaña *Files*. Los códigos creados se muestran en forma de lista y siempre se pueden eliminar, renombrar o hacerles un memo. Desde esta pestaña también se puede hacer la codificación de los archivos. Para ello, se debe, previamente, haber abierto el archivo deseado desde la pestaña *Files.* Con el archivo abierto marcamos o desmarcamos la codificación los fragmentos del texto para asignarles o quitarles el código directamente. Con el botón *coding* se muestra, en una ventana nueva, los fragmentos de texto codificados con ese código.

- *Code Categories*. La pestaña categorías de códigos es donde se agrupan y clasifican los códigos creados en la pestaña *Codes.* Las opciones que ofrece son muy similares a las otras pestañas: añadir, eliminar, renombrar y hacer memo de la categoría; pero también tiene botones para añadir o quitar los códigos de éstas. Las categorías creadas se muestran en forma de lista y a medida que se van seleccionando se muestran los códigos que las forman en una lista inferior. Desde esta última lista también se puede realizar la codificación de los archivos de la misma manera que se haría en *Codes*.
- *Cases.* Esta pestaña permite clasificar los archivos en diferentes casos para el estudio. Para ello, dispone de los botones antes vistos de añadir, eliminar y renombrar casos. En este apartado existe, además, la posibilidad de añadir atributos. Hay también la opción *profile* que muestra el recuento de códigos por cada caso.
  Al igual que en la pestaña *Code Categories,* se muestran los casos creados en una lista y a medida que se seleccionan se muestran los archivos en la lista inferior.
- *Attributes*. Esta opción permite crear los atributos que se podrán utilizar para caracterizar tanto archivos como casos. Éstos pueden ser numéricos, palabras o no especificados. Se sigue teniendo las opciones de eliminar, renombrar y hacer un memo.
- *File Categories*. En esta pestaña, de una forma similar a la de *Cases,* agruparemos los archivos en categorías. La principal diferencia entre ambas es que utilizaremos las categorías de los archivos para indicar la naturaleza de la fuente o tipo de archivo (ej. si es la transcripción de una entrevista o si un fragmento de texto de una página web) y los casos para separar los archivos en función de las características de estudio (ej. el lugar del que es proveniente la persona a la que se le ha hecho una entrevista). La forma de disposición de los datos será como con las dos listas de las pestañas anteriores.
- *Journals*. En esta penúltima pestaña, el programa permite que se vaya anotando el progreso de la codificación mediante una vista de texto. Una forma muy útil de que el usuario tenga un recordatorio de lo que lleva hecho y lo que le falta por hacer.
- *Settings.* En la última, se manejan diferentes formas para customizar la aplicación. Nos encontramos opciones como cambiar el color y el tamaño de la fuente de los archivos, cambiar el color de las codificaciones hechas, cambiar el cifrado de la fuente, mostrar las propiedades de los archivos u obtener una recuperación de las codificaciones por archivos o casos.

La GUI descrita lo que básicamente hace es crear una base de datos formada por tablas con diferentes relaciones entre ellas. Estás están descritas de forma detallada en el manual para que se puede acceder a ellas a través del paquete

RSQLite, que implementa el lenguaje SQL para R. Sin embargo, RQDA ofrece, para disposición del usuario, una serie de funciones adicionales que evitan que haga falta tener conocimientos avanzados en SQL.

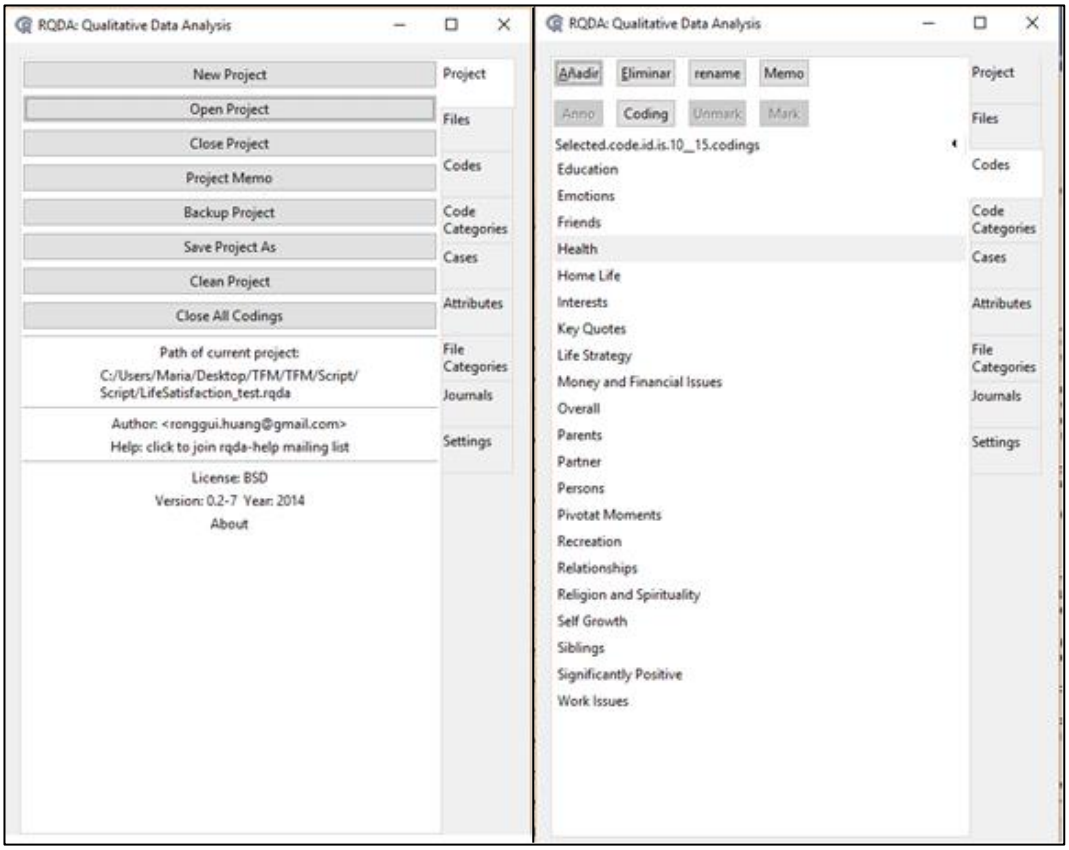


Figura 4. Captura del menú y de la pestaña "Codes" de RQDA

Dichas funciones son de utilidad para realizar las siguientes tareas de un análisis cualitativo y hay alrededor de unas 20. Las más significativas y relevantes para el investigador serían las siguientes:

- *RQDAQuery*(). Permite hacer una consulta de la base de datos del proyecto mediante una instrucción de SQL.
- *exportCodings*(). Exporta todas las codificaciones hechas con la interfaz en formato html.
- *summaryCodings*(). Da una matriz con un resumen de las codificaciones: número de codificaciones por código, longitud media de codificación por código, número de archivos donde sale el código y número de codificaciones por archivo.
- *filesCodedByAnd*(), *filesCodedByOr*(), *filesCodedByNot*(). Da los identificadores de los archivos que están codificados por un vector de códigos con la expresión lógica correspondiente.

- *casesCodedByAnd*(), *casesCodedByOr*(), *casesCodedByNot*(). Da los identificadores de los casos que están codificados por un vector de códigos con la expresión lógica correspondiente.
- *getFileNames*(), *getCasesNames*(). Da el nombre de los archivos y casos respectivamente a partir del vector de identificadores dado.
- *crossCodes*(). Encuentra la concurrencia de códigos a partir de un vector de códigos dados y en función al tipo de relación: inclusión, proximidad, superposición y exactitud.
- *crossTwoCodes*(). Encuentra la concurrencia entre 2 pares de códigos según el tipo de relación indicado: inclusión, proximidad, superposición y exactitud.
- *getCodingTable*(). Muestra la tabla de la base de datos correspondiente a las codificaciones.
- *filesByCodes*(). Da la tabla Archivo vs Código indicando con 1 y 0 si hay concurrencias.
- *getCodingsByOne*(). Encuentra y muestra las codificaciones de los códigos indicados mediante su identificador.

4.2 Carencias y soluciones

Tal y como se ha comentado en la justificación, el motivo que dado lugar a este proyecto es suplir y mejorar todas las carencias que tiene el paquete en el tratamiento de los datos una vez codificados y categorizados.

Las principales carencias que se han identificado y que destacan los investigadores que usan RQDA son:

- GUI para el tratamiento de los datos cuando se ha realizado la codificación. Esta es la principal carencia del paquete, no sólo por lo farragoso que es trabajar con las funciones sino también porque hay investigadores que no tienen nociones de programación y utilizan RQDA por su GUI.
- No tiene opciones de graficación ni en la GUI ni en las funciones. Para realizar gráficos se ha de recurrir a otros tipos de paquetes.
- Las funciones que dispone RQDA son farragosas. Hay veces que en los argumentos se piden identificadores y otras veces nombres, en otras ocasiones buscamos nombres, pero nos dan identificadores. Esto implica llamar continuamente a las tablas de las bases de datos para saber que identificador va con cada nombre e ir apuntándolo en hojas de papel.
- Se pueden ver concurrencias entre códigos con una función, pero se presentan de forma desordenada y no se ven de forma clara con la consola de R. Estaría bien que se presentará en forma de tabla Códigos vs Códigos y se mostrará en una GUI.
- Se pueden ver concurrencias entre archivos y códigos con una función, pero se presentan de forma desordenada y sólo muestra 1s y 0s

indicando si hay concurrencia o no, respectivamente. Estaría bien verlo en forma de tabla Archivos vs Códigos y estaría bien ver el número de concurrencias que hay (no sólo 1s y 0s). Aparte, como en la matriz de sólo códigos no se ve de forma clara con la consola.
- El sistema de filtraje para las tablas de concurrencias no es muy avanzado. Estaría bien poder filtrar por códigos, archivos, categorías y casos.
- Con las funciones que da RQDA hay cosas que sólo puedes hacer si tienes nociones de R y sus funciones.

Para suplir todas estas carencias, sobretodo la primera, se ha tratado de identificar qué tipos de análisis para el tratamiento de datos debe de hacer el programa. En un análisis de datos se pueden hacer gran cantidad de análisis, pero no todos ellos se usan con la misma frecuencia. Es por eso que, como dice Pareto y como se ha dicho en la justificación, se dedicará el 20% del esfuerzo en resolver el 80% de los análisis más frecuentes.

Se ha identificado que la GUI debería tener lo siguiente:

- Análisis de las concurrencias entre códigos mediante tabla Código vs Código y mediante diagramas de red. Con esto se analiza si pueden establecer relaciones o no entre los códigos.
- Análisis de las concurrencias entre archivos y códigos mediante tabla Código vs Archivo y mediante diagramas de red. Con esto se analiza si pueden establecer relaciones o no entre los códigos y los archivos/casos.
- Análisis de las codificaciones mediante la consulta de éstas. Estudiar qué es lo que se dice en determinados códigos.
- Análisis de las frecuencias de las codificaciones mediante tablas y gráficos. Mirar con herramientas estadísticas la frecuencia en la que aparecen las codificaciones en los diferentes archivos/casos.
- Análisis de los archivos mediante sus atributos. Tener la clasificación de los archivos y los casos en función de sus atributos.

5. Diseño del paquete

El paquete objeto de este proyecto se ha diseñado con la intención de cubrir con una GUI lo que no cubre RQDA. Es por eso se le ha dado el nombre de RQDAExt (*R-based Qualitative Data Analysis Extension*), ya sirve para indicar que el paquete será una extensión de la GUI y de las funciones de RQDA, pero a su vez complemento, es decir, requiere de un uso previo de la otra.

Lo primero que se ha hecho ha sido emplear los mismos paquetes de implementación de la GUI y que se recogen en el apartado 3.4. Así como para RQDA el programador decidió utilizar las funciones de gWidgetsRGtk2, en RQDAExt se emplean más las funciones de RGtk2. La principal razón es porque muchas de las funcionalidades que se buscaban sólo no se podían realizar con RGtk2. gwidgetsRGtk2 es un paquete que implementa las herramientas de gwidgets con el paquete RGtk2. Las interfaces de gwidgets implementan interfaces de bajo nivel que son mucho más simples, pero menos potentes que las herramientas nativas (como RGtk2) y están pensadas para aquellos programadores que no quieren invertir mucho tiempo en perfeccionar su GUI. Por el otro lado, RGtk2 (enlace entre R y la plataforma GTK+) es mucho más rico y potente, siendo un lenguaje de alto nivel, lo cual permite mayor control y más características especiales para la GUI.

Dado la complementariedad que tenía que tener el paquete, se ha buscado que tenga un diseño y un estilo parecido a su predecesor. Para ello la interfaz está basada en una ventana con un menú principal a través del cual el usuario podrá moverse por las diferentes funcionalidades del programa. Dicha navegación se efectúa con unos botones, en contra del RQDA que usa una libreta. La distribución de éste se basa en dos partes separadas horizontalmente. La izquierda para los botones y la derecha para la información referente al paquete y al proyecto abierto: el directorio del proyecto, el nombre del autor, la versión del paquete, etc. Véase Figura 5.

En la Figura 5 se puede observar la distribución de los widgets en el menú y en la Tabla 1 su finalidad que se corresponderán con los análisis indicados en el apartado 4.2.

Botón	Descripción
Open Proyect	Abre un proyecto con .rqda que contenga la base de datos con la que realizar el análisis. Es necesario abrir un proyecto o bien con RQDA o con RQDAExt para hacer servir las otros widgets del programa.
Close Proyect	Cierra el proyecto .rqda abierto.
Code Vs Code Analysis	Llama a una ventana con la GUI del análisis Código-Código.

Botón	Descripción
Code Vs File Analysis	Llama a una ventana con la GUI del análisis Código-Archivo.
Coding Analysis	Llama a una ventana con la GUI del análisis de codificaciones.
Summary of Codings Analysis	Llama a una ventana con la GUI del resumen de códigos.
Attributes Analysis	Llama a una ventana con la GUI de la búsqueda de atributos.
Help	Llama a una ventana dando indicaciones sobre cómo usar el programa.

Tabla 1. Distribución del menú de RQDAExt

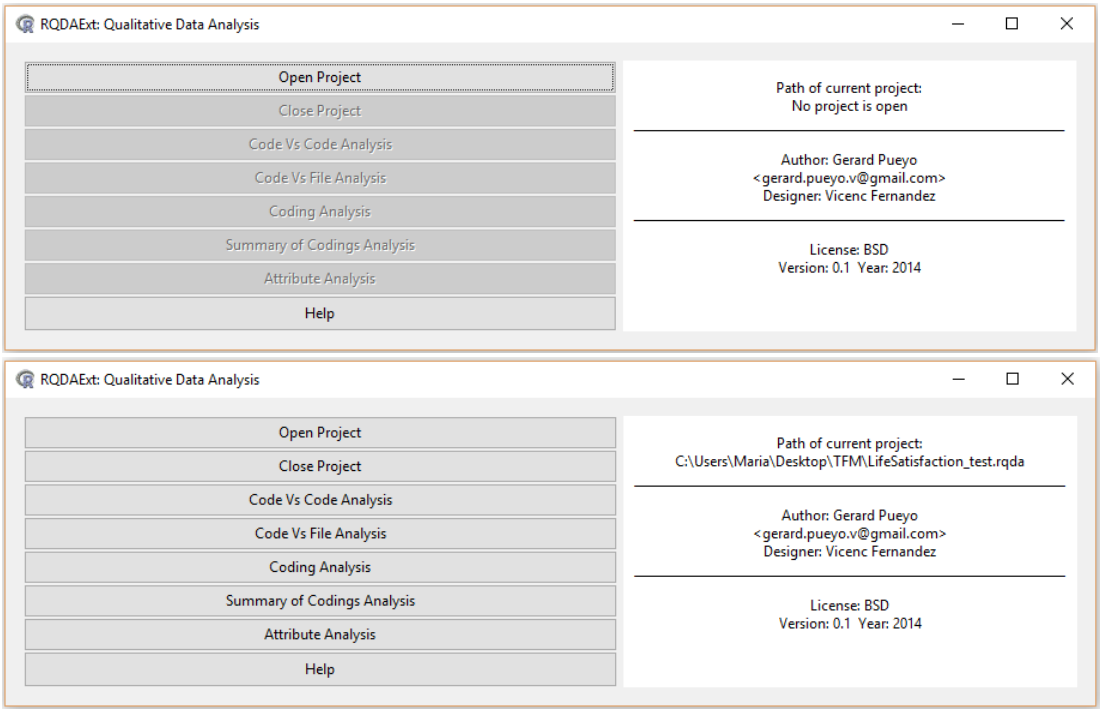


Figura 5. Distribución del menú de RQDAExt sin y con un proyecto

Para las interfaces a las que se accede a través del menú, se ha buscado dividir la ventana verticalmente en dos mitades. En la mitad izquierda se ha puesto lo que vendría a ser el sistema de filtraje del análisis, dónde el usuario selecciona los valores que quiere analizar; y los botones que permiten volver al menú principal. En la otra, la derecha, se ha decidido que sería el lugar dónde se verían las tablas, los gráficos y los textos, así como todos los widgets para manipularlos.

5.1 Análisis Código-Código

El diseño del análisis código-código se ha hecho buscando lo siguiente:

- Que el usuario pudiera filtrar los códigos de las filas y columnas de la matriz por códigos o categorías.
- Que el usuario pudiera filtrar los archivos donde encontrar las concurrencias por archivos, casos o categorías.
- Que el usuario pudiera seleccionar el tipo de relación de concurrencia entre los códigos. Existen 4 opciones:
    - Exactitud. Contará las concurrencias de los códigos si y sólo si el texto codificado es exactamente el mismo para ambos códigos.
    - Superposición. Contará las concurrencias de los códigos si y sólo si el texto codificado de uno de los códigos está superpuesto al del otro, es decir, el final del texto de uno incluye el principio del otro.
    - Proximidad. Contará las concurrencias de los códigos si y sólo si el texto codificado de uno está próximo a x unidades del otro.
    - Inclusión. Contará las concurrencias de los códigos si y sólo si el texto codificado de uno ésta dentro del texto codificado del otro sin comprender la totalidad del texto.
- Que el usuario pudiera exportar la tabla de concurrencias para los formatos Excel y texto plano.
- Que el usuario pudiera tuviese 2 formas diferentes de visualizar la relación entre los códigos: una en formato tabla y 2 en forma de gráfico.
- Que el usuario tuviera la opción de exportar los gráficos en el formato imagen que deseara.
- Que el usuario pudiera exportar, en formato Excel, los vértices y las aristas de los gráficos de red.

En la Figura 6 y la Tabla 2 podemos ver una captura de la ventana, así como la ubicación de diferentes partes que hay.

Ref.	Elementos	Descripción
1	Sistema de filtrado	Este conjunto de elementos permite que el usuario pueda seleccionar mediante un sistema de filtrado cuáles son los códigos que aparecerán en las filas y columnas de la matriz y de que archivos se hará el recuento de las concurrencias entre éstos. Por cada unidad de filtrado hay una caja con las opciones de filtro: códigos y categorías para los códigos y archivos, casos y categorías para los archivos. A medida que se eligen las opciones, el programa muestra en tablas la lista de los valores de éstos para que se seleccionen los que se desean añadir a las tablas de concurrencias y los

Ref.	Elementos	Descripción
		gráficos. Encontramos también, botones para seleccionarlo todo.
2	Tipo de relación	A través de una caja el usuario escogerá cuál quiere que sea la relación de concurrencia entre los códigos: exactitud, superposición, inclusión y proximidad.
3	Botones *Close* y *Help*	El botón *Close* es el botón que devuelve al usuario al menú principal cerrando la ventana y el botón *Help* abre una ventana en pantalla explicando el funcionamiento del análisis.
4	Visor de la tabla	En esta parte de la ventana, en la primera pestaña de la libreta, es donde se puede visualizar la tabla de concurrencias.
5	Botones visor de la tabla	Los botones del visor tienen funcionalidades para manejar el visor de la tabla y son los siguientes: • *Clear*. Limpia el visor dejándolo todo en blanco. • *Export in txt*. Exporta la tabla de concurrencias en formato txt. • *Export in csv*. Exporta la tabla de concurrencias en formato csv (Excel). • *Print*. Coge los valores seleccionados de (1) y (2) e imprime la matriz de concurrencias en el visor borrando, si lo hubiera, la matriz anterior.
6	Visor de gráficos	Es la parte de la ventana donde se visualizan los gráficos (segunda pestaña de la libreta). Cuenta también con una libreta para cambiar entre los diferentes ajustes de los dos gráficos que se pueden hacer: de red y mapa de calor. Dentro de esta libreta se encuentran diferentes widgets que permiten cambiar los aspectos principales del gráfico.
7	Botones visor de gráficos.	Los botones del visor de gráficos tienen diferentes funcionalidades y son: • *Clear.* Deja la vista de gráficos en blanco. • *Export Nodes.* Exporta los vértices del gráfico en formato csv (Excel). Pensado para aquellos que desean usar el paquete *igraph* directamente desde R. • *Export Links.* Exporta las aristas del gráfico en formato csv (Excel). Pensado para aquellos que desean usar el paquete *igraph* directamente desde R.

Ref.	Elementos	Descripción
		• *Export graph*. Exporta el gráfico del visor en el formato de imagen que desee el usuario. • *Plot*. Coge los valores seleccionados de (1), (2) y (6) y dibuja el gráfico en la vista borrando, si lo hubiera, el grafico anterior.

Tabla 2. Distribución del Code vs Code Analysis de RQDAExt

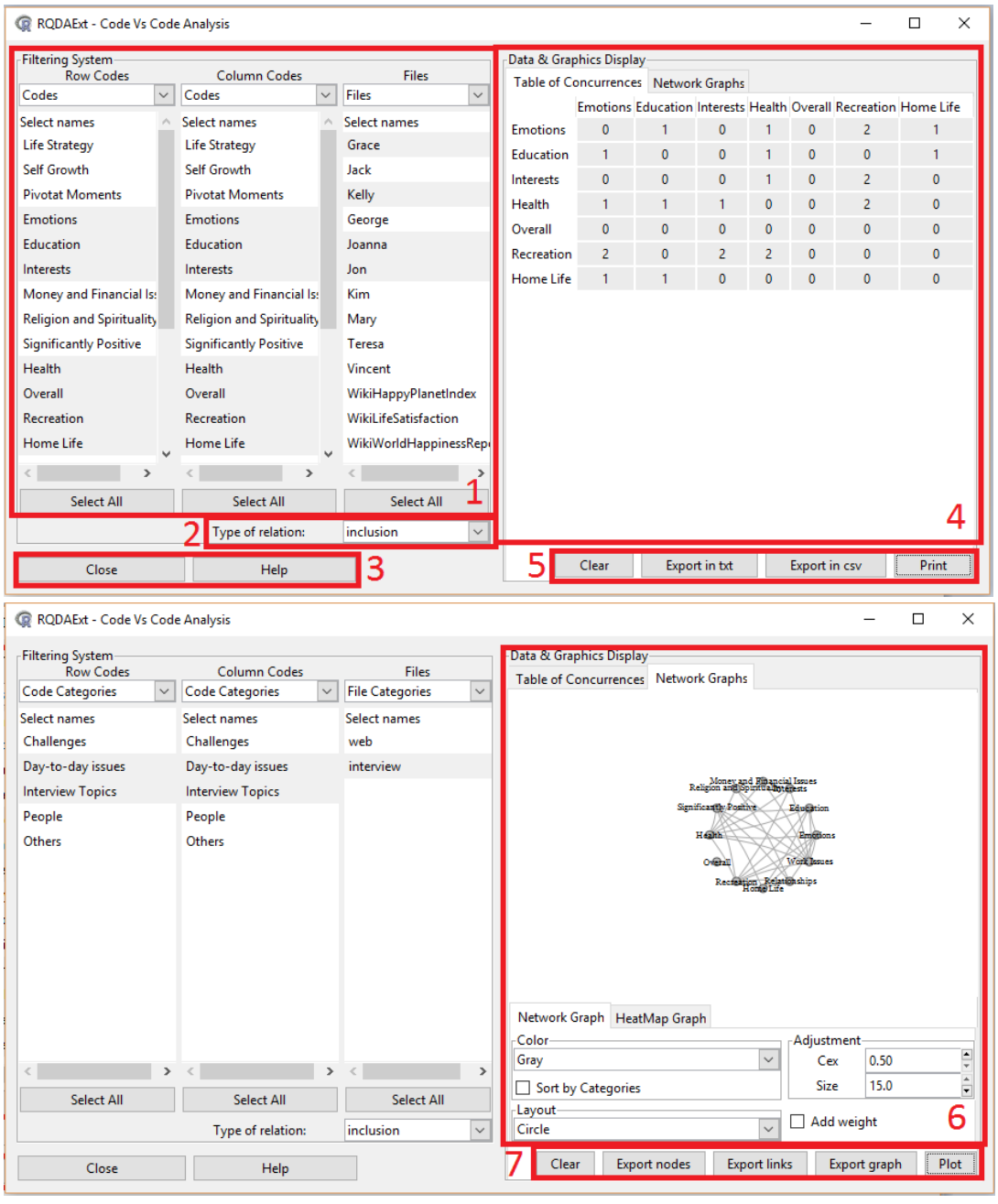


Figura 6.Distribución del Code vs Code Analysis de RQDAExt

5.2 Análisis Archivo-Código

El diseño de la ventana que realiza el análisis Archivos vs Códigos presenta un diseño similar al del análisis Códigos vs Códigos puesto que se buscaba prácticamente lo mismo:

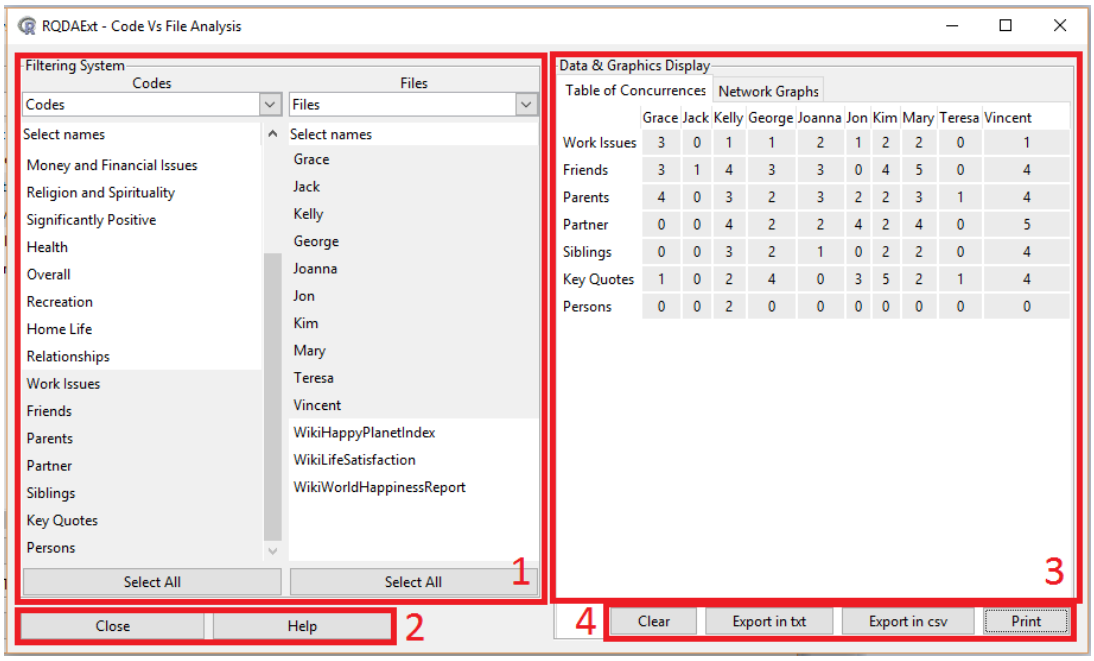
- Que el usuario pudiera filtrar los códigos por categorías o códigos.
- Que el usuario pudiera filtrar los archivos por archivos, casos o categorías.
- Que el usuario pudiera exportar la tabla de concurrencias para los formatos Excel y texto plano.
- Que el usuario tuviese 2 formas diferentes de visualizar la relación entre los códigos: una en formato tabla y 2 en forma de gráfico.
- Que el usuario tuviera la opción de exportar los gráficos en el formato imagen que deseara.
- Que el usuario pudiera exportar, en formato Excel, los vértices y las aristas de los gráficos de red.

A continuación, en la Figura 7 y la Tabla 3 podemos ver la ubicación de los elementos.

Ref.	Elementos	Descripción
1	Sistema de filtrado	El sistema de filtraje para este análisis es parecido al del análisis Código vs Código, pero sólo tiene 2 columnas, una para los códigos, filtrado por códigos y categorías; y otra para los archivos, filtrado por archivos, casos y categorías.
2	Botones *Close* y *Help*	El botón *Close* es el botón que devuelve al usuario al menú principal cerrando la ventana y el botón *Help* abre una ventana en pantalla explicando el funcionamiento del análisis.
3	Visor de la tabla	En esta parte de la ventana, en la primera pestaña de la libreta, es donde se puede visualizar la tabla de concurrencias.
4	Botones visor de la tabla	Los botones del visor tienen funcionalidades para manejar el visor de la tabla y son los siguientes: • *Clear*. Limpia el visor dejándolo todo en blanco. • *Export in txt*. Exporta la tabla de concurrencias en formato txt. • *Export in csv*. Exporta la tabla de concurrencias en formato csv (Excel). • *Print*. Coge los valores seleccionados de (1) y (2) e imprime la matriz de concurrencias en el visor borrando, si lo hubiera, la matriz anterior.

Ref.	Elementos	Descripción
5	Visor de gráficos	Es la parte de la ventana donde se visualizan los gráficos (segunda pestaña de la libreta). Cuenta también con una libreta para cambiar entre los diferentes ajustes de los dos gráficos que se puede hacer: de red y mapa de calor. Dentro de esta libreta se encuentran diferentes widgets que permiten cambiar los aspectos principales del gráfico.
6	Botones visor de gráficos.	Los botones del visor de gráficos tienen diferentes funcionalidades y son: • *Clear.* Deja la vista de gráficos en blanco. • *Export Nodes.* Exporta los vértices del gráfico en formato csv (Excel). Pensado para aquellos que desean usar el paquete *igraph* directamente desde R. • *Export Links.* Exporta las aristas del gráfico en formato csv (Excel). Pensado para aquellos que desean usar el paquete *igraph* directamente desde R. • *Export graph*. Exporta el gráfico del visor en el formato de imagen que desee el usuario. • *Plot*. Coge los valores seleccionados de (1), (2) y (6) y dibuja el gráfico en la vista borrando, si lo hubiera, el grafico anterior.

Tabla 3. Distribución del Code vs File Analysis de RQDAExt



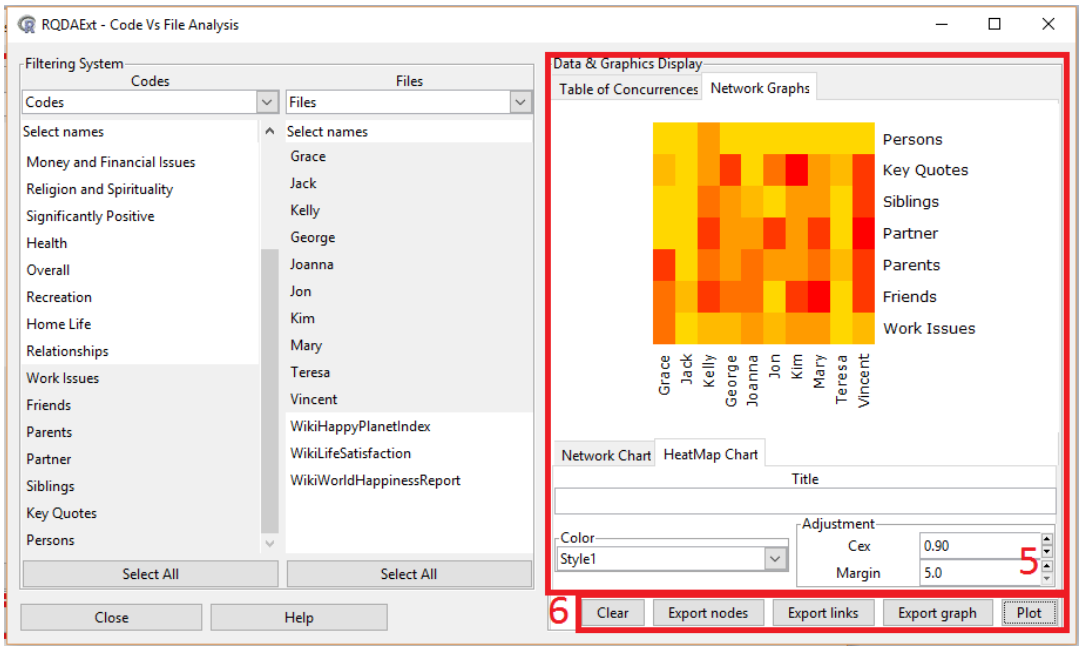


Figura 7. Distribución del Code vs File Analysis de RQDAExt

5.3 Análisis de codificaciones

Para la ventana de la búsqueda de codificaciones se ha buscado:

- Que el usuario pudiera escoger entre uno o más códigos para buscar las codificaciones.
- Que pudiera filtrar los códigos por códigos o categorías.
- Que el usuario pudiera escoger los archivos en donde buscar las codificaciones filtrando por archivos, casos o categorías.
- Que el usuario pudiera establecer condiciones lógicas entre los códigos escogidos. Existen 2 opciones.
    - AND. Se buscan todas las codificaciones que coincidan con todos los códigos seleccionados.
    - OR. Se buscan todas las codificaciones que hablen de los códigos seleccionados.
- Que el usuario pudiera exportar las codificaciones buscadas en formato de texto plano.

En la tabla 4 y la Figura 8 se puede ver como ha quedado su diseño y los elementos.

Ref.	Elementos	Descripción
1	Sistema de filtrado	El sistema de filtraje para este análisis funciona de la misma manera que el del análisis Código vs Archivo. Dos columnas, una para los códigos y la otra para los archivos.

Ref.	Elementos	Descripción
2	Tipo de conjunción lógica	Caja que permite seleccionar el tipo de conjunción lógica entre los códigos marcados. En caso de seleccionar un código su valor es indiferente.
3	Botones *Close* y *Help*	El botón *Close* es el botón que devuelve al usuario al menú principal cerrando la ventana y el botón *Help* abre una ventana en pantalla explicando el funcionamiento del análisis.
4	Visor de texto	En esta parte de la ventana en la que visualizar el texto de las codificaciones encontradas.
5	Botones visor de texto	Conjunto de botones que permiten tocar diferentes funcionalidades del texto. • *Clear.* Deja el visor de texto en blanco. • *Export en txt.* Exporta el texto del visor en formato .txt. • *Print*. Coge los valores de (1) y (2) e imprime el texto acorde a las características en el visor.

Tabla 4. Distribución del Coding Analysis de RQDAExt

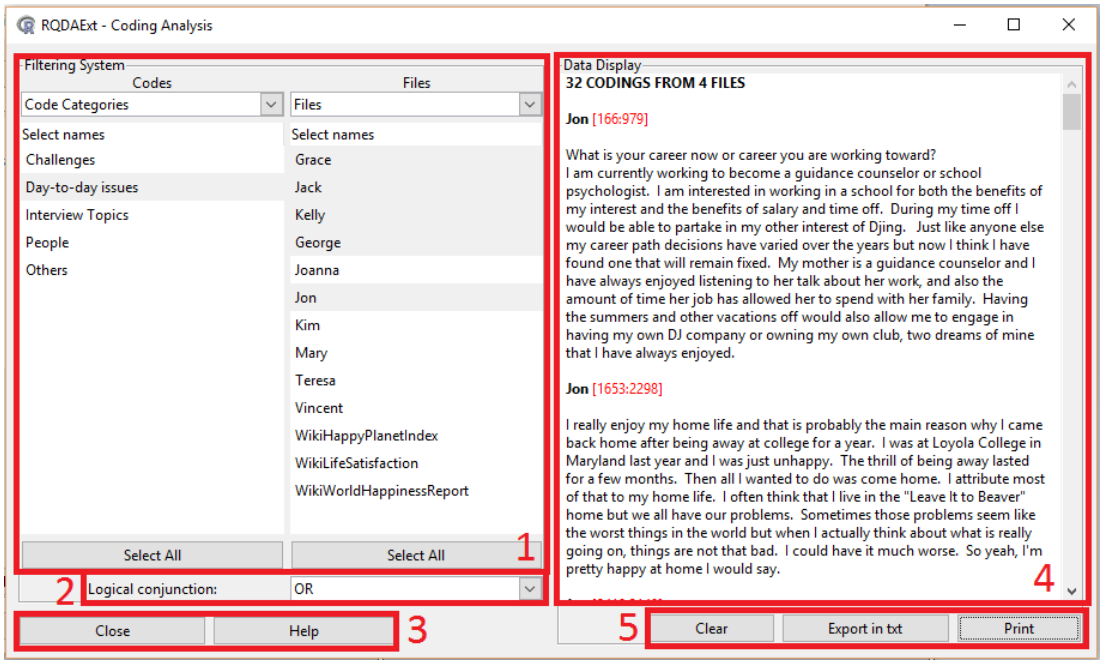


Figura 8. Distribución del Coding Analysis de RQDAExt

5.4 Análisis resumen de codificaciones

Con este análisis lo que se pretende es llevar las frecuencias de aparición de las codificaciones y se ha diseñado la GUI para que cumpla con lo siguiente:

- Que el usuario pudiera filtrar los códigos de los que se deseará hacer el conteo mediante códigos o categorías.
- Que el usuario pudiera filtrar los archivos de los cuáles buscar los códigos en archivos, casos y categorías.
- Que tuviera 4 cosas diferentes a contar:
    - Número de codificaciones por código. Cuenta el número de codificaciones de un código entre los archivos seleccionados.
    - Longitud media de las codificaciones. Dice la longitud media de las codificaciones de un código entre los archivos seleccionados.
    - Número de archivos por codificación. Cuenta el número de archivos, de los seleccionados, en los cuales sale una codificación de ese tipo.
    - Número de codificaciones por archivo. Cuenta el número de codificaciones totales de un archivo.
- Que el usuario, aparte de ver los resultados en formato tabla, pudiera ver los resultados de formato gráfico de 3 formas diferentes.
- Que el usuario pudiera exportar los resultados en formato Excel por si desea graficar desde allí.
- Que el usuario pudiera exportar los gráficos en el formato imagen que deseara.

En la Figura 9 y la Tabla 5 podemos ver en una captura de la ventana, así como la ubicación de diferentes partes.

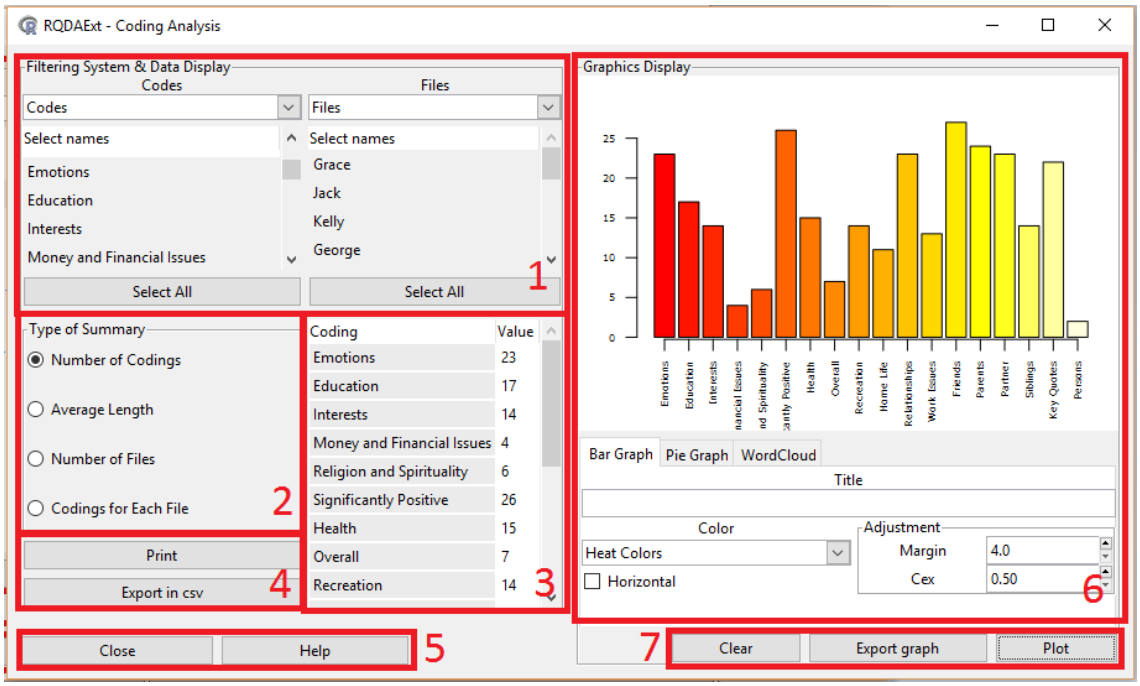


Figura 9. Distribución del Summary Of Codings de RQDAExt

Ref.	Elementos	Descripción
1	Sistema de filtrado	El sistema de filtraje para este análisis funciona de la misma manera que el del análisis Código vs Archivo. Dos columnas, una para los códigos y la otra para los archivos.
2	Seleccionador tipo de conteo	Caja que permite seleccionar entre los tipos de opciones a contar.
3	Visor de tabla	Elemento a través del cual se hará visible los resultados del conteo.
4	Botones del visor de tabla	Se agrupan dos botones: • *Print*. Coge los valores seleccionados en (1) y (2) e imprime en el visor de tabla los resultados. • *Export in csv*. Exporta la tabla del visor en formato Excel.
5	Botones *Close* y *Help*	El botón *Close* es el botón que devuelve al usuario al menú principal cerrando la ventana y el botón *Help* abre una ventana en pantalla explicando el funcionamiento del análisis.
6	Visor de gráficos.	En esta parte de la ventana en la que visualizar los gráficos. Contiene además una libreta con los diferentes gráficos y ajustes a realizar. Los 3 tipos de gráficos son: • Gráfico de barras • Gráfico circular • Nube de palabras Dentro de la libreta se encuentran diferentes widgets para modificar la apariencia del gráfico.
7	Botones visor de gráficos	Conjunto de botones que permiten tocar diferentes funcionalidades de los gráficos. • *Clear.* Deja el visor en blanco. • *Export graph*. Exporta el gráfico del visor en el formato de imagen que desee el usuario. • *Plot*. Coge los valores seleccionados de (1), (2) y (6) y dibuja el gráfico en la vista borrando, si lo hubiera, el grafico anterior.

Tabla 5. Distribución del Summary Of Codings de RQDAExt

5.5 Análisis de atributos

Para el análisis de atributos se requería:

- Que el usuario pudiera consultar los atributos de cada archivo o caso de forma rápida.
- Que el usuario pudiera tener clasificados los archivos o los casos en función de los valores de sus atributos.

Para hacerlo se decidió dividir ventana en dos mitades En la primera mitad, se ha puesto los elementos necesarios para resolver el primer requerimiento y en la segunda, el otro. En la Figura 10 y la Tabla 6 podemos ver como ha quedado y su funcionamiento.

Ref.	Elementos	Descripción
1	Sistema de filtrado	Sistema de filtraje entre archivos y casos para el primer requerimiento. Incluye la lista de elementos para ser seleccionados.
2	Visor de tabla	Tabla que permite seleccionar un código o caso del cual buscar los atributos. Incluye el botón *Print*, que recoge la información de (1) para realizar la tabla.
3	Botones *Close* y *Help*	El botón *Close* es el botón que devuelve al usuario al menú principal cerrando la ventana y el botón *Help* abre una ventana en pantalla explicando el funcionamiento del análisis.
4	Sistema de filtrado	Segundo sistema de filtraje entre archivos y casos para el segundo requerimiento.
5	Seleccionador del atributo	Elemento que muestra la lista de atributos y a través del cual se clasificaran los archivos o casos.
6	Visor de texto	Parte de la ventana donde se imprimirán las archivos o casos clasificados en función de los valores que tome el atributo escogido.
7	Botones del visor de texto	Conjunto de botones que permiten tocar diferentes funcionalidades del texto. • *Clear.* Deja el visor de texto en blanco. • *Export en txt.* Exporta el texto del visor en formato .txt. • *Print*. Coge los valores de (4) y (5) e imprime el texto acorde a las características en el visor.

Tabla 6. Distribución del Attribute Analysis de RQDAExt

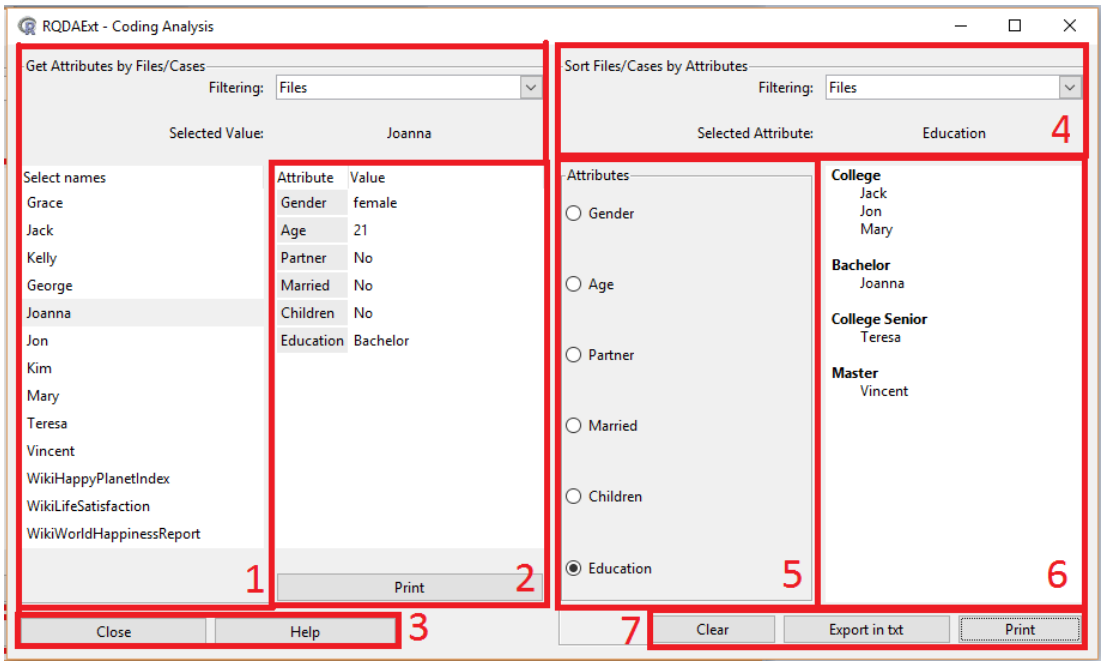


Figura 10. Distribución del Attribute Analysis de RQDAExt

5.6 Funciones generales

A parte de la GUI, el paquete incluye también una serie de funciones generales. Estas funciones son auxiliares y se han diseñado con la intención de ser llamadas a través de la GUI pero también están pensadas para que puedan ser utilizadas por un investigador con nociones de programación. Dichas funciones son las que se muestran en la Tabla 10.

Función	Argumentos	Uso
CodeCodeMatrix: Esta función se encarga de crear la matriz Código vs Código.	filter	Vector de longitud 3 indicando que clase de ids se van a pasar a través de los otros argumentos. Las posiciones del vector indicarían el tipo de los ids de las filas, los ids de las columnas y los ids de los archivos, por ese orden.
	rowcid	Vector de los ids que irán en las filas de la matriz. Los ids han de corresponderse con los códigos o las categorías, en función de lo indicado con argumento *filter*.
	colcid	Vector de los ids que irán en las columnas de la matriz. Los ids han de corresponderse con los códigos o las categorías, en función de lo indicado con argumento *filter*.

Función	Argumentos	Uso
	fid	Vector de los ids de los archivos utilizados en la matriz. Los ids han de corresponderse con los archivos, casos o las categorías, en función de lo indicado con argumento *filter*.
	relation	Tipo de relación que han de tener los códigos para contar concurrencias. Cuatro valores posibles: *inclusión*, *overlap*, *proximity* y *exact*.
	as.matrix	*TRUE* o *FALSE*. Se usa para señalar si la tabla será dada en formato matriz o no. Dicho argumento ha sido utilizado simplemente para dar mejor estética de la tabla dentro de la GUI. Por defecto el valor es *TRUE*.
CodeFileMatrix: Esta función se encarga de crear la matriz Código vs Archivo.	filter	Vector de longitud 2 indicando que clase de ids se van a pasar a través de los otros argumentos. Las posiciones del vector indicarían el tipo de los ids de los códigos y los ids de los archivos, por ese orden.
	cid	Vector de los ids de los códigos utilizado en la matriz. Los ids han de corresponderse con los códigos o las categorías, en función de lo indicado con argumento *filter*.
	fid	Vector de los ids de los archivos utilizados en la matriz. Los ids han de corresponderse con los archivos, casos o las categorías, en función de lo indicado con argumento *filter*.
	as.matrix	*TRUE* o *FALSE*. Se usa para señalar si la tabla será dada en formato matriz o no. Dicho argumento ha sido utilizado simplemente para dar mejor estética de la tabla dentro de la GUI. Por defecto el valor es *TRUE*.
getCodings: Esta función da el texto codificado por los códigos dados.	filter	Vector de longitud 2 indicando que clase de ids se van a pasar a través de los otros argumentos. Las posiciones del vector indicarían el tipo de los ids de los códigos y los ids de los archivos, por ese orden.
	cid	Vector de los ids de los códigos. Los ids

Función	Argumentos	Uso
		han de corresponderse con los códigos o las categorías, en función de lo indicado con argumento *filter*.
	fid	Vector de los ids de los archivos utilizados en la matriz. Los ids han de corresponderse con los los archivos, casos o categorías, en función de lo indicado con argumento *filter*.
	condition	*AND* o *OR*. Señala que clase de relación ha de haber entre los códigos dados. En caso de haber un sólo código, su valor puede ser cualquiera de los dos.
getAttributesbyFile: Esta función da los atributos de un archivo o caso y el valor que toman.	filter	Valor indicando si estamos buscando atributos de un archivo o un caso.
	fid	Valor indicando el id del archivo o caso (en función de lo indicado en *filter*) del cual buscar sus atributos
getFilebyAttributes: Esta función da los archivos o casos clasificados por los valores que toman con el atributo dado.	filter	Valor indicando si estamos buscando archivos o casos.
	attribute	Valor del atributo del cual se desea buscar los archivos o casos.
getLinksAndNodes: Esta función da la lista de nodos y aristas de una matriz de concurrencias, para que pueda ser graficada en un diagrama de red.	matrix	Matriz de concurrencias de la cual se quiere obtener la lista de vértices y aristas de la red.
	type	Toma valor de 1 o 2 e indica, si la matriz dada es una matriz Código vs Código o Archivo vs Código, respectivamente.
getCodesbyCodeCat: Esta función da una tabla con los códigos que están dentro de las categorías dadas.	x	Vector con los ids o nombres de las categorías de los códigos.
getFilesbyCases: Esta función da una tabla con los archivos que están dentro de los casos dados.	x	Vector con los ids o nombres de las casos.

Función	Argumentos	Uso
getFilesbyFileCat: Esta función da una tabla con los archivos que están dentro de las categorías dadas.	x	Vector con los ids o nombres de las categorías de los archivos.

Tabla 7. Funciones Generales de RQDAExt

5.7 Funciones internas

El paquete también incluye una serie de funciones internas que solamente se emplean para ser llamadas con las funciones antes citadas y que, por lo tanto, no podrán ser utilizadas por el usuario final. Tales funciones son:

- *gtkTreeViewSelectedIndices()*. Función utilizada para encontrar los índices de una vista de tabla cuando está se filtra.
- *gtkTreeViewClearColumns()*. Función utilizada para limpiar una vista de tabla, es decir, eliminar todos los componentes.
- *gtkTreeViewExportModel()*. Función que llama a un navegador de archivos que permite guardar el modelo de una vista de tabla en un formato .csv (Excel).
- *gtkTextViewExportBuffer()*. Función que llama a un navegador de archivos que permite guardar el texto de una vista de texto en formato.txt.
- *gtkExportNet()*. Función que llama a un navegador de archivos que permite guardar la red de un sistema de filtrado en formato .csv.
- *.onAttach*. Función que se ejecuta al adjuntar el paquete RQDAExt y que ejecuta la función *RQDAExt()*, que es la función que llama a la GUI.
- *.onUnLoad*. Función que se ejecuta al cerrar el paquete lanzando un mensaje de aviso en la consola de R.
- *CodeCodeWindow()*. Llama a la GUI del Code Vs Code Analysis.
- *CodeFileWindow().* Llama a la GUI Code Vs File Analysis.
- CodingsWindow(). Llama a la GUI del Coding Analysis.
- *SummaryCodingsWindow()*. Llama a la GUI del Summary of Codings Analysis.
- *AttributesWindow()*. Llama a la GUI del Attribute Analysis.
- *InstructionsWindow()*. Llama a la GUI de la ventana de ayuda (Figura 11).

5.8 Contenido del paquete

Para que el paquete pueda ser creado y subido a CRAN esté tenía que tener la serie de componentes que se indican en el apartado 3.3. Muchos de ellos son indispensables para que el paquete pueda ser construido y cargado (aparte de que tenían que ser comprobados mediante una instrucción para permitir su

subida al repositorio de CRAN). Los elementos forman la estructura de RQDAExt son:

- Carpeta /R. que es la carpeta que tenía que contener los archivos .R con los códigos fuente de lenguaje R. El código creado, puede ser visualizado en el *Anexo III*.
- Carpeta /man. Es donde se incluyen la descripción de las diferentes funciones creadas en archivos .Rmd. Éstos archivos tienen que tener un estructura muy concreta y escrita con lenguaje *LaTeX*, para que pueda en la construcción del paquete se generen los ficheros de ayuda en .html correctamente. Véase *Anexo II*.
- Carpeta /inst. Es donde se añade documentación del paquete de gran tamaño y que no es indispensable para la creación de éste. Se encuentran archivos como LICENCE o NEWS, así como, también, el icono utilizado para las ventanas y los archivos de las instrucciones de las diferentes ventanas. Véase *Anexo I*.
- Archivo DESCRIPTION. Es un archivo .txt que recoge los metadatos del paquete. Algunos como el título, el autor, el responsable de mantenimiento, la descripción o el tipo de licencia son obligatorios, pero hay otros campos que son meramente opcionales. Véase *Anexo I*.
- Archivo NAMESPACE. Es el archivo .txt que contiene los elementos del código que se exportarán, es decir, podrán ser utilizados una vez instalado el paquete; y los elementos que se importarán, es decir, los elementos de otros paquetes que será necesario instalar para que funcione el paquete. Véase *Anexo I*.

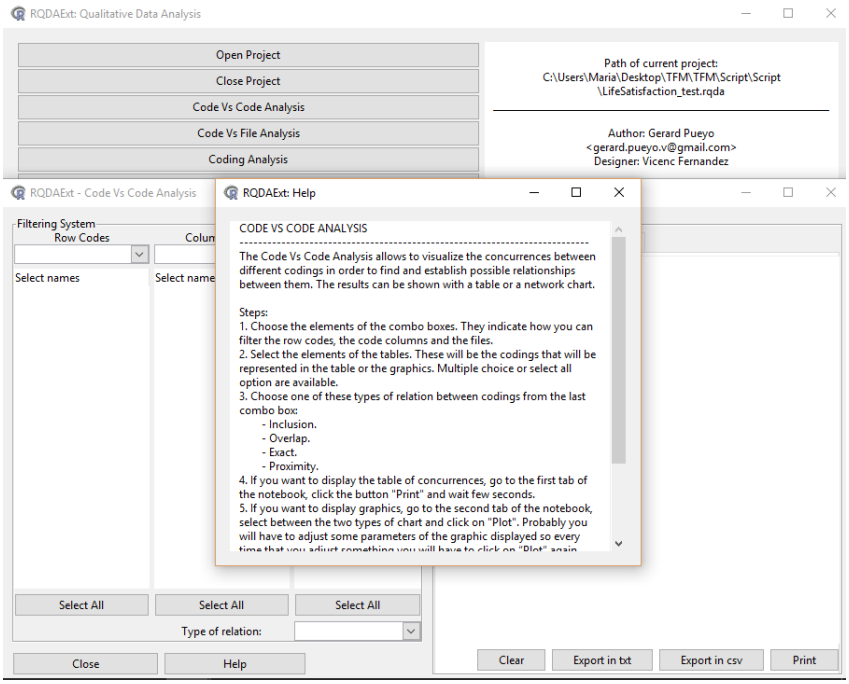


Figura 11. Caputura de la ventana de ayuda

6. Implicaciones ambientales

Este proyecto no contiene implicaciones medioambientales.

7. Análisis de viabilidad económica

Tal y como se ha comentado anteriormente, con este proyecto se ha pretendido ofrecer una alternativa gratuita para los investigadores que no quieren utilizar los programas que requieren caras licencias de uso. Es por eso que se ha tenido que argumentar si existen razones económicas que justifiquen la viabilidad del proyecto.

Puesto que el código creado es software libre y se ha colgado en un repositorio oficial para que todos aquellos que quieran se lo puedan descargar, no tiene mucho sentido estudiar si existirá un retorno de la inversión que haga el proyecto viable. Aun así, se ha planteado un estudio que consta de dos apartados para evaluar y demostrar los motivos que impulsan la realización del proyecto.

El primer apartado pretende hacer una comparativa entre el paquete objeto y el resto de programas del mercado que realizan la misma función. La finalidad es mirar si se ha conseguido alcanzar el objetivo de hacer que un investigador opte por un programa gratuito que, aunque sólo cubra las necesidades más comunes, le salga más a cuenta que uno que no lo sea.

El segundo apartado del estudio ha evaluado, mediante simulación, si el coste de realizar el proyecto pudiera otorgar una serie de ahorros económicos que impliquen una hipotética recuperación de la inversión.

7.1 Análisis comparativo entre programas

Se ha llevado un análisis comparativo entre los programas específicos de QDA y los paquetes RQDA y RQDAExt mediante una técnica de evaluación de alternativas, como lo es un VTP (valor técnico ponderado). Con esta técnica se evalúan una serie de opciones en base a unos criterios previamente ponderados.

Los programas evaluados son:

- MaxQDA.
- Atlas.ti.
- Nvivo9.
- OpenQDA.

Los criterios de evaluación empleados han sido los sido los que se muestran a continuación y han sido ponderados en base a lo que un investigador que realiza QDA busca en un programa de ésta índole.

- Precio (40%). Es el criterio que se ha evaluado en base al coste que tiene usar los diferentes programas. La gran mayoría de los programas requieren licencias en las cuales se pagan 1000€ anuales, y es una de las principales razones de la existencia de este proyecto, es por eso, que

se le ha dado una ponderación muy alta. Cuanto más económicos mayor calificación obtenían. En caso de ser libres (gratis) el programa obtenía la máxima nota.

- Flexibilidad con sistema Operativo (20%). Se pretende evaluar si los programas funcionan para diferentes sistemas operativos o si, por el contrario, sólo funcionan únicamente para uno. Se le ha asignado una ponderación considerable puesto que muchos investigadores no trabajan con Windows. Cuanto más grande es el abanico de sistemas operativos más nota tienen.
- Facilidad de uso en la codificación (15%). Es el criterio que evalúa la forma en la que el programa permite hacer la codificación del texto mediante el uso de códigos y la categorización de éstos. Cuanto más claridad y simplicidad a la hora de hacer la codificación más nota.
- Facilidad de uso en el análisis (15%). Es el criterio con el que se ha evaluado como es de claro hacer el análisis de los datos una vez están codificados. A mayor claridad y simplicidad más nota, pero también se busca la estética en presentación de los resultados (gráficos y tablas).
- Exportación de datos (10%). Se pretende evaluar la capacidad de exportar los datos obtenidos con el programa en las diferentes extensiones posibles. A mayor capacidad de exportación más nota.

Las calificaciones dadas a los programas en base a los diferentes criterios, así como la nota obtenida se muestra en la Tabla 8. Los rangos de las cualificaciones eran del 0 al 10.

	Precio	Sistema Operativo	Facilidad de uso codificación	Facilidad de uso de análisis	Exportación de datos	NOTA
RQDA	10	10	7	3	10	**8,5**
RQDA+	10	10	3	7	10	**8,5**
MaxQDA	7	7	9	9	8	7,7
Atlas.ti	7	7	8	9	8	7,55
Nvivo9	7	7	8	9	8	7,55
OpenQDA	10	3	7	7	8	7,5

Tabla 8. VTP para la comparativa de programas

Pese a que los programas especializados obtienen mayores calificaciones en cuanto al manejo del programa (codificación y análisis) no las obtienen para los otros criterios, puesto que son programas caros y la mayoría no funcionan para todos los sistemas operativos. Así pues, por el gran ahorro económico que

suponen y que se pueden usar en cualquier ordenador, tanto el paquete de R creado como el RQDA obtienen notas ligeramente superiores.

Con esta comparación vemos el gran atractivo que tiene el RQDAExt. Saldría a cuenta para muchos investigadores que buscarían menos funcionalidades, pero les supondría un ahorro de dinero considerable.

7.2 Avance económico

El avance económico se ha estudiado mediante simulación. Se ha buscado el ahorro en base al coste de trabajar únicamente con RQDA y al de trabajar con RQDA complementado con RQDAExt. Esa ganancia que encontremos será tratada como un flujo de caja para calcular parámetros de evaluación de inversiones como son el VAN (Valor Actual Neto) y el TIR (Tasa de Interés Rentable).

Imaginemos el siguiente escenario. Un investigador está haciendo un análisis QDA y ha acabado la parte correspondiente a la reducción de información mediante la codificación y categorización y quiere hacer la parte referente al análisis de esos datos. Si usará únicamente RQDA tendría que utilizar las funciones que le provee dicho paquete e iría llamándolas una a una en lo que sería un trabajo muy farragoso y muy monótono. Si por el contrario utilizará la interfaz que ofrece RQDAExt lo tendría mucho más sencillo porque de forma muy rápida puede ir llamando a las funciones seleccionando códigos o archivos pulsando botones. A parte, le permitiría visualizar y exportar de forma directa los resultados. Todo esto se traduce en que dicho investigador realizaría su análisis de datos en menos tiempo, lo que equivaldría a un ahorro de dinero también.

Estamos hablando por ejemplo de que, para un análisis estándar, en vez de tardar 80 horas se podrían tardar 30 horas menos. Si le añadimos el coste por hora del investigador de aproximadamente 30€/h podemos ver en la Tabla 9 el ahorro económico que supondría.

	Tiempo(h)	Coste(€)
Sin RQDAExt	50	1500
Con RQDAExt	80	2400
	Ahorro(€)	900

Tabla 9. Ahorro económico obtenido con RQDAExt

Estos 900€ corresponderían al dinero ahorrado por proyecto de investigación. A tenor de que un investigador dedica unos 3 meses por proyecto, lo que corresponde con 4 proyectos al año, estaríamos hablando de 3600€ anuales.

Si el coste que ha valido hacer el paquete RQDAExt, y que queda reflejado en el presupuesto, lo hubiera asumido todo el investigador, estaríamos hablando de que debería de recuperar 9.600€ para que le resultará viable.

Si calculamos el VAN para un horizonte de 4 años con un coste de oportunidad del 7% (equivalente a el coste que se tendría en dedicar el dinero en otro proyecto) se obtiene 2.593€. Un valor positivo lo que nos indica que la inversión hecha es buena.

$$VAN = \sum_{k}^{n} \frac{FC_k}{(1+i)^k} - I_0$$

En cuanto al TIR se obtiene un valor de 18% por lo que el proyecto da una rentabilidad superior a la tasa de rentabilidad mínima requerida.

$$0 = \sum_{k}^{n} \frac{FC_k}{(1+TIR)^k} - I_0$$

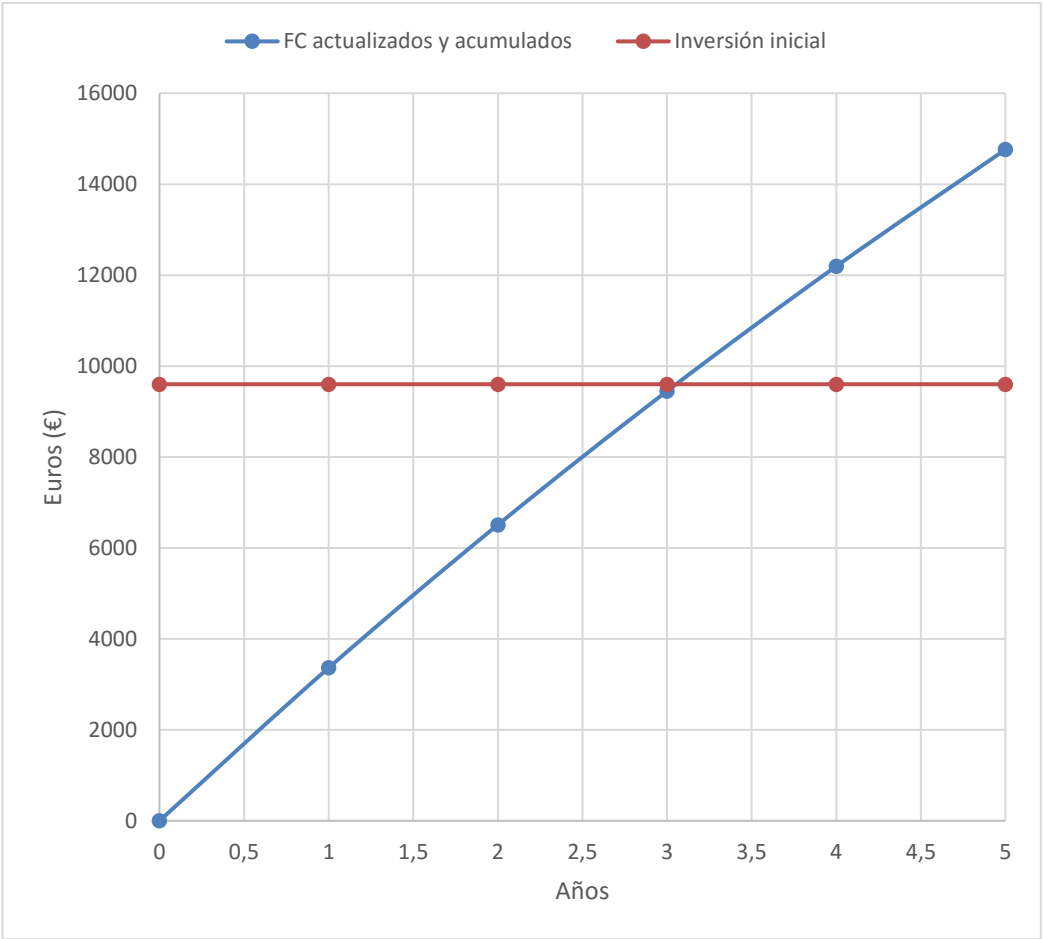


Figura 12. Cálculo de retorno de la inversión

De la Figura 11 vemos que, si un investigador pagará por desarrollar el paquete, le sería aconsejable y en 3 años se le habría retornado la inversión. Obtendría un ahorro de 2.593€ en 4 años considerando un interés de 7%. Como el programa está pensado para estar colgado en CRAN gratuitamente a disposición de todos en un futuro, el resto de investigadores obtendría un ahorro de 12.193€ en 4 años.

8. Planificación para futuras versiones

Una vez el paquete se ha finalizado el siguiente paso es subirlo a CRAN. Éste es un repositorio oficial donde puede ser compartido con toda la comunidad de R. Para que sea aceptado, debe de cumplir con las políticas del repositorio y con unos estándares establecidos. Sin embargo, no es algo directo, si no que son unos voluntarios los que en su tiempo libre revisan las solicitudes de paquetes en cola y comprueban si cumplen o no cumplen y en caso de que no cumplir te indican que es lo que hay que corregir. Aunque está fuera del alcance, el paquete RQDAExt será enviado a la web para intentar que esté a disposición de todos.

Una vez en CRAN, como todo elemento informático, es susceptible de ser mejorado con futuras actualizaciones. Con cada una de éstas, habrá un aumento en la versión del paquete. Ahora mismo tras su creación, RQDAExt se encuentra en la versión 0.1.

La persona podrá realizar dichas actualizaciones (en CRAN) es el responsable del mantenimiento, en éste caso, Gerard Pueyo, a no ser que se le pida permiso para cambiarlo. También, existe la posibilidad de que el paquete quede huérfano, no tenga responsable por falta de actividad de éste, y pueda ser actualizado por terceros.

Son por estos motivos que, en los apartados siguientes, se indicarán posibles mejoras que han quedado fuera del alcance del proyecto o que han quedado identificadas como funcionalidades poco comunes. Recordar que se ha pretendido (siguiendo lo que dice Pareto) resolver el 80% de las cosas que tiene que disponer un QDA con usando un 20% de los recursos.

8.1 Líneas futuras

Para este proyecto, posibles líneas futuras que se podrían implementar para mejorar el paquete y que pueden servir como base a futuros proyectos serían:

- Visualizador de los Memos de todos los elementos: códigos, archivos, categorías, casos, atributos, etc.
- Conseguir los archivos o casos donde existan las codificaciones dadas.
- Añadir nuevos tipos de diagramas de red como *arc diagram*, *hive diagram* o *biofabric diagram*.
- Añadir nuevos tipos de gráficos estadísticos para el recuento de códigos.
- Mejorar el tema de exportación de gráficos.
- Intentar implementar las ventanas con los diferentes tipos de análisis en una sola en vez de que se vayan abriendo.
- Añadir barras de menú, pop-ups, barra de status…
- Corrección de errores.

8.2 Identificación de tareas

Para llevar a cabo nuevas las actualizaciones descritas en el apartado anterior, se han identificado que se deberán llevar a cabo las tareas de la Tabla 10. A cada tarea se le hecho una estimación del tiempo que podrían durar en base a lo tardado con éste y se le ha identificado las relaciones de dependencia entre cada una. Si se le dedican aproximadamente 4-5horas al día, durante 5 días a la semana, podemos dejar expresado las horas en semanas de duración.

Tarea	Tareas	Horas	Semanas	Dependencia
1	Entender el funcionamiento y la programación de RQDAExt y aprender a manejar RGtk2.	80	4	-
2	Diseño de los elementos y las funciones.	20	1	-
3	Programar el análisis de memos	20	1	1,2
4	Programar la obtención de casos y archivos por codificación.	20	1	1,2
5	Añadir los nuevos diagramas de red.	60	3	1,2
6	Añadir nuevos gráficos estadísticos.	60	3	1,2
7	Implementar mejoras generales: barras de ayuda, pop-ups…	40	2	1,2
8	Depurar errores y testear paquete.	20	1	3,4,5,6,7
9	Hacer la documentación de ayuda.	20	1	3,4,5,6,7

Tabla 10. Tareas a llevar a cabo para futuras versiones

Dichas tareas están pensadas para una persona que no sabe cómo está programada RQDAExt, por lo que se le ha añadido la tarea 1.

8.3 Gantt del proceso

El Gantt del proceso de las tareas identificadas en la Tabla 10. Se mostraría en la Figura 12. Cómo se puede ver la duración total, según nuestras aproximaciones, es de 340 horas (17 semanas aproximadamente).

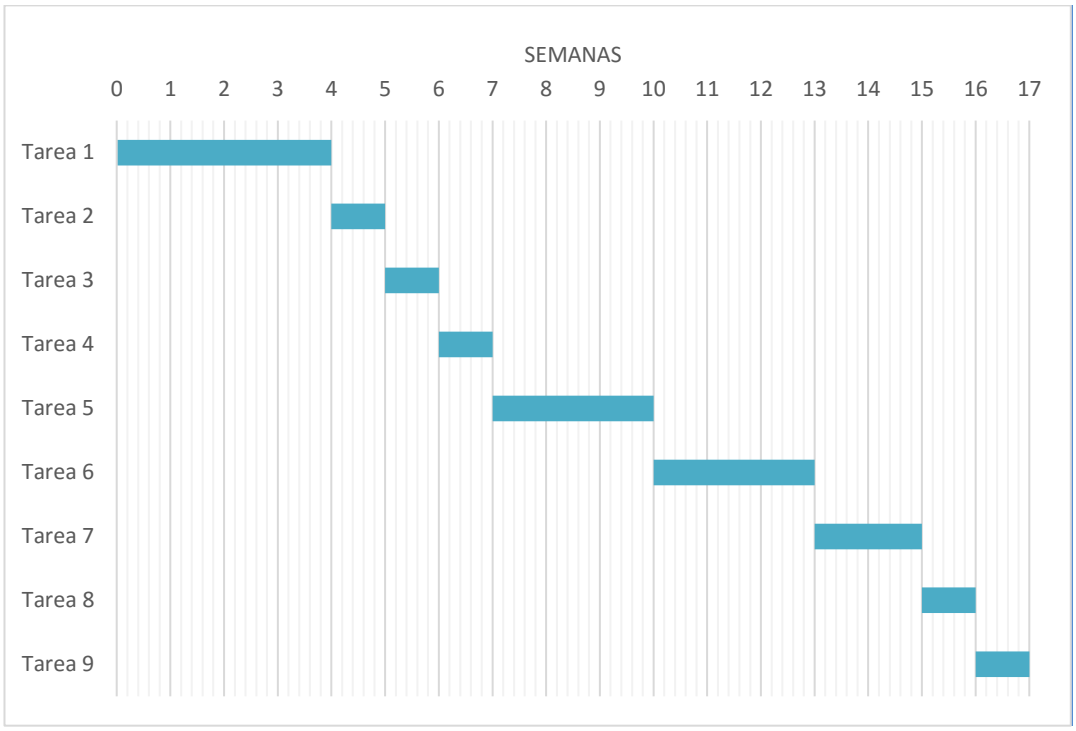


Figura 13. Gantt del proceso para futuras versiones

9. Conclusiones

El objeto de este proyecto consistía en el desarrollo de un paquete de R que contuviese una interfaz gráfica de usuario para complementar a la del paquete RQDA (*R-based Qualitative Data Analysis*) de Hang Rounggi (2014), ampliando y extendiendo sus funcionalidades dentro del campo del análisis cualitativo de datos.

El resultado al que se ha llegado ha sido el paquete RQDAExt (*R-based Qualitative Data Analysis Extension*) que extiende y complementa las opciones de RQDA. Éste último se focalizaba sólo en la parte de reducción de la información mediante el codificado del texto, por lo que RQDAExt buscaba llenar el vacío referente al análisis de esa información codificada.

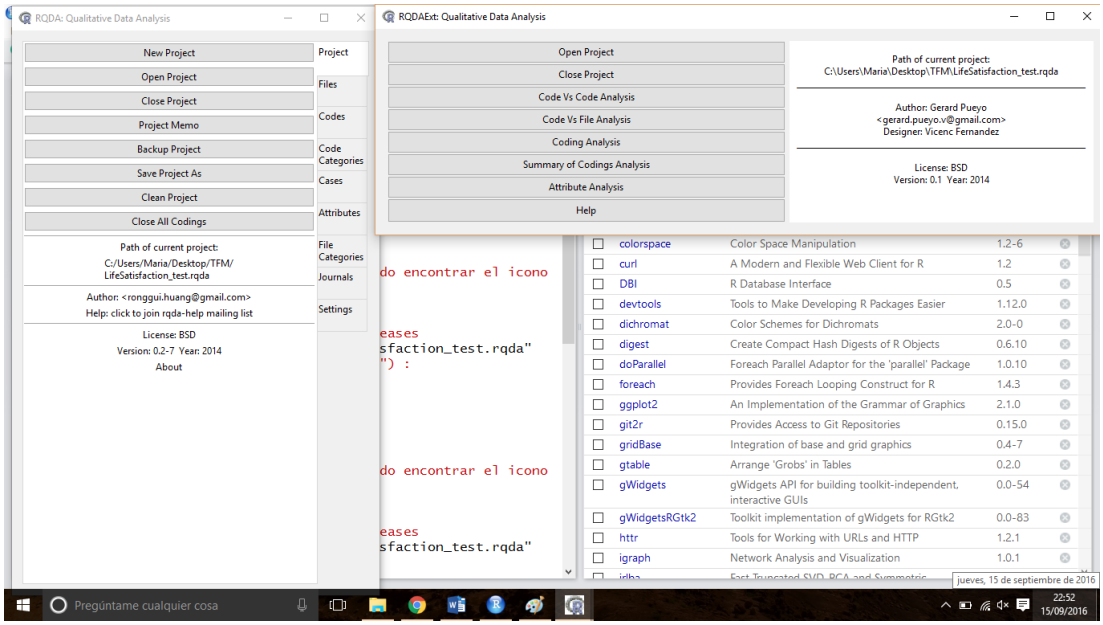


Figura 14. Captura con las GUIs de RQDA y RQDAExt

Puesto que existen grandes cantidad cosas que se pueden hacer dentro del análisis cualitativos de datos, una de las primeras tareas que abarcaba el alcance era la identificación de las funcionalidades que debería el paquete. Debían der ser aquellas más útiles y que más frecuentemente usan mayoría de investigadores. Se ha concluido que éstas eran debían de ser las herramientas:

- Análisis de las concurrencias entre códigos mediante tabla Código vs Código y mediante diagramas de red.
- Análisis de las concurrencias entre archivos y códigos mediante tabla Código vs Archivo y mediante diagramas de red.
- Análisis de las codificaciones mediante la consulta de éstas.

- Análisis de las frecuencias de las codificaciones mediante tablas y gráficos.
- Análisis de los archivos mediante sus atributos.

El resto de pasos del alcance se completado satisfactoriamente: el diseño, la programación, la redacción del manual, hasta, finalmente, la creación del paquete en .tar. Este archivo sigue la estructura fijada por el repositorio CRAN, con la intención de que pueda ser subido en un futuro. Allí, se trataría como código libre y abierto por lo que, todo usuario que lo deseara, pudiera copiarlo, estudiarlo, modificarlo, utilizarlo libremente con cualquier fin y redistribuido con o sin cambios o mejoras, siempre que se siga su licencia, que en este caso es una BSD de 3 cláusulas.

Se ha visto también como, comparándolo con sus análogos de pago mediante un VTP, RQDAExt tenía más nota gracias a que cualquiera puede usarlo gratuitamente y gracias a su versatilidad (funciona para todos los sistemas operativos). Analizando el VAN y el TIR, se ha conseguido estudiar que la totalidad del coste del proyecto, 9.600€ (véase el documento *Presupuesto*), se podría recuperar en 3 años gracias al ahorro económico que se obtiene respecto al no usarlo. Un aliciente que ha servido para dar peso a la justificación del proyecto.

Finalmente, también se ha hecho un estudio sobre lo que deberían de tener futuras versiones del paquete para mejorarlo.

10. Recomendaciones de continuidad

La continuación de éste proyecto es seguir todos los pasos necesarios para compartir el paquete con el resto de investigadores a través la plataforma oficial CRAN. Esto conlleva a tener un feedback con los voluntarios hasta que todo está correcto para ser alojado allí.

Al subir RQDAExt a CRAN, el responsable del mantenimiento se hace cargo de ir actualizándolo y comprobando que funcione con las futuras versiones de R. De no ser así, el paquete será archivado. También, tendrá que ir respondiendo todos los mails que el equipo de CRAN envíe. En caso contrario, el paquete será considerado como huérfano (no tendrá responsable de mantenimiento oficial) y será archivado en el momento que no funcione con la última versión de R.

Por ese motivo, la única recomendación que se da a las personas o entidades que quieran hacerse responsables para dar continuidad al paquete (aplicando mejoras como las nombradas en el apartado 8.1) es que procuren tener el paquete al día y hagan todo lo posible para que el paquete siga estando en el repositorio.

Es de señalar que únicamente es el responsable el que puede de actualizar el paquete y que si algún otro desea hacerlo debe de ponerse en contacto con él. Yo, Gerard Pueyo, actual responsable del paquete, sólo ofreceré mi renuncia y cederé los derechos a aquel que cumpla con la recomendación anterior y que se comprometa a seguí haciendo el trabajo más cómodo y fácil a los investigadores que usen RQDAExt.

11. Bibliografía

1. **Ronggui, Huan.** RQDA: R-based Qualitative Data Analysis. R package version 0.2-7, 2014. http://rqda.r-forge.r-project.org/.

2. **Paradis, Emmanuel.** *R para principiantes.* Montpellier : Universit Montpellier, 2003.

3. **Leisch, Friedrich.** *Creating R Packages: A Tutorial.* Munich : R Development Core Team, 2009.

4. **W.N. Venables, D.M. Smith.** *An Introduction to R.* s.l. : R Development Core Team, 2015.

5. **Patricia Schettini, Inés Cortazzo.** *Análisis de datos cualitativos en la investigación social.* Buenos Aires : Editorial Universidad de la Plata, 2015. ISBN 978-950-34-1231-2.

6. **Fernandez, Vicenç.** *Summer School: Qualitative Methods for Research in R.* Terrassa : s.n., 2015.

7. **Michael F. Lawrence, John Verzani.** *Programing Graphical User Interfaces in R.* New York : CRC Press. Taylor and Francis Group., 2012. ISBN 978-1-4398-5682-6.

8. **Team, R Core.** *Writing R Extensions.* s.l. : R Development Core Team, 2015.

9. **Hernández Pina, F., García-Sanz, M.P. y Maquilón, J.J.** *Análisis de datos cualitativos.* 2014.

10. **Fonditos3D.** [En línea] [Citado el: 4 de Setiembre de 2016.] http://www.fonditos3d.com/imagenes-rotuladores-de-colores-3d-jpg-1920x1080.