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 4.- ANEXOS

Índice de Contenidos

ANEXO I – Metadatos del paquete

- ## DESCRIPTION file

Package: RQDAExt
Type: Package
Title: R-Based Qualitative Data Analysis Extension
Version: 0.1
Date: 2016-07-21
Author: Gerard Pueyo [aut, cre]
  Vicenc Fernandez [ctb]
Maintainer: Gerard Pueyo <gerard.pueyo.v@gmail.com>
Description: R package for Qualitative Data Analysis that complements and
  extends the package 'RQDA' with new options for the analysis and graphics
  through a Graphical User Interface. It has been designed with the purpose of
  being used with the GUI provided by 'RQDA'.
License: BSD_3_clause + file LICENSE
Depends:
  R(>= 3.2.5),
  RGtk2(>= 2.20.31),
  RQDA(>= 0.2-7)
Imports:
  RSQLite(>= 1.0.0),
  DBI,
  gWidgetsRGtk2(>= 0.0.36),
  gWidgets(>= 0.0-31),
  plotrix(>= 3.6-2),
  wordcloud(>= 2.5),
  igraph(>= 1.0.1)

- ## NAMESPACE file

```
export(RQDAExt, getCodesbyCodeCat, getFilesbyCases, getFilesbyFileCat,
    CodeCodeMatrix, CodeFileMatrix, getFilesbyAttribute, getAttributesbyFile,
    getCodings, getLinksAndNodes,SummaryOfCodings)
import(RGtk2)
import(RQDA)
import(gWidgets)
import(gWidgetsRGtk2)
import(DBI)
import(RSQLite)
import(igraph)
importFrom("plotrix", pie3D, pie3D.labels)
importFrom("wordcloud", wordcloud)
```

ANEXO II – Archivos de ayuda

Este anexo contiene los archivos .Rd de la carpeta \man correspondientes a lo que serian los manuales de ayuda de todas las funciones.

- ## CodeCodeMatrix.Rd

```
\name{CodeCodeMatrix}
\alias{CodeCodeMatrix}
\title{
Code Vs Code Matrix
}
\description{
Return a matrix which give the concurrences between codes from a filtering system.
}
\usage{
CodeCodeMatrix(filter, rowcid, colcid, fid,
  relation=c("overlap","inclusion","exact","proximity"),
  as.matrix = TRUE)
}
\arguments{
  \item{filter}{
a three-dimensional vector indicating how filter the codes of the rows, the codes of the columns and the files respectively. "Codes" or "Code Categories" for the codes and "Files", "File Categories" or "Cases" for the files.
}
  \item{rowcid}{
an integer vector of code/code category ids for the rows.
}
  \item{colcid}{
an integer vector of code/code category ids for the columns.
}
  \item{fid}{
an integer vector of file/file category/case ids.
}
  \item{relation}{
the relation between codes.
}
  \item{as.matrix}{
logical.
}
}
\details{
It provides the table of concurrences between the codes in a matrix form. In case as.matrix=FALSE the table will be given in a data frame form. This last option is thought basically to be printed in a tree view widget so it is not advisable change the value to TRUE.

The first and the second value of the filter are used to indicate whether ids specified by rowcid and colcid respectively are referred to "Codes" or "Codes Categories" and the third value whether ids specified by fid are referred to "Files", "Cases" and "File Categories".

}
\value{
A matrix if as.matrix=TRUE and a data frame if as.matrix=FALSE.
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{CodeFileMatrix}},
\code{\link[RQDA]{relation}}
}
```

```
\examples{
\dontrun{
CodeCodeMatrix(filter=c("Codes","Codes","Files"),rowcid=c(1:5), colcid=c(1:5),
   fid=c(1:10),relation="inclusion",as.matrix=TRUE)
CodeCodeMatrix(filter=c("Code Categories","Code Categories","Cases"),
   rowcid=c(2), colcid=c(2),fid=c(1),relation="proximity",as.matrix=TRUE)
CodeCodeMatrix(filter=c("Codes","Codes","File Categories"),rowcid=c(1,2,3),
  colcid=c(1,2,3),fid=c(2),relation="exact",as.matrix=TRUE)
}
}
```

- **CodeFileMatrix.Rd**

```
\name{CodeFileMatrix}
\alias{CodeFileMatrix}
\title{
Code Vs File Matrix
}
\description{
Return a matrix which give the concurrences between codes and files from a filtering system.
}
\usage{
CodeFileMatrix(filter, cid, fid, as.matrix = TRUE)
}
\arguments{
  \item{filter}{
a two-dimensional vector indicating how filter the codes and files respectively. "Codes" or "Code Categories" for the codes and "Files", "File Categories" or "Cases" for the files.
}
  \item{cid}{
an integer vector of code/code category ids.
}
  \item{fid}{
an integer vector of file/file category/case ids.
}
  \item{as.matrix}{
logical
}
}
\details{
It provides the table of concurrences between the codes in matrix form. In case as.matrix=FALSE the table will be given in a data frame form. This last option is thought basically to be printed in a tree view widget so it is not advisable change the value to TRUE.

The first value of the filter is used to indicate whether ids specified by cid are referred to "Codes" or "Codes Categories" and the second value whether ids specified by fid are referred to "Files", "Cases" and "File Categories".
}
\value{
A matrix if as.matrix=TRUE and a data frame if as.matrix=FALSE.
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{CodeCodeMatrix}}
}
\examples{
\dontrun{
CodeFileMatrix(filter=c("Codes","Files"),cid=c(1:5),fid=c(1:10),as.matrix=TRUE)
CodeFileMatrix(filter=c("Code Categories","File Categories"),cid=c(1,2),
   fid=c(1),as.matrix=TRUE)
CodeFileMatrix(filter=c("Codes","Cases"),cid=c(1:5,10:15),fid=c(2),
```

```
   as.matrix=TRUE)
}
}
```

- **getAttributesbyFile.Rd**

```
\name{getAttributesbyFile}
\alias{getAttributesbyFile}
\title{
Get the attributes of a file or a case
}
\description{
Get the attributes of a file or a case
}
\usage{
getAttributesbyFile(filter=c("Files","Cases"), fid)
}
\arguments{
  \item{filter}{
Any one of "Files" and "Cases".
}
  \item{fid}{
an integer of file/case id.
}
}
\details{
It provides the attributes of the file/case specified by fid and the values of all of these attributes.

The argument filter is used to indicate whether the id of the fid is a file or a case.
}
\value{
A data frame of attributes.
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{getFilesbyAttribute}}
}
\examples{
\dontrun{
getFilesbyAttribute(filter="Files", fid=2)
getFilesbyAttribute(filter="Cases", fid=4)
}
}
```

- **getCodesbyCodeCat.Rd**

```
\name{getCodesbyCodeCat}
\alias{getCodesbyCodeCat}
\title{
Get the codes of code categories list
}
\description{
Get the codes of code categories list.
}
\usage{
getCodesbyCodeCat(x)
}
\arguments{
\item{x}{an integer vector of code categoriy ids or a character vector of code category names.
}
}
```

\details{
It provides the codes with their corresponding ids and names from a vector of code categories.

The vector of code categories can be given by ids or names, but not a combination of both.
}
\value{
A data frame of codes.
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{getFilesbyCases}},
\code{\link{getFilesbyFileCat}}
}
\examples{
\dontrun{
getCodesbyCodeCat(x=c(1,2,3))
getCodesbyCodeCat(x=c("Interview Topics","Day-to-day issues",
"People"))
}
}

- **getCodings.Rd**

\name{getCodings}
\alias{getCodings}
\title{
Get the codings of a filtering system
}
\description{
Get the codings of a filtering system.
}
\usage{
getCodings(filter, cid, fid, condition=c("AND","OR"))
}
\arguments{
  \item{filter}{
a two-dimensional vector indicating how filter the codes and files respectively. "Codes" or "Code Categories" for the codes and "Files", "File Categories" or "Cases" for the files.
}
  \item{cid}{
an integer vector of code/code category ids.
}
  \item{fid}{
an integer vector of file/file category/case ids.
}
  \item{condition}{
logical conjunction between codes.
}
}
\details{
It privides the codings of the codes/code categories specified by cid whose relationship between them is indicated with condition and whose files files/cases/file categories are specified by fid.

The first value of the filter is used to indicate whether ids specified by cid are referred to "Codes" or "Codes Categories" and the second value whether ids specified by fid are referred to "Files", "Cases" and "File Categories".

If only one code/code category is provided, the value of condition is indifferent, either "AND" or "OR".
}
\value{
a data frame with additional class of "codingsByOne"

```
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{SummaryOfCodings}}
\code{\link[RQDA]{getCodingsByOne}}
}
\examples{
\dontrun{
getCodings(filter=c("Codes","Files"),cid=c(1,2,3), fid=c(1:10),condition="AND")
getCodings(filter=c("Code Categories","Cases"), cid=c(3,4), fid=c(1,2),condition="OR")
}
}
```

- **getFilesbyAttribute.Rd**

```
\name{getFilesbyAttribute}
\alias{getFilesbyAttribute}
\title{
Get the files or the cases of an attribute
}
\description{
Get the files or the cases sorted by the values of an attribute
}
\usage{
getFilesbyAttribute(filter=c("Files", "Cases"), attribute)
}
\arguments{
  \item{filter}{
Any one of "Files" and "Cases".
}
  \item{attribute}{
name of the attribute.
}
}
\details{
It provides a classification of files/cases according to the value that they have with the specified attribute.

The argument filter is used to indicate whether it will sort files or cases.
}
\value{
A list of data frames. One for each value of the attribute.
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{getAttributesbyFile}}
}
\examples{
\dontrun{
getFilesbyAttribute(filter="Files", attribute="Gender")
getFilesbyAttribute(filter="Files", attribute="Age")
getFilesbyAttribute(filter="Cases", attribute="Education")
}
}
```

- **getFilesbyCases.Rd**

```
\name{getFilesbyCases}
\alias{getFilesbyCases}
```

```
\title{
Get the files of cases list
}
\description{
Get the files of cases list.
}
\usage{
getFilesbyCases(x)
}
\arguments{
\item{x}{
an integer vector of case ids or a character vector of case names.
}
}
\details{
It provides the files with their corresponding ids and names from a vector of cases.

The vector of cases can be given by ids or names, but not a combination of both.
}
\value{
A data frame of files.
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{getFilesbyFileCat}},
\code{\link{getCodesbyCodeCat}}
}
\examples{
\dontrun{
getFilesbyCases(x=c(1,2,3))
getFilesbyCases(x=c("New York","Indiana", "Illinois"))
}
}
```

- **getFilesbyFileCat.Rd**

```
\name{getFilesbyFileCat}
\alias{getFilesbyFileCat}
\title{
Get the files of file category list
}
\description{
Get the files of file category list.
}
\usage{
getFilesbyFileCat(x)
}
\arguments{
\item{x}{
an integer vector of file category IDs or a character vector of file category names.
}
}
\details{
It provides the files with their corresponding ids and names from a vector of file categories.

The vector of file categories can be given by ids or names, but not a combination of both.
}
\value{
A a data frame of files.
}
\author{
```

```
Gerard Pueyo
}
\seealso{
\code{\link{getFilesbyCases}},
\code{\link{getCodesbyCodeCat}}
}
\examples{
\dontrun{
getFilesbyFileCat(x=c(1,2,3))
getFilesbyFileCat(x=c("Interviews","News", "Webs"))
}
}
```

- **getLinksAndNodes.Rd**

```
\name{getLinksAndNodes}
\alias{getLinksAndNodes}
\title{
Get the links and the nodes from a matrix
}
\description{
Get the links and the nodes from a matrix
}
\usage{
getLinksAndNodes(matrix, type)
}
\arguments{
  \item{matrix}{
a matrix.
}
  \item{type}{
1 if the the matrix is a Code Vs Code Matrix and 2 if the Matrix is a Code Vs File Matrix.
}
}
\details{
It provides the nodes and the links for a network chart using a Code Vs Code Matrix or a Code Vs File Matrix.

For the first case, nodes are referred to codes. They include the code categories of the codes with the ids, too. Links contains information about the origin and the destination of the different connection between codes with the weight of them.

For the second case, nodes are referred to codes and files and include the code categories, cases, and file categories of them. Links show the connections between codes and files indicating their weight.

This functions is designed with the purpose of plot a network chart with the 'igraph' package.
}
\value{
A list.
\item{nodes}{the nodes of the matrix.}
\item{links}{the links between the nodes.}
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{CodeFileMatrix}},
\code{\link{CodeCodeMatrix}}
}
\examples{
\dontrun{
matrix <- CodeCodeMatrix(filter=c("Codes","Codes","Files"),rowcid=c(1:5),
```

```
                colcid=c(1:5),fid=c(1:10),relation="inclusion",
                as.matrix=TRUE)
getLinksAndNodes(matrix, type=1)
}
}
```

- **RQDAExt.Rd**

```
\name{RQDAExt}
\alias{RQDAExt}
\title{
RQDAExt Graphical User Interface
}
\description{
R-based Qualitative Data Analysis extension GUI for package 'RQDA'.
}
\usage{
RQDAExt()
}
\value{
A function.
}
\author{
Gerard Pueyo
}
```

- **RQDAExt-package.Rd**

```
\name{RQDAExt-package}
\alias{RQDAExt-package}
\alias{RQDAExt}
\docType{package}
\title{
\packageTitle{RQDAExt}
}
\description{
\packageDescription{RQDAExt}
}
\author{
\packageAuthor{RQDAExt}

Maintainer: \packageMaintainer{RQDAExt}
}
\keyword{ package }
\examples{
\dontrun{
library(RQDAExt)
RQDAExt()
}
}
```

- **SummaryOfCodings.Rd**

```
\name{SummaryOfCodings}
\alias{SummaryOfCodings}
\title{
Summary of Codings using a filtering system.
}
\description{
Summary of Codings using a filtering system.
}
\usage{
SummaryOfCodings(filter, cid, fid, variable=c("Number of Codings",
```

```
    "Average Length","Number of Files","Codings for Each File"))
}
\arguments{
  \item{filter}{
a two-dimensional vector indicating how filter the codes and files respectively. "Codes" or "Code Categories" for the codes and "Files", "File Categories" or "Cases" for the files.
}
  \item{cid}{
an integer vector of code/code category ids.
}
  \item{fid}{
an integer vector of file/file category/case ids.
}
  \item{variable}{
type of summary.
}
}
\details{
It provides the summary of the codings of the codes/code categories specified by cid whose files/cases/file categories are specified by fid.

The first value of the filter is used to indicate whether ids specified by cid are referred to "Codes" or "Codes Categories" and the tsecond value whether ids specified by fid are referred to "Files", "Cases" and "File Categories".
}
\value{
A data frame with the values.
}
\author{
Gerard Pueyo
}
\seealso{
\code{\link{getCodings}},
\code{\link[RQDA]{summaryCodings}}
}
\examples{
\dontrun{
SummaryOfCodings(filter=c("Codes","Files"), cid=c(1:3), fid=c(1:5),
                 variable="Number of Files")
SummaryOfCodings(filter=c("Codes","Cases"), cid=c(1:3), fid=c(1,3),
                 variable="Average Length")
SummaryOfCodings(filter=c("Code Categories","Files"), cid=c(2), fid=c(1:5),
                 variable="Number of Codings")
SummaryOfCodings(filter=c("Codes","File Categories"), cid=c(1:3), fid=c(1),
                 variable="Codings for Each File")
}
}
```

ANEXO III – Código fuente

Este anexo contiene los archivos .R de la carpeta \R, correspondiente a lo que sería el código fuente del programa que genera la GUI y las funciones.

- **AttributesWindow**

```
###----------ATTRIBUTE ANALYSIS----------###
AttributesWindow <- function(){
  ###----------DATA TABLES----------###
  sql <- RQDAQuery("select id,name from source where status=1")
  tableFiles <- data.frame(id = sql[[1]], name = sql[[2]],
                           type= rep("Files", nrow(sql)),
                           visible = rep(FALSE, nrow(sql)))
  sql <- RQDAQuery("select id,name from cases where status=1")
  tableCases <- data.frame(id = sql[[1]], name = sql[[2]],
                           type= rep("Cases", nrow(sql)),
                           visible = rep(FALSE, nrow(sql)))
  sqlAttributes <- RQDAQuery("select name from attributes where status=1")
  DFs <- list()
  DFs$FilesCases <- rbind(tableFiles, tableCases)
  ###----------WINDOW
  aa <- gtkWindow(show=FALSE)
  aa$setTitle("RQDAExt - Attribute Analysis")
  aa$setDefaultSize(gdkScreenWidth()*2/3,gdkScreenHeight()*2/3)
  aa["border-width"] <- 10
  icon <- system.file("icon", "mainIcon.png", package = "RQDAExt")
  image <- gdkPixbufNewFromFile(icon)
  aa$setIcon(image$retval)
  ###----------COMPONENTS
  aa.boxes <- list()
  aa.boxes$filescases <- gtkComboBoxNewText()
  aa.boxes$attr <- gtkComboBoxNewText()
  sapply(c("Files","Cases"), aa.boxes$files$appendText)
  sapply(c("Files","Cases"), aa.boxes$attr$appendText)
  #
  aa.valuelabel <- gtkLabel()
  aa.attrlabel <- gtkLabel()
  #
  aa.general<-list()
  aa.general$close <- gtkButton("Close")
  aa.general$help <- gtkButton("Help")
  #
  aa.print <- gtkButton("Print")
  #
  aa.labels <- sqlAttributes$name
  aa.radiogp <- list()
  aa.radiogp[[aa.labels[1]]] <- gtkRadioButton(label=aa.labels[1])
  for (label in aa.labels[-1]){
    aa.radiogp[[label]] <- gtkRadioButton(aa.radiogp,label=label)
  }
  #
  aa.text <-list()
  aa.text$clear <- gtkButton("Clear")
  aa.text$txt <- gtkButton("Export in txt")
  aa.text$print <- gtkButton("Print")
  #
  aa.view <- gtkTextView()
  aa.scroll <- gtkScrolledWindow()
  aa.scroll$setPolicy("never","automatic")
  aa.scroll$add(aa.view)
  aa.buffer <- aa.view$getBuffer()
```

```
 aa.view["editable"] <- FALSE
 aa.view["wrap-mode"] <- "word-char"
 aa.view["left-margin"] <- 5
 aa.view["right-margin"] <- 5
 gtkWidgetSetSizeRequest(aa.view, 100, 300)
 aa.buffer$createTag(tag.name = "valuename", weight = PangoWeight["bold"])
 #
 aa.view2<- gtkTreeView()
 aa.scroll2 <- gtkScrolledWindow()
 aa.scroll2$setPolicy("automatic", "automatic")
 aa.scroll2$add(aa.view2)
 gtkWidgetSetSizeRequest(aa.view2, 100, 300)
 #
 aa.modelFilesCases<- rGtkDataFrame(DFs$FilesCases)
 #
 aa.filteredFilesCases <- aa.modelFilesCases$filter()
 aa.filteredFilesCases$setVisibleColumn(ncol(DFs$FilesCases)-1)
 #
 aa.viewFilesCases <- gtkTreeView(aa.filteredFilesCases)
 aa.viewFilesCases$insertColumnWithAttributes (0, "Select names", gtkCellRendererText(), text =
1)
 #
 aa.scrollFilesCases <- gtkScrolledWindow()
 aa.scrollFilesCases$add(aa.viewFilesCases)
 aa.scrollFilesCases$setPolicy("automatic", "automatic")
 ###----------CONTAINERS
 aa.container <- gtkHBox(TRUE, 10)
 aa.frame1 <- gtkFrame ("Get Attributes by Files/Cases")
 aa.frame2 <- gtkFrame ("Sort Files/Cases by Attributes")
 aa.vbox1 <- gtkVBox(FALSE, 10)
 #
 aa.grid1 <- gtkTable(rows = 4, columns = 2, homogeneous = FALSE)
 aa.hbox1 <- gtkHBox(FALSE, 5)
 #
 aa.vbox2 <- gtkVBox(FALSE, 10)
 aa.grid2 <- gtkTable(rows = 3, columns = 2, homogeneous = FALSE)
 aa.hbox2 <- gtkHBox(FALSE, 5)
 ###----------LAYOUT
 aa$add(aa.container)
 aa.container$packStart(aa.vbox1)
 aa.container$packStart(aa.frame2)
 #
 aa.vbox1$packStart(aa.frame1, expand = TRUE, fill = TRUE)
 aa.vbox1$packStart(aa.hbox1, expand=FALSE, fill = FALSE)
 #
 aa.frame1$add(aa.grid1)
 aa.grid1$attach(label <- gtkLabel("Filtering:"), left.attach =0,1 , top.attach = 0,1,
              xoptions = c("expand", "fill"), yoptions = "")
 label["xalign"] <- 1
 aa.grid1$attach(aa.boxes$filescases, left.attach =1,2 , top.attach = 0,1,
              xoptions = c("expand", "fill"), yoptions = "")
 aa.grid1$attach(label <- gtkLabel("Selected Value:"), left.attach =0,1 , top.attach = 1,2,
              xoptions = c("expand", "fill"), yoptions = "")
 label["xalign"] <- 1
 aa.grid1$attach(aa.valuelabel, left.attach =1,2 , top.attach = 1,2,
              xoptions = "", yoptions = "")
 aa.grid1$attach(aa.scrollFilesCases, left.attach =0,1 , top.attach = 2,3,
              xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
 aa.grid1$attach(aa.scroll2, left.attach = 1,2, top.attach = 2,3,
              xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
 aa.grid1$attach(gtkLabel(""), left.attach = 0,1, top.attach =3,4,
              xoptions = c("expand", "fill"), yoptions = "")
 aa.grid1$attach(gtkLabel(""), left.attach = 1,2, top.attach = 3,4,
```

```
            xoptions = c("expand", "fill"), yoptions = "")
aa.grid1$attach(aa.print, left.attach = 1,2, top.attach = 3,4,
            xoptions = c("expand", "fill"), yoptions = "")
#
aa.grid1$setRowSpacing(0,20)
aa.grid1$setRowSpacing(1,20)
aa.grid1$setRowSpacing(2,20)
aa.grid1$setColSpacing(0,10)
#
sapply(aa.general, aa.hbox1$packStart, expand = TRUE)
aa.hbox1$packStart(gtkLabel(""),expand=TRUE, fill = TRUE)
#
aa.frame2$add(aa.vbox2)
aa.vbox2$packStart(aa.grid2, expand = TRUE, fill = TRUE)
aa.vbox2$packStart(aa.hbox2, expand=FALSE, fill = FALSE)
#
aa.grid2$attach(label <- gtkLabel("Filtering:"), left.attach =0,1 , top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
label["xalign"] <- 1
aa.grid2$attach(aa.boxes$attr, left.attach =1,2 , top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
aa.grid2$attach(label <- gtkLabel("Selected Attribute:"), left.attach =0,1 , top.attach = 1,2,
            xoptions = c("expand", "fill"), yoptions = "")
label["xalign"] <- 1
aa.grid2$attach(aa.attrlabel, left.attach =1,2 , top.attach = 1,2,
            xoptions = "", yoptions = "")
aa.grid2$attach(aa.frame3 <- gtkFrame("Attributes"), left.attach =0,1 , top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
aa.radiobox <- gtkVBox(FALSE, 10)
aa.frame3$add(aa.radiobox)
aa.grid2$attach(aa.scroll, left.attach = 1,2, top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
aa.grid2$setRowSpacing(0,20)
aa.grid2$setRowSpacing(1,20)
aa.grid2$setRowSpacing(2,20)
aa.grid2$setColSpacing(0,10)
sapply(aa.radiogp, gtkBoxPackStart, object=aa.radiobox)
#
aa.hbox2$packStart(gtkLabel(""), expand = TRUE, fill = TRUE)
sapply(aa.text, aa.hbox2$packStart, expand = TRUE)
###----------SIGNAL CONNECTIONS
gSignalConnect(aa.boxes$filescases, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF[,"type"]
  DF[,"visible"] <- grepl(pattern, values)
}, data= aa.filteredFilesCases)
gSignalConnect(aa.general$close, "clicked", function(button){
  aa$destroy()
})
gSignalConnect(aa.general$help, "clicked", function(button){
  InstructionsWindow(active=6)
})
gSignalConnect(aa.print, "clicked", function(button, user.data){
  type <- aa.boxes$filescases$getActiveText()
  ind <- aa.viewFilesCases$selectedIndices()
  value <- aa.modelFilesCases[ind, 1]
  if(is.null(type)){
    dialog <- gtkMessageDialog(parent = aa,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "Select Filtering")
    response <- dialog$run()
```

```
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }
  if(ind==0){
    dialog <- gtkMessageDialog(parent = aa,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No value selected from the table")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  } else {
    aa.valuelabel$setText(aa.modelFilesCases[ind, 2])
    ###Clear the Main View
    gtkTreeViewClearColumns(aa.view2)
    ###Make the Table
    DFTable <- getAttributesbyFile(filter = type, fid=value)
    TableModel <- rGtkDataFrame(DFTable)
    aa.view2$setModel(TableModel)
    cell_renderer1 <- gtkCellRendererText()
    cell_renderer1["background"] <-"gray92"
    mapply(aa.view2$insertColumnWithAttributes,
           position = -1,
           title= "Attribute",
           cell =list(cell_renderer1),
           text = 1-1)
    cell_renderer2 <- gtkCellRendererText()
    mapply(aa.view2$insertColumnWithAttributes,
           position = -1,
           title= "Value",
           cell =list(cell_renderer2),
           text = 2-1)
  }
}, data = aa.filteredFilesCases)
#
gSignalConnect(aa.text$print, "clicked", function(button){
  aa.buffer$setText("")
  TextBounds<- aa.buffer$getBounds()
  type <- aa.boxes$attr$getActiveText()
  if(is.null(type)){
    dialog <- gtkMessageDialog(parent = aa,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "Select Filtering")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  } else {
    for (i in (1:length(aa.radiogp))){
      if(aa.radiobox[[i]]["active"]){
        attribute <- aa.radiobox[[i]]$getLabel()
      }
    }
    aa.attrlabel$setText(attribute)
    values <- getFilesbyAttribute(type, attribute)
    if(length(values)){
      for (i in (1:length(values))){
        aa.buffer$insertWithTagsByName(TextBounds$end,names(values)[i], "valuename")
        aa.buffer$insert(TextBounds$end,"\n")
        for (j in (1:nrow(values[[i]]))){
          aa.buffer$insert(TextBounds$end,"\t")
          aa.buffer$insert(TextBounds$end, values[[i]]$name[j])
          aa.buffer$insert(TextBounds$end,"\n")
```

```
        }
        aa.buffer$insert(TextBounds$end,"\n")
      }
    }else {
      if(type=="Files"){aa.buffer$setText("No files with this attribute")}
      if(type=="Cases"){aa.buffer$setText("No cases with this attribute")}
    }
  }
 })
 gSignalConnect(aa.text$clear, "clicked",function(clear){
  aa.buffer$setText("")
 })
 gSignalConnect(aa.text$txt, "clicked", function(txt){
  gtkTextViewExportBuffer(aa.buffer, "Attribute")
 })
 ###---------------END
 aa$show()
 aa$move(gdkScreenWidth()*1/3,gdkScreenHeight()*1/4)
}
```

- **CodeCodeWindow.R**

```
###---------------CODE CODE ANALYSIS-------------------------###
CodeCodeWindow <- function(){
 ###---------------DATA TABLES----------------------###
 sql <- RQDAQuery("select id,name from freecode where status=1")
 tableCodes <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Codes", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select catid,name from codecat where status=1")
 tableCodeCat <- data.frame(id = sql[[1]], name = sql[[2]],
                            type= rep("Code Categories", nrow(sql)),
                            visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select id,name from source where status=1")
 tableFiles <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Files", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select catid,name from filecat where status=1")
 tableFileCat <- data.frame(id = sql[[1]], name = sql[[2]],
                            type= rep("File Categories", nrow(sql)),
                            visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select id,name from cases where status=1")
 tableCases <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Cases", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sqlAttributes <- RQDAQuery("select name from attributes where status=1")
 DFs <- list()
 DFs$codes <- rbind(tableCodes, tableCodeCat)
 DFs$files <- rbind(tableFiles, tableFileCat, tableCases)
 DFs$FilesCases <- rbind(tableFiles, tableCases)
 ###---------------WINDOW
 cca <- gtkWindow(show=FALSE)
 cca$setTitle("RQDAExt - Code Vs Code Analysis")
 cca$setDefaultSize(gdkScreenWidth()*2/3,gdkScreenHeight()*2/3)
 cca["border-width"] <- 10
 icon <- system.file("icon", "mainIcon.png", package = "RQDAExt")
 image <- gdkPixbufNewFromFile(icon)
 cca$setIcon(image$retval)
 ###---------------COMPONENTS
 cca.boxes <- list()
 cca.boxes$coderows <- gtkComboBoxNewText()
 cca.boxes$codecolumns <- gtkComboBoxNewText()
 cca.boxes$files <- gtkComboBoxNewText()
```

```
cca.boxes$relation <- gtkComboBoxNewText()
sapply(c("Codes", "Code Categories"), cca.boxes$coderows$appendText)
sapply(c("Codes", "Code Categories"), cca.boxes$codecolumns$appendText)
sapply(c("Files","File Categories","Cases"), cca.boxes$files$appendText)
sapply(c("overlap","inclusion","exact","proximity"), cca.boxes$relation$appendText)
#
cca.selectall <- list()
cca.selectall$button1 <- gtkButton("Select All")
cca.selectall$button2 <- gtkButton("Select All")
cca.selectall$button3 <- gtkButton("Select All")
#
cca.general <-list()
cca.general$close <- gtkButton("Close")
cca.general$help <- gtkButton("Help")
#
cca.table <-list()
cca.table$clear <- gtkButton("Clear")
cca.table$txt <- gtkButton("Export in txt")
cca.table$csv <- gtkButton("Export in csv")
cca.table$print <- gtkButton("Print")
#
cca.graph <-list()
cca.graph$clear <-gtkButton("Clear")
cca.graph$expnod <-gtkButton("Export nodes")
cca.graph$explin <-gtkButton("Export links")
cca.graph$export <-gtkButton("Export graph")
cca.graph$plot <-gtkButton("Plot")
#
cca.adjust.nw <- list()
cca.adjust.nw$cex <- gtkSpinButton(min = 0.5, max = 1.5, step= 0.05)
cca.adjust.nw$size <- gtkSpinButton(min = 5, max = 15, step= 0.5)
cca.adjust.nw$color <- gtkComboBoxNewText()
cca.adjust.nw$layout <- gtkComboBoxNewText()
cca.adjust.nw$codecat <- gtkCheckButton("Sort by Categories")
cca.adjust.nw$weight <- gtkCheckButton("Add weight")
sapply(c("Randomly","Star","Circle","Grid","Nicely","Davidson.Harel",
        "Kamada.Kawai","Fruchterman.Reingold","GEM","Graphopt"),
      cca.adjust.nw$layout$appendText)
cca.adjust.nw$layout["active"] <- 0
sapply(c("Gray","Blue1","Blue2","Green1","Green2", "Yellow","Red", "Orange",
        "Purple"), cca.adjust.nw$color$appendText)
cca.adjust.nw$color["active"] <- 0
#
cca.adjust.hm <- list()
cca.adjust.hm$title <- gtkEntry()
cca.adjust.hm$cex <- gtkSpinButton(min = 0.5, max = 1, step= 0.05)
cca.adjust.hm$margin <- gtkSpinButton(min = 5, max = 15, step= 0.5)
cca.adjust.hm$color <- gtkComboBoxNewText()
sapply(c("Style1","Style2","Style3","Style4","Style5", "Style6","Style7",
        "Style8"), cca.adjust.hm$color$appendText)
cca.adjust.hm$color["active"] <- 0
#
cca.view <- gtkTreeView()
cca.scroll <- gtkScrolledWindow()
cca.scroll$setPolicy("automatic", "automatic")
cca.scroll$add(cca.view)
#
cca.device <- gtkDrawingArea()
asCairoDevice(cca.device)
cca.book <- gtkNotebook()
cca.book2 <- gtkNotebook()
#
cca.models <- list()
```

```
cca.models$coderows <- rGtkDataFrame(DFs$codes)
cca.models$codecolumns <- rGtkDataFrame(DFs$codes)
cca.models$files <- rGtkDataFrame(DFs$files)
#
cca.filteredmodels <-list()
cca.filteredmodels$coderows <- cca.models$coderows$filter()
cca.filteredmodels$coderows$setVisibleColumn(ncol(DFs$codes)-1)
cca.filteredmodels$codecolumns <- cca.models$codecolumns$filter()
cca.filteredmodels$codecolumns$setVisibleColumn(ncol(DFs$codes)-1)
cca.filteredmodels$files <- cca.models$files$filter()
cca.filteredmodels$files$setVisibleColumn(ncol(DFs$files)-1)
#
cca.views <- list()
cca.views$coderows <- gtkTreeView(cca.filteredmodels$coderows)
cca.views$codecolumns <- gtkTreeView(cca.filteredmodels$codecolumns)
cca.views$files <- gtkTreeView(cca.filteredmodels$files)
sapply (cca.views, function (view){
  view$insertColumnWithAttributes (0, "Select names", gtkCellRendererText(), text = 1)
})
sapply (cca.views, function (view){
  selection <- view$getSelection()
  selection$setMode("multiple")
})
#
cca.scrolls <- list()
cca.scrolls$coderows <- gtkScrolledWindow()
cca.scrolls$codecolumns <- gtkScrolledWindow()
cca.scrolls$files <- gtkScrolledWindow()
mapply (gtkContainerAdd, object = cca.scrolls, widget = cca.views)
mapply (gtkScrolledWindowSetPolicy, cca.scrolls, "automatic", "automatic")
###---------------CONTAINERS
cca.container <- gtkHBox(TRUE, 10)
cca.frame1 <- gtkFrame ("Filtering System")
cca.frame2 <- gtkFrame ("Data & Graphics Display")
cca.vbox1 <- gtkVBox(FALSE, 10)
cca.vbox2 <- gtkVBox(FALSE, 10)
cca.vbox3 <- gtkVBox(FALSE, 5)
cca.vbox4 <- gtkVBox(FALSE, 5)
cca.grid <- gtkTable(rows = 5, columns = 3, homogeneous = FALSE)
#
cca.grid.nw <- gtkTable(rows = 2, columns = 2, homogeneous = FALSE)
cca.frame.nw <- gtkFrame ("Color")
cca.vbox5 <- gtkVBox(FALSE, 5)
cca.frame.nw2 <- gtkFrame ("Layout")
cca.frame.nw3 <- gtkFrame ("Adjustment")
cca.grid.nw2 <- gtkTable(rows = 2, columns = 2, homogeneous = FALSE)
#
cca.grid.hm <- gtkTable(rows = 3, columns = 2, homogeneous = FALSE)
cca.grid.hm2 <- gtkTable(rows = 2, columns = 2, homogeneous = FALSE)
cca.frame.hm <- gtkFrame ("Color")
cca.frame.hm2 <- gtkFrame ("Adjustment")
#
cca.hbox1 <- gtkHBox(FALSE, 5)
cca.hbox2 <- gtkHBox(FALSE, 5)
cca.hbox3 <- gtkHBox(FALSE, 5)
#
cca.book$appendPage(cca.vbox3, gtkLabel("Table of Concurrences"))
cca.book$appendPage(cca.vbox4, gtkLabel("Network Graphs"))
cca.book2$appendPage(cca.grid.nw, gtkLabel("Network Graph"))
cca.book2$appendPage(cca.grid.hm, gtkLabel("HeatMap Graph"))
###---------------GENERAL LAYOUT
cca$add(cca.container)
cca.container$packStart(cca.vbox1)
```

```
cca.container$packStart(cca.frame2)
#
cca.vbox1$packStart(cca.frame1, expand = TRUE, fill = TRUE)
cca.vbox1$packStart(cca.hbox1, expand=FALSE, fill = FALSE)
#
cca.frame1$add(cca.grid)
cca.grid$attach(gtkLabel("Row Codes"), left.attach =0,1 , top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(gtkLabel("Column Codes"), left.attach = 1,2, top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(gtkLabel("Files"), left.attach = 2,3, top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(cca.boxes$coderows, left.attach = 0,1, top.attach = 1,2,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(cca.boxes$codecolumns, left.attach = 1,2, top.attach =1,2 ,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(cca.boxes$files, left.attach = 2,3, top.attach = 1,2,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(cca.scrolls$coderows, left.attach = 0,1, top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
cca.grid$attach(cca.scrolls$codecolumns, left.attach = 1,2, top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
cca.grid$attach(cca.scrolls$files, left.attach = 2,3, top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
cca.grid$attach(cca.selectall$button1, left.attach = 0,1, top.attach = 3,4,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(cca.selectall$button2, left.attach = 1,2, top.attach = 3,4,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(cca.selectall$button3, left.attach = 2,3, top.attach = 3,4,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(gtkLabel("Type of relation:"), left.attach = 1,2, top.attach = 4,5,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$attach(cca.boxes$relation, left.attach = 2,3, top.attach = 4,5,
            xoptions = c("expand", "fill"), yoptions = "")
cca.grid$setColSpacing(0,5)
cca.grid$setColSpacing(1,5)
cca.grid$setRowSpacing(1,5)
cca.grid$setRowSpacing(2,5)
cca.grid$setRowSpacing(3,5)
cca.grid$setRowSpacing(4,5)
#
sapply(cca.general, cca.hbox1$packStart, expand = TRUE)
cca.hbox1$packStart(gtkLabel(""),expand=TRUE, fill = TRUE)
#
cca.frame2$add(cca.book)
cca.vbox3$packStart(cca.scroll, expand =TRUE, fill=TRUE)
cca.vbox3$packStart(cca.hbox2, expand =FALSE)
#
cca.hbox2$packStart(gtkLabel(""), expand = TRUE, fill = TRUE)
sapply(cca.table, cca.hbox2$packStart, expand = TRUE)
#
cca.vbox4$packStart(cca.device, expand =TRUE, fill=TRUE)
cca.vbox4$packStart(cca.book2, expand =FALSE)
cca.vbox4$packStart(cca.hbox3, expand =FALSE)
#
cca.hbox3$packStart(gtkLabel(""), expand = TRUE, fill = TRUE)
sapply(cca.graph, cca.hbox3$packStart, expand = TRUE)
###--------------ADJUST LAYOUT
cca.grid.nw$attach(cca.frame.nw, left.attach =0,1 , top.attach = 0,1,
              xoptions = c("expand", "fill"), yoptions = "")
cca.frame.nw$add(cca.vbox5)
cca.vbox5$packStart(cca.adjust.nw$color,expand=FALSE, fill=FALSE)
cca.vbox5$packStart(cca.adjust.nw$codecat,expand=FALSE, fill=FALSE)
```

```
cca.grid.nw$attach(cca.frame.nw2, left.attach =0,1 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
cca.frame.nw2$add(cca.adjust.nw$layout)
cca.grid.nw$attach(cca.frame.nw3, left.attach =1,2 , top.attach = 0,1,
                   xoptions = c("expand", "fill"), yoptions = "")
cca.grid.nw$attach(cca.adjust.nw$weight, left.attach =1,2 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
cca.grid.nw$setColSpacing(0,5)
cca.frame.nw3$add(cca.grid.nw2)
cca.grid.nw2$attach(gtkLabel("Cex"), left.attach =0,1 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
cca.grid.nw2$attach(gtkLabel("Size"), left.attach =0,1 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
cca.grid.nw2$attach(cca.adjust.nw$cex, left.attach =1,2 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
cca.grid.nw2$attach(cca.adjust.nw$size, left.attach =1,2 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
#
cca.grid.hm$attach(gtkLabel("Title"), left.attach =0,2 , top.attach = 0,1,
                   xoptions = c("expand", "fill"), yoptions = "")
cca.grid.hm$attach(cca.adjust.hm$title, left.attach =0,2 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
cca.grid.hm$attach(cca.frame.hm, left.attach =0,1 , top.attach = 2,3,
                   xoptions = c("expand", "fill"), yoptions = "")
cca.frame.hm$add(cca.adjust.hm$color)
cca.grid.hm$attach(cca.frame.hm2, left.attach =1,2 , top.attach = 2,3,
                   xoptions = c("expand", "fill"), yoptions = "")
cca.grid.hm$setColSpacing(0,5)
cca.frame.hm2$add(cca.grid.hm2)
cca.grid.hm2$attach(gtkLabel("Cex"), left.attach =0,1 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
cca.grid.hm2$attach(gtkLabel("Margin"), left.attach =0,1 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
cca.grid.hm2$attach(cca.adjust.hm$cex, left.attach =1,2 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
cca.grid.hm2$attach(cca.adjust.hm$margin, left.attach =1,2 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
###--------------SIGNAL CONNECTIONS
gSignalConnect(cca.boxes$coderows, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF[,"type"]
  DF[,"visible"] <- grepl(pattern, values)
}, data= cca.filteredmodels$coderows)
gSignalConnect(cca.boxes$codecolumns, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF[,"type"]
  DF[,"visible"] <- grepl(pattern, values)
}, data= cca.filteredmodels$codecolumns)
gSignalConnect(cca.boxes$files, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF[,"type"]
  DF[,"visible"] <- grepl(pattern, values)
}, data= cca.filteredmodels$files)
gSignalConnect(cca.selectall$button1, "clicked", function (button){
  cca.views$coderows$getSelection()$SelectAll()
})
gSignalConnect(cca.selectall$button2, "clicked", function (button){
  cca.views$codecolumns$getSelection()$SelectAll()
})
gSignalConnect(cca.selectall$button3, "clicked", function (button){
```

```
  cca.views$files$getSelection()$SelectAll()
})
#
gSignalConnect(cca.general$close, "clicked", function(button){
  cca$destroy()
})
gSignalConnect(cca.general$help, "clicked", function(button){
  InstructionsWindow(active=2)
})
#
gSignalConnect(cca.graph$expnod, "clicked", function (button){
  CBoxValues <- c(cca.boxes$coderows$getActiveText(),
                  cca.boxes$codecolumns$getActiveText(),
                  cca.boxes$files$getActiveText())
  CBoxRelation <- cca.boxes$relation$getActiveText()
  ind1 <- cca.views$coderows$selectedIndices()
  ind2<- cca.views$codecolumns$selectedIndices()
  ind3 <- cca.views$files$selectedIndices()
  if(is.null(CBoxRelation)){
    dialog <- gtkMessageDialog(parent = cca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "Select type of relation")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }
  if(ind1==0 || ind2==0 || ind3==0){
    dialog <- gtkMessageDialog(parent = cca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    RowValues <- cca.models$coderows[ind1,1]
    ColumnValues <- cca.models$codecolumns[ind2,1]
    FileValues <- cca.models$files[ind3,1]


    DFMatrix <- CodeCodeMatrix(filter= CBoxValues,rowcid  = RowValues,
                               colcid = ColumnValues, fid = FileValues,
                               relation=CBoxRelation)
    NodLin<- getLinksAndNodes(DFMatrix, type = 1)
    gtkExportNet(NodLin$nodes, "nodes")
  }
})
gSignalConnect(cca.graph$explin, "clicked", function (button){
  CBoxValues <- c(cca.boxes$coderows$getActiveText(),
                  cca.boxes$codecolumns$getActiveText(),
                  cca.boxes$files$getActiveText())
  CBoxRelation <- cca.boxes$relation$getActiveText()
  ind1 <- cca.views$coderows$selectedIndices()
  ind2<- cca.views$codecolumns$selectedIndices()
  ind3 <- cca.views$files$selectedIndices()
  if(is.null(CBoxRelation)){
    dialog <- gtkMessageDialog(parent = cca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "Select type of relation")

    response <- dialog$run()
```

```
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }
  if(ind1==0 || ind2==0 || ind3==0){
    dialog <- gtkMessageDialog(parent = cca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    RowValues <- cca.models$coderows[ind1,1]
    ColumnValues <- cca.models$codecolumns[ind2,1]
    FileValues <- cca.models$files[ind3,1]
    DFMatrix <- CodeCodeMatrix(filter= CBoxValues,rowcid  = RowValues,
                               colcid = ColumnValues, fid = FileValues,
                               relation=CBoxRelation)
    NodLin<- getLinksAndNodes(DFMatrix, type = 1)
    gtkExportNet(NodLin$links, "links")
  }
})
gSignalConnect(cca.table$print, "clicked", function (button){
  CBoxValues <- c(cca.boxes$coderows$getActiveText(),
                  cca.boxes$codecolumns$getActiveText(),
                  cca.boxes$files$getActiveText())
  CBoxRelation <- cca.boxes$relation$getActiveText()
  ind1 <- cca.views$coderows$selectedIndices()
  ind2<- cca.views$codecolumns$selectedIndices()
  ind3 <- cca.views$files$selectedIndices()
  if(is.null(CBoxRelation)){
    dialog <- gtkMessageDialog(parent = cca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "Select type of relation")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }
  if(ind1==0 || ind2==0 || ind3==0){
    dialog <- gtkMessageDialog(parent = cca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    RowValues <- cca.models$coderows[ind1,1]
    ColumnValues <- cca.models$codecolumns[ind2,1]
    FileValues <- cca.models$files[ind3,1]
    ###Clear the Main View
    gtkTreeViewClearColumns(cca.view)
    DFMatrix <- CodeCodeMatrix(filter= CBoxValues,rowcid  = RowValues,
                               colcid = ColumnValues, fid = FileValues,
                               relation=CBoxRelation, as.matrix = FALSE)
    MatrixModel <- rGtkDataFrame(DFMatrix)
    cca.view$setModel(MatrixModel)
    cca.view$insertColumnWithAttributes(position=-1,
                                        title="",
                                        cell=gtkCellRendererText(),
                                        text = 0)
    cell_renderer <- gtkCellRendererText()
```

```
    cell_renderer["background"] <-"gray92"
    cell_renderer["xalign"] <- 0.5
    cell_names <- colnames(MatrixModel)
    mapply(cca.view$insertColumnWithAttributes,
           position = -1,
           title=cell_names[2:(ncol(MatrixModel))],
           cell =list(cell_renderer),
           text = seq_len(ncol(MatrixModel)-1))
  }
})
gSignalConnect(cca.table$clear, "clicked", function (button){
  gtkTreeViewClearColumns(cca.view)
})
gSignalConnect(cca.table$csv, "clicked", function (button){
  gtkTreeViewExportModel(cca.view, ".csv", "CodeCodeMatrix")
})
gSignalConnect(cca.table$txt, "clicked", function (button){
  gtkTreeViewExportModel(cca.view, ".txt", "CodeCodeMatrix")
})
#
gSignalConnect(cca.graph$clear, "clicked", function (button){
  frame()
})
gSignalConnect(cca.graph$plot, "clicked", function (button){
  CBoxValues <- c(cca.boxes$coderows$getActiveText(),
                  cca.boxes$codecolumns$getActiveText(),
                  cca.boxes$files$getActiveText())
  CBoxRelation <- cca.boxes$relation$getActiveText()
  ind1 <- cca.views$coderows$selectedIndices()
  ind2<- cca.views$codecolumns$selectedIndices()
  ind3 <- cca.views$files$selectedIndices()
  if(is.null(CBoxRelation)){
    dialog <- gtkMessageDialog(parent = cca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "Select type of relation")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }
  if(ind1==0 || ind2==0 || ind3==0){
    dialog <- gtkMessageDialog(parent = cca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    RowValues <- cca.models$coderows[ind1,1]
    ColumnValues <- cca.models$codecolumns[ind2,1]
    FileValues <- cca.models$files[ind3,1]
    DFMatrix <- CodeCodeMatrix(filter= CBoxValues,rowcid  = RowValues,
                               colcid = ColumnValues, fid = FileValues,
                               relation=CBoxRelation)
    if(cca.book2$getCurrentPage()==0){
      NodLin <- getLinksAndNodes(DFMatrix, type = 1)
      if(ncol(NodLin$links)==0){stop(("No edges between nodes"))}else {
        Net <- graph.data.frame(NodLin$links, NodLin$nodes, directed=F)
        Net <- simplify(Net, remove.multiple = F, remove.loops = T)
        cex <- cca.adjust.nw$cex$getValue()
        size <- cca.adjust.nw$size$getValue()
        weight <- ifelse(cca.adjust.nw$weight$active, 1,2)
```

```
    codecat <- ifelse(cca.adjust.nw$codecat$active, 1,2)
    switch(cca.adjust.nw$layout$getActiveText(), Randomly= {l<-layout_randomly(Net)},
          Circle={l<-layout_in_circle(Net)}, GEM= {l<-layout_with_gem(Net)},
          Star={l<-layout_as_star(Net)}, Grid={l<-layout_on_grid(Net)},
          Nicely={l<-layout_nicely(Net)}, Davison.Harel={l<-layout_with_dh(Net)},
          Graphopt={l<-layout_with_graphopt(Net)}, Kamada.Kawai={l<-layout_with_kk(Net)},
          Fruchterman.Reingold={l<-layout_with_fr(Net)})
    switch(weight, {weight<- SummaryOfCodings(c("Codes",CBoxValues[3]),
                                    NodLin$nodes$id,FileValues, "Number of Codings");
    E(Net)$width <- E(Net)$weight/4;V(Net)$size <- weight$value*size/10},
    V(Net)$size <- size)
    switch(codecat, {
     switch(cca.adjust.nw$color$getActiveText(),
          Blue1={colrs<- c("cyan","light cyan","turquoise","dark turquoise",
                    "aquamarine", "medium aquamarine", "cadet blue","powder blue")},
          Blue2={colrs<- c("sky blue","deep sky blue","dodger blue",
                    "royal blue", "blue", "navy","slate blue","snow")},
          Gray={colrs<- c("light gray","gray","light slate gray",
                   "slate gray", "dim gray", "lavender","bisque4","snow")},
          Green1={colrs<- c("green", "chartreuse","spring green","green yellow",
                    "lime green", "yellow green", "forest green","snow")},
          Green2={colrs<- c("forest green", "olive drab","sea green","medium sea green",
                    "aquamarine", "pale green", "turquoise","snow")},
          Yellow={colrs<- c("yellow", "gold","light goldenrod","goldenrod",
                    "dark goldenrod", "khaki", "light yellow","snow")},
          Orange={colrs<- c("orange", "dark orange","coral","wheat",
                    "sandy brown", "peru", "sienna","brown")},
          Red={colrs<- c("red", "tomato","indian red","salmon",
                  "orange","brown","maroon","snow")},
          Purple={colrs<- c("purple", "medium orchid", "pink", "medium purple",
                    "deep pink","magenta","dark violet","snow")})
    },{switch(cca.adjust.nw$color$getActiveText(), Blue1={V(Net)$color<-"cyan"},
           Blue2={V(Net)$color<-"dodger blue"},Gray={V(Net)$color<-"gray"},
           Green1={V(Net)$color<-"green"},Green2={V(Net)$color<-"forest green"},
           Yellow={V(Net)$color<-"yellow"},Orange={V(Net)$color<-"orange"},
           Red={V(Net)$color<-"red"},Purple={V(Net)$color<-"purple"})})
    switch(codecat, V(Net)$color<-colrs[V(Net)$catid])
    plot(Net,vertex.label.cex=cex, vertex.label.dist=0, edge.curved=.1,
        vertex.frame.color="#555555", vertex.label.color="black", layout=l)
   }
  }
  if(cca.book2$getCurrentPage()==1){
   margin <- cca.adjust.hm$margin$getValue()
   cex <- cca.adjust.hm$cex$getValue()
   title <- cca.adjust.hm$title$getText()
   switch(cca.adjust.hm$color$getActiveText(),
         Style1={palf <- colorRampPalette(c("gold", "dark orange","red"))},
         Style2={palf <- colorRampPalette(c("blue", "sky blue", "cyan"))},
         Style3={palf <- colorRampPalette(c("green", "yellow", "red"))},
         Style4={palf <- colorRampPalette(c("blue", "green", "yellow"))},
         Style5={palf <- colorRampPalette(c("dark green", "green", "green yellow"))},
         Style6={palf <- colorRampPalette(c("black", "gray", "white"))},
         Style7={palf <- colorRampPalette(c("purple", "magenta", "pink"))},
         Style8={palf <- colorRampPalette(c("blue", "green", "red"))})
   heatmap(DFMatrix, Rowv = NA, Colv = NA, col = palf(100),
          cexRow = cex, cexCol = cex, main=title,cex.main= cex+0.15,
          scale="none", margins=c(margin,margin))
  }
 }
})
###----------END
cca$show()
cca$move(gdkScreenWidth()*1/3,gdkScreenHeight()*1/4)
```

```
}
```

- **CodeFileWindow.R**

```
###---------------CODE FILE ANALYSIS-------------------------###
CodeFileWindow <- function(){
  ###---------------DATA TABLES----------------------###
  sql <- RQDAQuery("select id,name from freecode where status=1")
  tableCodes <- data.frame(id = sql[[1]], name = sql[[2]],
                           type= rep("Codes", nrow(sql)),
                           visible = rep(FALSE, nrow(sql)))
  sql <- RQDAQuery("select catid,name from codecat where status=1")
  tableCodeCat <- data.frame(id = sql[[1]], name = sql[[2]],
                             type= rep("Code Categories", nrow(sql)),
                             visible = rep(FALSE, nrow(sql)))
  sql <- RQDAQuery("select id,name from source where status=1")
  tableFiles <- data.frame(id = sql[[1]], name = sql[[2]],
                           type= rep("Files", nrow(sql)),
                           visible = rep(FALSE, nrow(sql)))
  sql <- RQDAQuery("select catid,name from filecat where status=1")
  tableFileCat <- data.frame(id = sql[[1]], name = sql[[2]],
                             type= rep("File Categories", nrow(sql)),
                             visible = rep(FALSE, nrow(sql)))
  sql <- RQDAQuery("select id,name from cases where status=1")
  tableCases <- data.frame(id = sql[[1]], name = sql[[2]],
                           type= rep("Cases", nrow(sql)),
                           visible = rep(FALSE, nrow(sql)))
  sqlAttributes <- RQDAQuery("select name from attributes where status=1")
  DFs <- list()
  DFs$codes <- rbind(tableCodes, tableCodeCat)
  DFs$files <- rbind(tableFiles, tableFileCat, tableCases)
  DFs$FilesCases <- rbind(tableFiles, tableCases)
  ###---------------WINDOW
  cfa <- gtkWindow(show=FALSE)
  cfa$setTitle("RQDAExt - Code Vs File Analysis")
  cfa$setDefaultSize(gdkScreenWidth()*2/3,gdkScreenHeight()*2/3)
  cfa["border-width"] <- 10
  icon <- system.file("icon", "mainIcon.png", package = "RQDAExt")
  image <- gdkPixbufNewFromFile(icon)
  cfa$setIcon(image$retval)
  ###---------------COMPONENTS
  cfa.boxes <- list()
  cfa.boxes$codes <- gtkComboBoxNewText()
  cfa.boxes$files <- gtkComboBoxNewText()
  sapply(c("Codes", "Code Categories"), cfa.boxes$codes$appendText)
  sapply(c("Files","File Categories","Cases"), cfa.boxes$files$appendText)
  #
  cfa.selectall <- list()
  cfa.selectall$button1 <- gtkButton("Select All")
  cfa.selectall$button2 <- gtkButton("Select All")
  #
  cfa.general <-list()
  cfa.general$close <- gtkButton("Close")
  cfa.general$help <- gtkButton("Help")
  #
  cfa.table <-list()
  cfa.table$clear <- gtkButton("Clear")
  cfa.table$txt <- gtkButton("Export in txt")
  cfa.table$csv <- gtkButton("Export in csv")
  cfa.table$print <- gtkButton("Print")
  #
  cfa.graph<-list()
  cfa.graph$clear <-gtkButton("Clear")
```

```
cfa.graph$expnod <-gtkButton("Export nodes")
cfa.graph$explin <-gtkButton("Export links")
cfa.graph$export <-gtkButton("Export graph")
cfa.graph$plot <-gtkButton("Plot")
#
cfa.view <- gtkTreeView()
cfa.scroll <- gtkScrolledWindow()
cfa.scroll$setPolicy("automatic", "automatic")
cfa.scroll$add(cfa.view)
#
cfa.device <- gtkDrawingArea()
asCairoDevice(cfa.device)
cfa.book <- gtkNotebook()
cfa.book2 <- gtkNotebook()
#
cfa.adjust.nw <- list()
cfa.adjust.nw$cex <- gtkSpinButton(min = 0.5, max = 1.5, step= 0.05)
cfa.adjust.nw$size <- gtkSpinButton(min = 5, max = 15, step= 0.5)
cfa.adjust.nw$color <- gtkComboBoxNewText()
cfa.adjust.nw$color2 <- gtkComboBoxNewText()
cfa.adjust.nw$layout <- gtkComboBoxNewText()
cfa.adjust.nw$weight <- gtkCheckButton("Add weight")
sapply(c("Randomly","Star","Circle","Grid","Bipartite","Nicely","Davidson.Harel",
         "Kamada.Kawai","Fruchterman.Reingold","GEM","Graphopt"),
       cfa.adjust.nw$layout$appendText)
cfa.adjust.nw$layout["active"] <- 0
sapply(c("Gray","Blue1","Blue2","Green1","Green2", "Yellow","Red", "Orange","Purple"),
       cfa.adjust.nw$color$appendText)
sapply(c("Gray","Blue1","Blue2","Green1","Green2", "Yellow","Red", "Orange","Purple"),
       cfa.adjust.nw$color2$appendText)
cfa.adjust.nw$color["active"] <- 0
cfa.adjust.nw$color2["active"] <- 0
#
cfa.adjust.hm <- list()
cfa.adjust.hm$title <- gtkEntry()
cfa.adjust.hm$cex <- gtkSpinButton(min = 0.5, max = 1, step= 0.05)
cfa.adjust.hm$margin <- gtkSpinButton(min = 5, max = 15, step= 0.5)
cfa.adjust.hm$color <- gtkComboBoxNewText()
sapply(c("Style1","Style2","Style3","Style4","Style5", "Style6","Style7", "Style8"),
       cfa.adjust.hm$color$appendText)
cfa.adjust.hm$color["active"] <- 0
#
cfa.models <- list()
cfa.models$codes <- rGtkDataFrame(DFs$codes)
cfa.models$files <- rGtkDataFrame(DFs$files)
#
cfa.filteredmodels <-list()
cfa.filteredmodels$codes <- cfa.models$codes$filter()
cfa.filteredmodels$codes$setVisibleColumn(ncol(DFs$codes)-1)
cfa.filteredmodels$files <- cfa.models$files$filter()
cfa.filteredmodels$files$setVisibleColumn(ncol(DFs$files)-1)
#
cfa.views <- list()
cfa.views$codes <- gtkTreeView(cfa.filteredmodels$codes)
cfa.views$files <- gtkTreeView(cfa.filteredmodels$files)
sapply (cfa.views, function (view){
  view$insertColumnWithAttributes (0, "Select names", gtkCellRendererText(), text = 1)
})
sapply (cfa.views, function (view){
  selection <- view$getSelection()
  selection$setMode("multiple")
})
#
```

```
cfa.scrolls <- list()
cfa.scrolls$codes <- gtkScrolledWindow()
cfa.scrolls$files <- gtkScrolledWindow()
mapply (gtkContainerAdd, object = cfa.scrolls, widget = cfa.views)
mapply (gtkScrolledWindowSetPolicy, cfa.scrolls, "automatic", "automatic")
###----------CONTAINERS
cfa.container <- gtkHBox(TRUE, 10)
cfa.frame1 <- gtkFrame ("Filtering System")
cfa.frame2 <- gtkFrame ("Data & Graphics Display")
cfa.vbox1 <- gtkVBox(FALSE, 10)
cfa.vbox2 <- gtkVBox(FALSE, 10)
cfa.vbox3 <- gtkVBox(FALSE, 5)
cfa.vbox4 <- gtkVBox(FALSE, 5)
cfa.grid <- gtkTable(rows = 4, columns = 2, homogeneous = FALSE)
cfa.hbox1 <- gtkHBox(FALSE, 5)
cfa.hbox2 <- gtkHBox(FALSE, 5)
cfa.hbox3 <- gtkHBox(FALSE, 5)
#
cfa.grid.nw <- gtkTable(rows = 2, columns = 2, homogeneous = FALSE)
cfa.frame.nw <- gtkFrame ("Color")
cfa.vbox5 <- gtkVBox(FALSE, 5)
cfa.frame.nw2 <- gtkFrame ("Layout")
cfa.frame.nw3 <- gtkFrame ("Adjustment")
cfa.grid.nw2 <- gtkTable(rows = 2, columns = 2, homogeneous = FALSE)
#
cfa.grid.hm <- gtkTable(rows = 3, columns = 2, homogeneous = FALSE)
cfa.grid.hm2 <- gtkTable(rows = 2, columns = 2, homogeneous = FALSE)
cfa.frame.hm <- gtkFrame ("Color")
cfa.frame.hm2 <- gtkFrame ("Adjustment")
###----------LAYOUT
cfa.book$appendPage(cfa.vbox3, gtkLabel("Table of Concurrences"))
cfa.book$appendPage(cfa.vbox4, gtkLabel("Network Graphs"))
cfa.book2$appendPage(cfa.grid.nw, gtkLabel("Network Chart"))
cfa.book2$appendPage(cfa.grid.hm, gtkLabel("HeatMap Chart"))
#
cfa$add(cfa.container)
cfa.container$packStart(cfa.vbox1)
cfa.container$packStart(cfa.frame2)
#
cfa.vbox1$packStart(cfa.frame1, expand = TRUE, fill = TRUE)
cfa.vbox1$packStart(cfa.hbox1, expand=FALSE, fill = FALSE)
#
cfa.frame1$add(cfa.grid)
cfa.grid$attach(gtkLabel("Codes"), left.attach =0,1 , top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
cfa.grid$attach(gtkLabel("Files"), left.attach = 1,2, top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
cfa.grid$attach(cfa.boxes$codes, left.attach = 0,1, top.attach = 1,2,
            xoptions = c("expand", "fill"), yoptions = "")
cfa.grid$attach(cfa.boxes$files, left.attach = 1,2, top.attach =1,2 ,
            xoptions = c("expand", "fill"), yoptions = "")
cfa.grid$attach(cfa.scrolls$codes, left.attach = 0,1, top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
cfa.grid$attach(cfa.scrolls$files, left.attach = 1,2, top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
cfa.grid$attach(cfa.selectall$button1, left.attach = 0,1, top.attach = 3,4,
            xoptions = c("expand", "fill"), yoptions = "")
cfa.grid$attach(cfa.selectall$button2, left.attach = 1,2, top.attach = 3,4,
            xoptions = c("expand", "fill"), yoptions = "")
cfa.grid$setColSpacing(0,5)
cfa.grid$setRowSpacing(1,5)
cfa.grid$setRowSpacing(2,5)
cfa.grid$setRowSpacing(3,5)
```

```
#
sapply(cfa.general, cfa.hbox1$packStart, expand = TRUE)
cfa.hbox1$packStart(gtkLabel(""),expand=TRUE, fill = TRUE)
#
cfa.frame2$add(cfa.book)
cfa.vbox3$packStart(cfa.scroll, expand =TRUE, fill=TRUE)
cfa.vbox3$packStart(cfa.hbox2, expand =FALSE)
#
cfa.hbox2$packStart(gtkLabel(""), expand = TRUE, fill = TRUE)
sapply(cfa.table, cfa.hbox2$packStart, expand = TRUE)
#
cfa.vbox4$packStart(cfa.device, expand =TRUE, fill=TRUE)
cfa.vbox4$packStart(cfa.book2, expand =FALSE)
cfa.vbox4$packStart(cfa.hbox3, expand =FALSE)
#
cfa.hbox3$packStart(gtkLabel(""), expand = TRUE, fill = TRUE)
sapply(cfa.graph, cfa.hbox3$packStart, expand = TRUE)
#
cfa.grid.nw$attach(cfa.frame.nw, left.attach =0,1 , top.attach = 0,1,
                   xoptions = c("expand", "fill"), yoptions = "")
cfa.frame.nw$add(cfa.vbox5)
cfa.vbox5$packStart(cfa.adjust.nw$color,expand=FALSE, fill=FALSE)
cfa.vbox5$packStart(cfa.adjust.nw$color2,expand=FALSE, fill=FALSE)
cfa.grid.nw$attach(cfa.frame.nw2, left.attach =0,1 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
cfa.frame.nw2$add(cfa.adjust.nw$layout)
cfa.grid.nw$attach(cfa.frame.nw3, left.attach =1,2 , top.attach = 0,1,
                   xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.nw$attach(cfa.adjust.nw$weight, left.attach =1,2 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.nw$setColSpacing(0,5)

cfa.frame.nw3$add(cfa.grid.nw2)
cfa.grid.nw2$attach(gtkLabel("Size"), left.attach =0,1 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.nw2$attach(gtkLabel("Cex"), left.attach =0,1 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.nw2$attach(cfa.adjust.nw$size, left.attach =1,2 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.nw2$attach(cfa.adjust.nw$cex, left.attach =1,2 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
#
cfa.grid.hm$attach(gtkLabel("Title"), left.attach =0,2 , top.attach = 0,1,
                   xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.hm$attach(cfa.adjust.hm$title, left.attach =0,2 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.hm$attach(cfa.frame.hm, left.attach =0,1 , top.attach = 2,3,
                   xoptions = c("expand", "fill"), yoptions = "")
cfa.frame.hm$add(cfa.adjust.hm$color)
cfa.grid.hm$attach(cfa.frame.hm2, left.attach =1,2 , top.attach = 2,3,
                   xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.hm$setColSpacing(0,5)
cfa.frame.hm2$add(cfa.grid.hm2)
cfa.grid.hm2$attach(gtkLabel("Cex"), left.attach =0,1 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.hm2$attach(gtkLabel("Margin"), left.attach =0,1 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.hm2$attach(cfa.adjust.hm$cex, left.attach =1,2 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
cfa.grid.hm2$attach(cfa.adjust.hm$margin, left.attach =1,2 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
###----------SIGNAL CONNECTIONS
gSignalConnect(cfa.boxes$codes, "changed", function(combo, user.data){
```

```
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF[,"type"]
  DF[,"visible"] <- grepl(pattern, values)
}, data= cfa.filteredmodels$codes)
gSignalConnect(cfa.boxes$files, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF[,"type"]
  DF[,"visible"] <- grepl(pattern, values)
}, data= cfa.filteredmodels$files)
gSignalConnect(cfa.selectall$button1, "clicked", function (button){
  cfa.views$codes$getSelection()$SelectAll()
})
gSignalConnect(cfa.selectall$button2, "clicked", function (button){
  cfa.views$files$getSelection()$SelectAll()
})
#
gSignalConnect(cfa.general$close, "clicked", function(button){
  cfa$destroy()
})
gSignalConnect(cfa.general$help, "clicked", function(button){
  InstructionsWindow(active=3)
})
#
gSignalConnect(cfa.table$print, "clicked", function (button){
  CBoxValues <- list(cfa.boxes$codes$getActiveText(),
                     cfa.boxes$files$getActiveText())
  ind1 <- cfa.views$codes$selectedIndices()
  ind2 <- cfa.views$files$selectedIndices()
  if(ind1==0 || ind2==0){
    dialog <- gtkMessageDialog(parent = cfa,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    CodeValues <- cfa.models$codes[ind1,1]
    FileValues <- cfa.models$files[ind2,1]
    ###Clear the Main View
    gtkTreeViewClearColumns(cfa.view)
    ###Defintion of the Code vs Code Matrix
    DFMatrix <- CodeFileMatrix(filter= CBoxValues,cid = CodeValues,
                               fid = FileValues, as.matrix = FALSE)
    MatrixModel <- rGtkDataFrame(DFMatrix)
    cfa.view$setModel(MatrixModel)
    cfa.view$insertColumnWithAttributes(position=-1,
                                        title="",
                                        cell=gtkCellRendererText(),
                                        text = 0)
    cell_renderer <- gtkCellRendererText()
    cell_renderer["background"] <-"gray92"
    cell_renderer["xalign"] <- 0.5
    cell_names <- colnames(MatrixModel)
    mapply(cfa.view$insertColumnWithAttributes,
           position = -1,
           title=cell_names[2:(ncol(MatrixModel))],
           cell =list(cell_renderer),
           text = seq_len(ncol(MatrixModel)-1))
  }
})
```

```
gSignalConnect (cfa.table$clear, "clicked", function (button){
  gtkTreeViewClearColumns(cfa.view)
})
gSignalConnect(cfa.table$csv, "clicked", function (button){
  gtkTreeViewExportModel(cfa.view, ".csv", "CodeFileMatrix")
})
gSignalConnect(cfa.table$txt, "clicked", function (button){
  gtkTreeViewExportModel(cfa.view, ".txt", "CodeFileMatrix")
})
#
gSignalConnect(cfa.graph$clear, "clicked", function (button){
  frame()
})
gSignalConnect(cfa.graph$plot, "clicked", function(button){
  CBoxValues <- list(cfa.boxes$codes$getActiveText(),
                     cfa.boxes$files$getActiveText())
  ind1 <- cfa.views$codes$selectedIndices()
  ind2 <- cfa.views$files$selectedIndices()
  if(ind1==0 || ind2==0){
    dialog <- gtkMessageDialog(parent = cfa,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    CodeValues <- cfa.models$codes[ind1,1]
    FileValues <- cfa.models$files[ind2,1]
    DFMatrix <- CodeFileMatrix(filter= CBoxValues, cid  = CodeValues,
                               fid = FileValues)
    NodLin <- getLinksAndNodes(DFMatrix, type = 2)
    if(cfa.book2$getCurrentPage()==0){
      if(ncol(NodLin$links)==0){stop(("No edges between nodes"))}else {
        Net <- graph.data.frame(NodLin$links, NodLin$nodes, directed=F)
        Net <- simplify(Net, remove.multiple = F, remove.loops = T)
        cex <- cfa.adjust.nw$cex$getValue()
        size <- cfa.adjust.nw$size$getValue()
        weight <- ifelse(cfa.adjust.nw$weight$active, 1,2)
        a <- 0
        switch(cfa.adjust.nw$color$getActiveText(),
               Blue1={color <- "cyan"},
               Blue2={color <- "dodger blue"},
               Gray={color <- "gray"},
               Green1={color <- "green"},
               Green2={color <- "forest green"},
               Yellow={color <- "yellow"},
               Orange={color <- "orange"},
               Red={color <- "red"},
               Purple={color <- "purple"})
        switch(cfa.adjust.nw$color2$getActiveText(),
               Blue1={color2 <- "cyan"},
               Blue2={color2 <- "dodger blue"},
               Gray={color2 <- "gray"},
               Green1={color2 <- "green"},
               Green2={color2 <- "forest green"},
               Yellow={color2 <- "yellow"},
               Orange={color2 <- "orange"},
               Red={color2 <- "red"},
               Purple={color2 <- "purple"})
        switch(cfa.adjust.nw$layout$getActiveText(), Randomly= {l<-layout_randomly(Net)},
               Circle={l<-layout_in_circle(Net)}, GEM= {l<-layout_with_gem(Net)},
               Star={l<-layout_as_star(Net)}, Grid={l<-layout_on_grid(Net)},
```

```
            Nicely={l<-layout_nicely(Net)}, Davison.Harel={l<-layout_with_dh(Net)},
            Graphopt={l<-layout_with_graphopt(Net)}, Kamada.Kawai={l<-layout_with_kk(Net)},
            Fruchterman.Reingold={l<-layout_with_fr(Net)},
            Bipartite={Net <-graph_from_incidence_matrix(DFMatrix);
            l<-layout_as_bipartite(Net); a <- 1})
      switch(weight, {weight<- SummaryOfCodings(CBoxValues,
                                    CodeValues,FileValues, "Number of Codings");
      weight2 <- SummaryOfCodings(CBoxValues,
                          CodeValues,FileValues, "Codings for Each File");
      weight <- c(weight$value,weight2$value);
      E(Net)$width <- E(Net)$weight/4;V(Net)$size <- weight*size/15},
      V(Net)$size <- size)
      V(Net)$color<-c(color, color2)[V(Net)$type+a]
      V(Net)$shape <- c("circle","square")[V(Net)$type+a]
      plot(Net,vertex.label.cex=cex, vertex.label.dist=0, edge.curved=.1,
          vertex.frame.color="#555555", vertex.label.color="black", layout=l)
     }
    }
    if(cfa.book2$getCurrentPage()==1){
      margin <- cfa.adjust.hm$margin$getValue()
      cex <- cfa.adjust.hm$cex$getValue()
      title <- cfa.adjust.hm$title$getText()
      switch(cfa.adjust.hm$color$getActiveText(),
            Style1={palf <- colorRampPalette(c("gold", "dark orange","red"))},
            Style2={palf <- colorRampPalette(c("blue", "sky blue", "cyan"))},
            Style3={palf <- colorRampPalette(c("green", "yellow", "red"))},
            Style4={palf <- colorRampPalette(c("blue", "green", "yellow"))},
            Style5={palf <- colorRampPalette(c("dark green", "green", "green yellow"))},
            Style6={palf <- colorRampPalette(c("black", "gray", "white"))},
            Style7={palf <- colorRampPalette(c("purple", "magenta", "pink"))},
            Style8={palf <- colorRampPalette(c("blue", "green", "red"))})
      heatmap(DFMatrix, Rowv = NA, Colv = NA, col = palf(100),
            cexRow = cex, cexCol = cex, main=title,cex.main= cex+0.15,
            scale="none", margins=c(margin,margin))
    }
   }
})
gSignalConnect(cfa.graph$expnod, "clicked", function (button){
  CBoxValues <- list(cfa.boxes$codes$getActiveText(),
                cfa.boxes$files$getActiveText())
  ind1 <- cfa.views$codes$selectedIndices()
  ind2 <- cfa.views$files$selectedIndices()
  if(ind1==0 || ind2==0){
    dialog <- gtkMessageDialog(parent = cfa,
                          flags = "destroy-with-parent",
                          type = "warning", button = "ok",
                          "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    CodeValues <- cfa.models$codes[ind1,1]
    FileValues <- cfa.models$files[ind2,1]
    DFMatrix <- CodeFileMatrix(filter= CBoxValues, cid  = CodeValues,
                          fid = FileValues)
    NodLin <- getLinksAndNodes(DFMatrix, type = 2)
    gtkExportNet(NodLin$nodes, "nodes")
  }
})
gSignalConnect(cfa.graph$explin, "clicked", function (button){
  CBoxValues <- list(cfa.boxes$codes$getActiveText(),
                cfa.boxes$files$getActiveText())
  ind1 <- cfa.views$codes$selectedIndices()
```

```
  ind2 <- cfa.views$files$selectedIndices()
  if(ind1==0 || ind2==0){
    dialog <- gtkMessageDialog(parent = cfa,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    CodeValues <- cfa.models$codes[ind1,1]
    FileValues <- cfa.models$files[ind2,1]
    DFMatrix <- CodeFileMatrix(filter= CBoxValues, cid  = CodeValues,
                               fid = FileValues)
    NodLin <- getLinksAndNodes(DFMatrix, type = 2)
    gtkExportNet(NodLin$links, "links")
  }
 })
 ###---------------END
 cfa$show()
 cfa$move(gdkScreenWidth()*1/3,gdkScreenHeight()*1/4)
}
```

- **CodingsWindow.R**

```
###---------------CODING ANALYSIS----------------------------###
CodingsWindow <- function(){
 ###---------------DATA TABLES-----------------------###
 sql <- RQDAQuery("select id,name from freecode where status=1")
 tableCodes <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Codes", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select catid,name from codecat where status=1")
 tableCodeCat <- data.frame(id = sql[[1]], name = sql[[2]],
                            type= rep("Code Categories", nrow(sql)),
                            visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select id,name from source where status=1")
 tableFiles <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Files", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select catid,name from filecat where status=1")
 tableFileCat <- data.frame(id = sql[[1]], name = sql[[2]],
                            type= rep("File Categories", nrow(sql)),
                            visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select id,name from cases where status=1")
 tableCases <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Cases", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sqlAttributes <- RQDAQuery("select name from attributes where status=1")
 DFs <- list()
 DFs$codes <- rbind(tableCodes, tableCodeCat)
 DFs$files <- rbind(tableFiles, tableFileCat, tableCases)
 DFs$FilesCases <- rbind(tableFiles, tableCases)
 ###---------------WINDOW
 ca <- gtkWindow(show=FALSE)
 ca$setTitle("RQDAExt - Coding Analysis")
 ca$setDefaultSize(gdkScreenWidth()*2/3,gdkScreenHeight()*2/3)
 ca["border-width"] <- 10
 icon <- system.file("icon", "mainIcon.png", package = "RQDAExt")
 image <- gdkPixbufNewFromFile(icon)
 ca$setIcon(image$retval)
 ###---------------COMPONENTS
 ca.boxes <- list()
```

```
ca.boxes$codes <- gtkComboBoxNewText()
ca.boxes$files <- gtkComboBoxNewText()
ca.boxes$logical <- gtkComboBoxNewText()
sapply(c("Codes", "Code Categories"), ca.boxes$codes$appendText)
sapply(c("Files","File Categories","Cases"), ca.boxes$files$appendText)
sapply(c("AND","OR"), ca.boxes$logical$appendText)
ca.boxes$logical["active"] <- 0
#
ca.selectall <- list()
ca.selectall $button1 <- gtkButton("Select All")
ca.selectall $button2 <- gtkButton("Select All")
#
ca.general <- list()
ca.general$close <- gtkButton("Close")
ca.general$help <- gtkButton("Help")
#
ca.text <-list()
ca.text$clear <- gtkButton("Clear")
ca.text$txt <- gtkButton("Export in txt")
ca.text$print <- gtkButton("Print")
#
ca.view <- gtkTextView()
ca.scroll <- gtkScrolledWindow(hadjustment = ca.view)
ca.scroll$setPolicy("never","automatic")
ca.scroll$add(ca.view)
ca.buffer <- ca.view$getBuffer()
ca.view["editable"] <- FALSE
ca.view["wrap-mode"] <- "word-char"
ca.view["left-margin"] <- 5
ca.view["right-margin"] <- 5
gtkWidgetSetSizeRequest(ca.view, 100, 300)
ca.buffer$createTag(tag.name = "filename", weight = PangoWeight["bold"])
ca.buffer$createTag(tag.name = "position", foreground = "red")
#
ca.models <- list()
ca.models$codes <- rGtkDataFrame(DFs$codes)
ca.models$files <- rGtkDataFrame(DFs$files)
#
ca.filteredmodels <-list()
ca.filteredmodels$codes <- ca.models$codes$filter()
ca.filteredmodels$codes$setVisibleColumn(ncol(DFs$codes)-1)
ca.filteredmodels$files <- ca.models$files$filter()
ca.filteredmodels$files$setVisibleColumn(ncol(DFs$files)-1)
#
ca.views <- list()
ca.views$codes <- gtkTreeView(ca.filteredmodels$codes)
ca.views$files <- gtkTreeView(ca.filteredmodels$files)
sapply (ca.views, function (view){
  view$insertColumnWithAttributes (0, "Select names", gtkCellRendererText(), text = 1)
})
sapply (ca.views, function (view){
  selection <- view$getSelection()
  selection$setMode("multiple")
})
#
ca.scrolls <- list()
ca.scrolls$codes <- gtkScrolledWindow()
ca.scrolls$files <- gtkScrolledWindow()
mapply (gtkContainerAdd, object = ca.scrolls, widget = ca.views)
mapply (gtkScrolledWindowSetPolicy, ca.scrolls, "automatic", "automatic")
###---------------CONTAINERS
ca.container <- gtkHBox(TRUE, 10)
ca.frame1 <- gtkFrame ("Filtering System")
```

```
ca.frame2 <- gtkFrame ("Data Display")
ca.vbox1 <- gtkVBox(FALSE, 10)
ca.vbox2 <- gtkVBox(FALSE, 10)
ca.grid <- gtkTable(rows = 5, columns = 2, homogeneous = FALSE)
ca.hbox1 <- gtkHBox(FALSE, 5)
ca.hbox2 <- gtkHBox(FALSE, 5)
###----------LAYOUT
ca$add(ca.container)
ca.container$packStart(ca.vbox1)
ca.container$packStart(ca.frame2)
#
ca.vbox1$packStart(ca.frame1, expand = TRUE, fill = TRUE)
ca.vbox1$packStart(ca.hbox1, expand=FALSE, fill = FALSE)
#
ca.frame1$add(ca.grid)
ca.grid$attach(gtkLabel("Codes"), left.attach =0,1 , top.attach = 0,1,
               xoptions = c("expand", "fill"), yoptions = "")
ca.grid$attach(gtkLabel("Files"), left.attach = 1,2, top.attach = 0,1,
               xoptions = c("expand", "fill"), yoptions = "")
ca.grid$attach(ca.boxes$codes, left.attach = 0,1, top.attach = 1,2,
               xoptions = c("expand", "fill"), yoptions = "")
ca.grid$attach(ca.boxes$files, left.attach = 1,2, top.attach =1,2 ,
               xoptions = c("expand", "fill"), yoptions = "")
ca.grid$attach(ca.scrolls$codes, left.attach = 0,1, top.attach = 2,3,
               xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
ca.grid$attach(ca.scrolls$files, left.attach = 1,2, top.attach = 2,3,
               xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
ca.grid$attach(ca.selectall$button1, left.attach = 0,1, top.attach = 3,4,
               xoptions = c("expand", "fill"), yoptions = "")
ca.grid$attach(ca.selectall$button2, left.attach = 1,2, top.attach = 3,4,
               xoptions = c("expand", "fill"), yoptions = "")
ca.grid$attach(gtkLabel("Logical conjunction:"), left.attach = 0,1, top.attach = 4,5,
               xoptions = c("expand", "fill"), yoptions = "")
ca.grid$attach(ca.boxes$logical, left.attach = 1,2, top.attach = 4,5,
               xoptions = c("expand", "fill"), yoptions = "")
ca.grid$setColSpacing(0,5)
ca.grid$setRowSpacing(1,5)
ca.grid$setRowSpacing(2,5)
ca.grid$setRowSpacing(3,5)
ca.grid$setRowSpacing(4,5)
#
sapply(ca.general, ca.hbox1$packStart, expand = TRUE)
ca.hbox1$packStart(gtkLabel(""),expand=TRUE, fill = TRUE)
#
ca.frame2$add(ca.vbox2)
ca.vbox2$packStart(ca.scroll, expand = TRUE, fill = TRUE)
ca.vbox2$packStart(ca.hbox2, expand =FALSE)
#
ca.hbox2$packStart(gtkLabel(""), expand = TRUE, fill = TRUE)
sapply(ca.text, ca.hbox2$packStart, expand = TRUE)
###----------SIGNAL CONNECTIONS
gSignalConnect(ca.boxes$codes, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF[,"type"]
  DF[,"visible"] <- grepl(pattern, values)
}, data= ca.filteredmodels$codes)
#
gSignalConnect(ca.boxes$files, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF[,"type"]
  DF[,"visible"] <- grepl(pattern, values)
```

```
}, data= ca.filteredmodels$files)
gSignalConnect(ca.selectall$button1, "clicked", function (button){
  ca.views$codes$getSelection()$SelectAll()
})
gSignalConnect(ca.selectall$button2, "clicked", function (button){
  ca.views$files$getSelection()$SelectAll()
})
#
gSignalConnect(ca.general$close, "clicked", function(button){
  ca$destroy()
})
gSignalConnect(ca.general$help, "clicked", function(button){
  InstructionsWindow(active=4)
})
#
gSignalConnect(ca.text$print, "clicked", function(button){
  CBoxValues <- list(ca.boxes$codes$getActiveText(),
                     ca.boxes$files$getActiveText(),
                     ca.boxes$logical$getActiveText())
  ind1 <- ca.views$codes$selectedIndices()
  ind2 <- ca.views$files$selectedIndices()
  if(ind1==0 || ind2==0){
    dialog <- gtkMessageDialog(parent = ca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  } else {
    CodeValues <- ca.models$codes[ind1,1]
    FileValues <- ca.models$files[ind2,1]
    codings <- getCodings(filter= c(CBoxValues[[1]],CBoxValues[[2]]),
                          cid = CodeValues, fid = FileValues,
                          condition = CBoxValues[[3]])
    ###Print the text in the text buffer
    ca.buffer$setText("")
    TextBounds<- ca.buffer$getBounds()
    maininfo <- paste(nrow(codings), "CODINGS FROM", length(c(table(codings$fid))) ,"FILES")
    ca.buffer$insertWithTagsByName(TextBounds$end, maininfo, "filename")
    ca.buffer$insert(TextBounds$end, "\n\n")
    if(nrow(codings)==0){
      ca.buffer$insert(TextBounds$end, "No codings")
      dialog <- gtkMessageDialog(parent = ca,
                                 flags = "destroy-with-parent",
                                 type = "warning", button = "ok",
                                 "No codings")
      response <- dialog$run()
      if (response == GtkResponseType["ok"]){
        dialog$destroy()}
    } else {
      for (i in (1:nrow(codings))){
        ca.buffer$insertWithTagsByName(TextBounds$end, codings$filename[i], "filename")
        ca.buffer$insert(TextBounds$end, " ")
        ca.buffer$insertWithTagsByName(TextBounds$end, "[", "position")
        ca.buffer$insertWithTagsByName(TextBounds$end, codings$index1[i], "position")
        ca.buffer$insertWithTagsByName(TextBounds$end, ":", "position")
        ca.buffer$insertWithTagsByName(TextBounds$end, codings$index2[i],"position")
        ca.buffer$insertWithTagsByName(TextBounds$end, "]\n\n", "position")
        ca.buffer$insert(TextBounds$end, codings$coding[i])
        ca.buffer$insert(TextBounds$end, "\n\n")
      }
    }
```

```
  }
 })
 gSignalConnect(ca.text$clear,"clicked",function(clear){
  ca.buffer$setText("")
 })
 gSignalConnect(ca.text$txt,"clicked",function(txt){
  gtkTextViewExportBuffer(ca.buffer,"Codings")
 })
 ###---------------END
 ca$show()
 ca$move(gdkScreenWidth()*1/3,gdkScreenHeight()*1/4)
}
```

- **General_Functions.R**

```
#----------------------------GENERAL FUNCTIONS-----------------------------#
#
###Get the Codes Filtering by Code Categories
#
getCodesbyCodeCat <- function(x){
 if (is.numeric(x)){ #if it is a vector of ids
  sql <- paste("catid =",x, collapse = " or ")
  sql <- paste("select cid from treecode where status=1 and (", sql,
               ") order by cid")
  sql <- RQDAQuery(sql)[[1]]
  sql <- paste("id =",sql, collapse = " or ")
  sql <- paste("select id,name from freecode where status=1 and (",sql, ")")
  code_list <- RQDAQuery(sql)
 }
 if (is.character(x)){  #if it is a vector of names
  sql <- paste("'",x,"'", sep="")
  sql <- paste("name =",sql, collapse = " or ")
  sql <- paste("select catid from codecat where status=1 and (", sql,")")
  sql <- RQDAQuery(sql)[[1]]
  sql <- paste("catid =",sql, collapse = " or ")
  sql <- paste("select cid from treecode where status=1 and (", sql,
               ") order by cid")
  sql <- RQDAQuery(sql)[[1]]
  sql <- paste("id =",sql, collapse = " or ")
  sql <- paste("select id,name from freecode where status=1 and (",sql, ")")
  code_list <- RQDAQuery(sql)
 }
 code_list
} #end of the function
#
####Get the Files Filtering by File Categories
getFilesbyFileCat <- function(x){
 if (is.numeric(x)){ #if it is a vector of ids
  sql <- paste("catid =",x, collapse = " or ")
  sql <- paste("select fid from treefile where status=1 and (", sql,
               ") order by fid")
  sql <- RQDAQuery(sql)[[1]]
  sql <- paste("id =",sql, collapse = " or ")
  sql <- paste("select id,name from source where status= 1 and (",sql,")")
  file_list <- RQDAQuery(sql)
 }
 if (is.character(x)){ #if it is a vector of names
  sql <- paste("'",x,"'", sep="")
  sql <- paste("name =",sql, collapse = " or ")
  sql <- paste("select catid from filecat where status=1 and (", sql,")")
  sql <- RQDAQuery(sql)[[1]]
  sql <- paste("catid =",sql, collapse = " or ")
  sql <- paste("select fid from treefile where status=1 and (", sql,
```

```
                ") order by fid")
    sql <- RQDAQuery(sql)[[1]]
    sql <- paste("id =",sql, collapse = " or ")
    sql <- paste("select id,name from source where status= 1 and (",sql,")")
    file_list <- RQDAQuery(sql)
  }
  file_list
}#end of the function
#
####Get the Files Filtering by Cases
getFilesbyCases <- function(x){
  if (is.numeric(x)){   #if it is a vector of ids
    sql <- paste("caseid =",x, collapse = " or ")
    sql <- paste("select fid from caselinkage where status=1 and (", sql,
                 ") order by fid ")
    sql <- RQDAQuery(sql)[[1]]
    sql <- paste("id =",sql, collapse = " or ")
    sql <- paste("select id,name from source where status=1 and (",sql, ")")
    file_list <- RQDAQuery(sql)
  }
  if (is.character(x)){  #if it is a vector of names
    sql <- paste("'",x,"'", sep="")
    sql <- paste("name =",sql, collapse = " or ")
    sql <- paste("select id from cases where status=1 and (", sql,")")
    sql <- RQDAQuery(sql)[[1]]
    sql <- paste("caseid =",sql, collapse = " or ")
    sql <- paste("select fid from caselinkage where status=1 and (", sql,
                 ") order by fid ")
    sql <- RQDAQuery(sql)[[1]]
    sql <- paste("id =",sql, collapse = " or ")
    sql <- paste("select id,name from source where status=1 and (",sql, ")")
    file_list <- RQDAQuery(sql)
  }
  file_list
}#end of the function
#
####Make a Code vs Code Matrix
CodeCodeMatrix <- function (filter, rowcid, colcid, fid, relation=
                            c("overlap", "inclusion", "exact", "proximity"), as.matrix=TRUE){
  #Find the ids of the row codes
  if (filter[1]=="Codes"){
    row_codes <- rowcid
  }
  if (filter[1]=="Code Categories"){
    row_codes <- getCodesbyCodeCat(rowcid)
    row_codes <- row_codes$id
  }
  #Find the ids of the column codes
  if (filter[2]=="Codes"){
    col_codes <- colcid
  }
  if (filter[2]=="Code Categories"){
    col_codes <- getCodesbyCodeCat(colcid)
    col_codes <- col_codes$id
  }
  #Find the ids of the files
  if (filter[3]=="Files"){
    files <- fid
  }
  if (filter[3]=="File Categories"){
    files <- getFilesbyFileCat(fid)
    files<- files$id
  }
```

```
 if (filter[3]=="Cases"){
   files <- getFilesbyCases(fid)
   files <- files$id
 }
 #Create the table of files
 allTable <- getCodingTable()
 old_table <- data.frame()
 for (i in (1:length(files))){
   new_table <- subset(allTable, fid==files[i])
   dataTable <- rbind(old_table, new_table)
   old_table <- dataTable
 }
 #Defintion of the column and row names
 sql <- paste ("id =", row_codes, collapse= " or ")
 sql <- paste("select name from freecode where", sql)
 row_names <- RQDAQuery(sql)[[1]]
 sql <- paste ("id =", col_codes, collapse= " or ")
 sql <- paste("select name from freecode where", sql)
 column_names <- c("",RQDAQuery(sql)[[1]])
 column_names2 <- RQDAQuery(sql)[[1]]
 #Create the Matrix as data frame.This option provides a good look for the GUI
 if(as.matrix==FALSE){
   Matrix <- array(dim=c(length(row_codes),length(col_codes)+1),
                   dimnames = list(NULL, column_names))
   Matrix[,1]<- row_names #Move the row names to the first column
   for (i in (1:length(row_codes))){
     for (j in (1:length(col_codes))){
       Matrix[i,j+1] <- crossTwoCodes(row_codes[i], col_codes[j],
                                      data=dataTable, relation=relation)
     }
   }
   Matrix<- as.data.frame(Matrix)
 }
 #Create a Matrix as an array
 if(as.matrix==TRUE){
   Matrix <- array(dim=c(length(row_codes),length(col_codes)),
                   dimnames = list(row_names, column_names2))
   for (i in (1:length(row_codes))){
     for (j in (1:length(col_codes))){
       Matrix[i,j] <- crossTwoCodes(row_codes[i], col_codes[j],
                                    data=dataTable, relation=relation)
     }
   }
 }
 Matrix
}# end of the function
#
###Make a Code vs File Matrix
CodeFileMatrix <- function(filter,cid, fid, as.matrix=TRUE){
 #Find the ids of the codes
 if (filter[1]=="Codes"){
   codes <- cid
 }
 if (filter[1]=="Code Categories"){
   codes <- getCodesbyCodeCat(cid)
   codes <- codes$id
 }
 #Find the  ids of the  files
 if (filter[2]=="Files"){
   files <-  fid
 }
 if (filter[2]=="File Categories"){
   files <- getFilesbyFileCat(fid)
```

```
  files <- files$id
 }
 if (filter[2]=="Cases"){
  files <- getFilesbyCases(fid)
  files <- files$id
 }
 #Definition of the column and row names
 sql <- paste ("id =", codes, collapse= " or ")
 sql <- paste("select name from freecode where", sql)
 row_names <- RQDAQuery(sql)[[1]]
 sql <- paste ("id =", files, collapse= " or ")
 sql <- paste("select name from source where", sql)
 column_names <- c("",RQDAQuery(sql)[[1]])
 column_names2 <- RQDAQuery(sql)[[1]]
 #Create the Matrix as data frame.This option provides a good look for the GUI
 if(as.matrix==FALSE){
  Matrix <- array(dim=c(length(codes),length(files)+1),
                  dimnames = list(NULL, column_names))
  Matrix[,1]<- row_names #Move the row names to the first column
  for (i in (1:length(codes))){
   for (j in (1:length(files))){
    Matrix[i,j+1] <- nrow(getCodingsByOne(codes[i], files[j]))
   }
  }
  as.data.frame(Matrix)
 }
 #Create a Matrix as an array
 if(as.matrix==TRUE){
  Matrix <- array(dim=c(length(codes),length(files)),
                  dimnames = list(row_names, column_names2))
  for (i in (1:length(codes))){
   for (j in (1:length(files))){
    Matrix[i,j] <- nrow(getCodingsByOne(codes[i], files[j]))
   }
  }
 }
 Matrix
}#end of the function
#
###Get the codings with AND or OR
getCodings<- function(filter, cid, fid, condition=c("AND", "OR")){
 #Find the ids of the codes
 if(filter[1]=="Codes"){
  codes <- cid
 }
 if(filter[1]=="Code Categories"){
  codes <- getCodesbyCodeCat(cid)
  codes <- codes$id
 }
 #Find the ids of the files
 if(filter[2]=="Files"){
  files <- fid
 }
 if(filter[2]=="File Categories"){
  files <- getFilesbyFileCat(fid)
  files <- files$id
 }
 if(filter[2]=="Cases"){
  files <- getFilesbyCases(fid)
  files <- files$id
 }
 codings <- getCodingsByOne(codes[1], files) #First coding
 #Search the rest of the codings if they exsist
```

```
  if(length(codes)>1){
    for (i in (2:length(codes))){
      #Codings with AND condition
      if (condition=="AND"){
        codings <- and(codings,getCodingsByOne(codes[i], files))
      }
      #Codings with OR condition
      if(condition=="OR"){
        codings2 <- getCodingsByOne(codes[i], files)
        if(nrow(codings2)>0){
          codings <- or(codings,codings2)
        }
      }
    }# end for
  }# end if length(codes>1)
  codings
}# end of the function
#
###Get Attributes by File/Case
getAttributesbyFile <- function(filter=c("Files", "Cases"), fid){
  #Find the attributes using the filter
  if (filter=="Files"){
    sql <- paste("select variable, value from fileAttr where status = 1  and fileID =",
                 fid)
  }
  if (filter=="Cases"){
    sql <- paste("select variable, value from caseAttr where status = 1  and caseID =",
                 fid)
  }
  #Final query
  attributes <- RQDAQuery(sql)
  attributes
}#end of the function
#
###Get Files/Cases by Attributes
getFilesbyAttribute <- function(filter=c("Files", "Cases"), attribute){
  aux_paste <- paste("'",attribute,"'",sep="")
  summary <- list()
  #Find the attributes of the files
  if(filter=="Files"){
    sql <- paste("select distinct value from fileAttr where status = 1  and variable =",
                 aux_paste)
    values <- RQDAQuery(sql) #Take all the values of the attribute
    if (nrow(values)){
      #Find all the ids of the files for each values
      for (i in (1:nrow(values))){
        aux_paste2 <- paste("'", values$value[i] ,"'",sep="")
        aux_paste2 <- paste("select fileID from fileAttr where status = 1  and value =",
                            aux_paste2)
        sql <- paste(aux_paste2, " and variable=", aux_paste, " order by fileID")
        files <- RQDAQuery(sql)#ids of the files
        for (j in (1:nrow(files))){
          sql <- paste("select name from source where status=1 and id=",
                       files$fileID[j])
          files$name[j] <- RQDAQuery(sql)#names of the files
        }
        summary[[values$value[i]]] <- files#Add the files in each value
      }
    } else {print("No files with this attribute")}
  }
  #Find the attributes of the cases
  if(filter=="Cases"){
    sql <- paste("select distinct value from caseAttr where status = 1  and variable =",
```

```
            aux_paste)
  values <- RQDAQuery(sql) #Take all the values of the attribute
  if (nrow(values)){
    #Find all the ids of the cases for each values
    for (i in (1:nrow(values))){
      aux_paste2 <- paste("'", values$value[i] ,"'",sep="")
      aux_paste2 <- paste("select caseID from caseAttr where status = 1  and value =",
                          aux_paste2)
      sql <- paste(aux_paste2, " and variable=", aux_paste, " order by caseID")
      cases <- RQDAQuery(sql)#ids of the cases
      for (j in (1:nrow(cases))){
        sql <- paste("select name from cases where status=1 and id=",
                     cases$caseID[j])
        cases$name[j] <- RQDAQuery(sql)#names of the cases
      }
      summary[[values$value[i]]] <- cases#Add the cases in each value
    }
  } else {print("No cases with this attribute")}
 }
 summary
}# end of the function
#
###Make a summary of codings
SummaryOfCodings <- function(filter, cid, fid, variable=
                             c("Number of Codings", "Average Length",
                               "Number of Files", "Codings for Each File")){
 #Find the ids of the codes
 if (filter[1]=="Codes"){
   codes <- cid
 }
 if (filter[1]=="Code Categories"){
   codes <- getCodesbyCodeCat(cid)
   codes <- codes$id
 }
 #Find the ids of the files
 if (filter[2]=="Files"){
   files <-  fid
 }
 if (filter[2]=="File Categories"){
   files <- getFilesbyFileCat(fid)
   files <- files$id
 }
 if (filter[2]=="Cases"){
   files <- getFilesbyCases(fid)
   files <- files$id
 }
 #Auxiliary variables for the queries
 aux_paste <- paste("fid=", files, collapse =" or ")
 aux_paste <- paste0("and (", aux_paste,")")
 aux_paste2 <- paste("cid=", codes, sep="")
 aux_paste3 <- paste("fid=", files, sep="")
 aux_paste4 <- paste("cid=", codes, collapse =" or ")
 aux_paste4 <- paste0("and (", aux_paste4,")")
 #Find the number of codings in the files
 if (variable=="Number of Codings"){
   value <- c()
   value2 <- c()
   sql <- c()
   for (i in 1:length(codes)){
     sql[i] <- paste("select count(cid) from coding where status=1 and",
                     aux_paste2[i],aux_paste)
     value2[i] <- RQDAQuery(sql[i])
     value[i] <- value2[[i]]
```

```
  }
}# end if "Number of Codings"
#Find the average length of the codings
if (variable=="Average Length"){
  value <- c()
  minim <- c()
  maxim <- c()
  total <- c()
  for (i in 1:length(codes)){
    sql <- paste("select sum(selfirst) from coding where status=1 and",
                 aux_paste2[i],aux_paste)
    minim[i] <- RQDAQuery(sql)
    sql <- paste("select sum(selend) from coding where status=1 and",
                 aux_paste2[i],aux_paste)
    maxim[i] <- RQDAQuery(sql)
    sql <- paste("select count(selfirst) from coding where status=1 and",
                 aux_paste2[i],aux_paste)
    total[i]<- RQDAQuery(sql)
    if(total[i]==0){
      value[i] <- 0
    } else {
      value[i] <- (maxim[[i]]-minim[[i]])/total[[i]]
    }
  }
}# end if "Average length"
#Find the number of files with the codings
if (variable=="Number of Files"){
  value <- c()
  value2 <- c()
  sql <- c()
  for (i in 1:length(codes)){
    sql[i] <- paste("select count(distinct(fid)) from coding where status=1 and",
                    aux_paste2[i],aux_paste)
    value2[i] <- RQDAQuery(sql[i])
    value[i] <- value2[[i]]
  }
}# end if "Number of Files"
#Find the codings for each file
if (variable=="Codings for Each File"){
  value <- c()
  value2 <- c()
  sql <- c()
  for (i in 1:length(files)){
    sql[i] <- paste("select count(cid) from coding where status=1 and",
                    aux_paste3[i],aux_paste4)
    value2[i] <- RQDAQuery(sql[i])
    value[i] <- value2[[i]]
  }
}# end if "Coding for Each File"
#Create the data frame with the results
if(variable=="Number of Codings" || variable== "Average Length" ||
   variable == "Number of Files"){ #Option 1,2,3
  #Find the name of the codings and the files
  sql <- paste ("id =", codes, collapse= " or ")
  sql <- paste("select name from freecode where", sql)
  code_names <- RQDAQuery(sql)[[1]]
  summary <- data.frame(coding=code_names, value=value)
}
if(variable=="Codings for Each File"){ #Option 4
  sql <- paste ("id =", files, collapse= " or ")
  sql <- paste("select name from source where", sql)
  file_names <- RQDAQuery(sql)[[1]]
  summary <- data.frame(coding=file_names, value=value)
```

```
  }
  summary
}#end of the function
#
###Make a Links and Nodes for the Networks with a matrix of concurrences
getLinksAndNodes <- function(matrix,type){
  #Nodes, If the given matrix is a Code vs Code Matrix
  if(type==1){
    rows <- dimnames(matrix)[[1]]
    columns<- dimnames(matrix)[[2]]
    codes <- c(rows, columns)#Take the nodes of row and column codes
    codes <- codes[!duplicated(codes)]#Delete the duplicateds
    sql <- paste("'",codes,"'", sep="")
    sql <- paste("name =",sql, collapse = " or ")
    cids <- RQDAQuery(paste("select id from freecode where status=1 and (",sql, ")"))
    codecat<-c()
    for(i in (1:length(codes))){#Find the code categories of the codes
      sql <- paste("cid=",cids$id[i], collapse = " or ")
      sql <- RQDAQuery(paste("select catid from treecode where status=1 and ",sql))
      if(nrow(sql)==0) codecat[i] <-NA else codecat[i]<-sql$catid
    }
    #Definition of the nodes
    nodes <- data.frame(name= codes, id=cids$id, catid=codecat)
  }
  #Nodes, If the given matrix is a Code vs File Matrix
  if(type==2){
    codes <- dimnames(matrix)[[1]]
    files<- dimnames(matrix)[[2]]#Take the nodes of row and column codes
    sql <- paste("'",codes,"'", sep="")
    sql <- paste("name =",sql, collapse = " or ")
    cids <- RQDAQuery(paste("select id from freecode where status=1 and (",sql, ")"))
    sql <- paste("'",files,"'", sep="")
    sql <- paste("name =",sql, collapse = " or ")
    fids <- RQDAQuery(paste("select id from source where status=1 and (",sql, ")"))
    codecat<-c()
    filecat <-c()
    cases <- c()
    for(i in (1:length(codes))){#Find the code categories of the codes
      sql <- paste("cid=",cids$id[i], collapse = " or ")
      sql <- RQDAQuery(paste("select catid from treecode where status=1 and ",sql))
      if(nrow(sql)==0) codecat[i] <-NA else codecat[i]<-sql$catid
    }
    for(i in (1:length(files))){#Find the file categories of the files
      sql <- paste("fid=",fids$id[i], collapse = " or ")
      sql <- RQDAQuery(paste("select catid from treefile where status=1 and ",sql))
      if(nrow(sql)==0) filecat[i] <-NA else filecat[i] <- sql$catid
    }
    for(i in (1:length(files))){#Find the cases of the files
      sql <- paste("fid=",fids$id[i], collapse = " or ")
      sql<- RQDAQuery(paste("select caseid from caselinkage where status=1 and ",sql))
      if(nrow(sql)==0) cases[i] <-NA else cases[i]<- sql$caseid
    }
    #Definition of the nodes
    nodes <-data.frame(name= c(codes,files), id=c(cids$id,fids$id),
                       type=c(rep(1, length(codes)),rep(2, length(files))),
                       catid=c(codecat, rep(NA,length(files))),
                       fileid=c(rep(NA,length(codes)), filecat),
                       cases=c(rep(NA,length(codes)), cases))
  }
  if(type!=1 && type!=2){
    stop("Invalid type. Select type 1 for a CodeCodeMatrix and 2
         for a CodeFileMatrix")#Error message
  }
```

```
#Links
origin <- c()
end <- c()
value <- c()
count <- 1
for (i in (1:nrow(matrix))){
  for (j in (1:ncol(matrix))){
    if(matrix[i,j]!=0){#only if we have a value !=0
      origin[count] <- dimnames(matrix)[[1]][i]
      end[count]<- dimnames(matrix)[[2]][j]
      value[count]<- matrix[i,j]
      count <- count+1
    }
  }
}
#Definiton of the links
links <- data.frame(from=origin, to=end, weight=value)
links.sort <- t(apply(links, 1, sort))
links <- links[!duplicated(links.sort),]
#List with the nodes and the links
elements <- list(nodes=nodes, links=links)
}
```

- **InstructionsWindow.R**

```
###----------INSTRUCTIONS WINDOW----------###
InstructionsWindow <- function(active=c(1:6)){
  ###----------WINDOW----------###
  instr <- gtkWindow(show=FALSE)
  instr$setTitle("RQDAExt: Help")
  instr$setDefaultSize(gdkScreenWidth()*1/3,gdkScreenHeight()*1/2)
  instr["border-width"] <- 15
  icon <- system.file("icon", "mainIcon.png", package = "RQDAExt")
  image <- gdkPixbufNewFromFile(icon)
  instr$setIcon(image$retval)
  ###----------COMPONENTS----------###
  instr.text <- gtkTextView()
  instr.scroll <- gtkScrolledWindow(hadjustment = instr.text)
  instr.scroll$setPolicy("never","automatic")
  instr.scroll$add(instr.text)
  instr.text["editable"] <- FALSE
  instr.text["wrap-mode"] <- "word"
  instr.text["left-margin"] <- 10
  instr.text["right-margin"] <- 10
  instr$add(instr.scroll)
  gtkWidgetSetSizeRequest(instr.text, 50, 50)
  ###----------TEXT----------###
  instr.buffer <- instr.text$getBuffer()
  instr.bounds<- instr.buffer$getBounds()
  if(active==1){text <- readLines(system.file("extdata", "help1.txt", package = "RQDAExt"), n=-1,
skipNul=FALSE)}
  if(active==2){text <- readLines(system.file("extdata", "help2.txt", package = "RQDAExt"), n=-1,
skipNul=FALSE)}
  if(active==3){text <- readLines(system.file("extdata", "help3.txt", package = "RQDAExt"), n=-1,
skipNul=FALSE)}
  if(active==4){text <- readLines(system.file("extdata", "help4.txt", package = "RQDAExt"), n=-1,
skipNul=FALSE)}
  if(active==5){text <- readLines(system.file("extdata", "help5.txt", package = "RQDAExt"), n=-1,
skipNul=FALSE)}
  if(active==6){text <- readLines(system.file("extdata", "help6.txt", package = "RQDAExt"), n=-1,
skipNul=FALSE)}
  for(i in (1:length(text))){
    instr.buffer$insert(instr.bounds$end,text[i])
```

```
  instr.buffer$insert(instr.bounds$end,"\n")
 }
 ###---------------END----------------------###
 instr$show()
 instr$move(gdkScreenWidth()*1/2,gdkScreenHeight()*1/4)
}
```

- **RQDAExt.R**

```
#------------------------------RQDAExt PROGRAM-------------------------------#
RQDAExt <- function(){
 ###---------------TOP LEVEL WINDOW-----------------###
 window <- gtkWindow(show=FALSE)
 window$setTitle("RQDAExt: Qualitative Data Analysis")
 window$setDefaultSize(gdkScreenWidth()*2/3,gdkScreenHeight()*1/3)
 window["border-width"] <- 15
 icon <- system.file("icon", "mainIcon.png", package = "RQDAExt")
 image <- gdkPixbufNewFromFile(icon)
 window$setIcon(image$retval)
 ###---------------COMPONENTS
 menu.buttons <- list()
 menu.buttons$open <- gtkButton("Open Project")
 menu.buttons$close <- gtkButton("Close Project")
 menu.buttons$codecode <- gtkButton("Code Vs Code Analysis")
 menu.buttons$codefile <- gtkButton("Code Vs File Analysis")
 menu.buttons$coding <- gtkButton("Coding Analysis")
 menu.buttons$summary <- gtkButton("Summary of Codings Analysis")
 menu.buttons$attributes <- gtkButton("Attribute Analysis")
 menu.buttons$help <- gtkButton("Help")
 menu.buttons$close$setSensitive(FALSE)
 menu.buttons$codecode$setSensitive(FALSE)
 menu.buttons$codefile$setSensitive(FALSE)
 menu.buttons$coding$setSensitive(FALSE)
 menu.buttons$summary$setSensitive(FALSE)
 menu.buttons$attributes$setSensitive(FALSE)
 menu.text <- gtkTextView()
 menu.text["editable"] <- FALSE
 menu.text["wrap-mode"] <- "word-char"
 menu.text["justification"] <- "center"
 menu.text["cursor-visible"] <- FALSE
 gtkWidgetSetSizeRequest(menu.text, 50, 50)
 menu.buffer <- menu.text$getBuffer()
 menu.bounds<- menu.buffer$getBounds()
 menu.buffer$insert(menu.bounds$end,"\n")
 menu.buffer$insert(menu.bounds$end,"Path of current project:\n")
 menu.buffer$insert(menu.bounds$end,"No project is open\n")

menu.buffer$insert(menu.bounds$end,"_____________________________________________
_____________________\n\n")
 menu.buffer$insert(menu.bounds$end,"Author: Gerard Pueyo\n")
 menu.buffer$insert(menu.bounds$end,"<gerard.pueyo.v@gmail.com>\n")
 menu.buffer$insert(menu.bounds$end,"Designer: Vicenc Fernandez\n")

menu.buffer$insert(menu.bounds$end,"_____________________________________________
_____________________\n\n")
 menu.buffer$insert(menu.bounds$end,"License: BSD\n")
 menu.buffer$insert(menu.bounds$end,"Version: 0.1  Year: 2014\n")
 ###---------------CONTAINERS
 menu <-gtkHBox(FALSE,5)
 menu.group <- gtkVBox()
 menu.group2 <- gtkVBox()
 ###---------------LAYOUT
 menu$packStart(menu.group)
```

```
  menu$packStart(menu.group2)
  sapply(menu.buttons, menu.group$packStart, expand=TRUE)
  menu.group2$packStart(menu.text, expand=TRUE)
  window$add(menu)
  ###----------SIGNAL CONNECTIONS
  gSignalConnect(menu.buttons$open,"clicked", function(button,user.data){
    path <- gfile("Open a .rqda file", type="open",filter=c(".rqda files"="rqda"))
    openProject(path, updateGUI = TRUE)
    menu.buttons$close$setSensitive(TRUE)
    menu.buttons$codecode$setSensitive(TRUE)
    menu.buttons$codefile$setSensitive(TRUE)
    menu.buttons$coding$setSensitive(TRUE)
    menu.buttons$summary$setSensitive(TRUE)
    menu.buttons$attributes$setSensitive(TRUE)
    menu.buffer$setText("")
    menu.bounds<- menu.buffer$getBounds()
    menu.buffer$insert(menu.bounds$end,"\n")
    menu.buffer$insert(menu.bounds$end,"Path of current project:\n")
    menu.buffer$insert(menu.bounds$end, path)
    menu.buffer$insert(menu.bounds$end,"\n")

menu.buffer$insert(menu.bounds$end,"_____________________________________________
_____________________\n\n")
    menu.buffer$insert(menu.bounds$end,"Author: Gerard Pueyo\n")
    menu.buffer$insert(menu.bounds$end,"<gerard.pueyo.v@gmail.com>\n")
    menu.buffer$insert(menu.bounds$end,"Designer: Vicenc Fernandez\n")

menu.buffer$insert(menu.bounds$end,"_____________________________________________
_____________________\n\n")
    menu.buffer$insert(menu.bounds$end,"License: BSD\n")
    menu.buffer$insert(menu.bounds$end,"Version: 0.1  Year: 2014\n")
  })
  gSignalConnect(menu.buttons$close,"clicked", function(button){
    closeProject()
    menu.buttons$close$setSensitive(FALSE)
    menu.buttons$codecode$setSensitive(FALSE)
    menu.buttons$codefile$setSensitive(FALSE)
    menu.buttons$coding$setSensitive(FALSE)
    menu.buttons$summary$setSensitive(FALSE)
    menu.buttons$attributes$setSensitive(FALSE)
    menu.buffer$setText("")
    menu.bounds<- menu.buffer$getBounds()
    menu.buffer$insert(menu.bounds$end,"\n")
    menu.buffer$insert(menu.bounds$end,"Path of current project:\n")
    menu.buffer$insert(menu.bounds$end,"No project is open\n")

menu.buffer$insert(menu.bounds$end,"_____________________________________________
_____________________\n\n")
    menu.buffer$insert(menu.bounds$end,"Author: Gerard Pueyo\n")
    menu.buffer$insert(menu.bounds$end,"<gerard.pueyo.v@gmail.com>\n")
    menu.buffer$insert(menu.bounds$end,"Designer: Vicenc Fernandez\n")

menu.buffer$insert(menu.bounds$end,"_____________________________________________
_____________________\n\n")
    menu.buffer$insert(menu.bounds$end,"License: BSD\n")
    menu.buffer$insert(menu.bounds$end,"Version: 0.1  Year: 2014\n")
  })
  gSignalConnect(menu.buttons$codecode,"clicked", function(button){
    CodeCodeWindow()
  })
  gSignalConnect(menu.buttons$codefile,"clicked", function(button){
    CodeFileWindow()
  })
```

```
 gSignalConnect(menu.buttons$coding,"clicked", function(button){
   CodingsWindow()
 })
 gSignalConnect(menu.buttons$summary,"clicked", function(button){
   SummaryCodingsWindow()
 })
 gSignalConnect(menu.buttons$attributes,"clicked", function(button){
   AttributesWindow()
 })
 gSignalConnect(menu.buttons$help,"clicked", function(button){
   InstructionsWindow(active=1)
 })

 ###---------------------------END----------------------------###
 window$show()
 window$move(gdkScreenWidth()*1/3,0)
 gSignalConnect (window, "delete-event", function (event , ...){
   dialog <- gtkMessageDialog(parent = window, flags = 0,
                              type = "question",
                              buttons = "yes-no",
                              "Are you sure you want to quit?")
   out <- dialog$run(); dialog$destroy()
   out != GtkResponseType["yes"]
 })

}
```

- **SummaryCodingsWindow.R**

```
###---------------SUMMARY OF CODINGS ANALYSIS----------------###
SummaryCodingsWindow <- function(){
 ###---------------DATA TABLES----------------------###
 sql <- RQDAQuery("select id,name from freecode where status=1")
 tableCodes <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Codes", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select catid,name from codecat where status=1")
 tableCodeCat <- data.frame(id = sql[[1]], name = sql[[2]],
                            type= rep("Code Categories", nrow(sql)),
                            visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select id,name from source where status=1")
 tableFiles <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Files", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select catid,name from filecat where status=1")
 tableFileCat <- data.frame(id = sql[[1]], name = sql[[2]],
                            type= rep("File Categories", nrow(sql)),
                            visible = rep(FALSE, nrow(sql)))
 sql <- RQDAQuery("select id,name from cases where status=1")
 tableCases <- data.frame(id = sql[[1]], name = sql[[2]],
                          type= rep("Cases", nrow(sql)),
                          visible = rep(FALSE, nrow(sql)))
 sqlAttributes <- RQDAQuery("select name from attributes where status=1")
 DFs <- list()
 DFs$codes <- rbind(tableCodes, tableCodeCat)
 DFs$files <- rbind(tableFiles, tableFileCat, tableCases)
 DFs$FilesCases <- rbind(tableFiles, tableCases)
 ###---------------WINDOW
 sca <- gtkWindow(show=FALSE)
 sca$setTitle("RQDAExt - Summary of Codings Analysis")
 sca$setDefaultSize(gdkScreenWidth()*2/3,gdkScreenHeight()*2/3)
 sca["border-width"] <- 10
 icon <- system.file("icon", "mainIcon.png", package = "RQDAExt")
```

```
image <- gdkPixbufNewFromFile(icon)
sca$setIcon(image$retval)
###--------------COMPONENTS
sca.boxes <- list()
sca.boxes$codes <- gtkComboBoxNewText()
sca.boxes$files <- gtkComboBoxNewText()
sapply(c("Codes", "Code Categories"), sca.boxes$codes$appendText)
sapply(c("Files","File Categories","Cases"), sca.boxes$files$appendText)
#
sca.labels <- c("Number of Codings","Average Length",
                "Number of Files","Codings for Each File")
sca.radiogp <- list()
sca.radiogp[[sca.labels[1]]] <- gtkRadioButton(label=sca.labels[1])
for (label in sca.labels[-1]){
  sca.radiogp[[label]] <- gtkRadioButton(sca.radiogp,label=label)
}
#
sca.device <- gtkDrawingArea()
asCairoDevice(sca.device)
sca.book <- gtkNotebook()
#
sca.view <- gtkTreeView()
sca.scroll <- gtkScrolledWindow()
sca.scroll$setPolicy("automatic", "automatic")
sca.scroll$add(sca.view)
#
sca.selectall <- list()
sca.selectall$button1 <- gtkButton("Select All")
sca.selectall$button2 <- gtkButton("Select All")
#
sca.table <-list()
sca.table$print <- gtkButton("Print")
sca.table$export <- gtkButton("Export in csv")
#
sca.general <-list()
sca.general$close <- gtkButton("Close")
sca.general$help <- gtkButton("Help")
#
sca.graph <- list()
sca.graph$plot <- gtkButton("Plot")
sca.graph$export <- gtkButton("Export graph")
sca.graph$clear <- gtkButton("Clear")
#
sca.adjust.bc <- list()
sca.adjust.bc$title <- gtkEntry()
sca.adjust.bc$horiz <- gtkCheckButton("Horizontal")
sca.adjust.bc$color <- gtkComboBoxNewText()
sca.adjust.bc$margin <- gtkSpinButton(min = 4, max = 15, step= 0.5)
sca.adjust.bc$cex <- gtkSpinButton(min = 0.5, max = 1, step= 0.05)
sapply(c("Gray","White", "Black","Red","Blue","Cyan","Green","Dark Green", "Purple",
        "Black Palette", "Red Palette", "Blue Palette", "Green Palette",
        "Purple Palette","Rainbow", "Heat Colors", "Terrain Colors",
        "CM Colors"), sca.adjust.bc$color$appendText)
sca.adjust.bc$color["active"]<- 0
#
sca.adjust.wc <- list()
sca.adjust.wc$color <- gtkComboBoxNewText()
sca.adjust.wc$randcolor <- gtkCheckButton("Random Colors")
sca.adjust.wc$rotat <- gtkCheckButton("% Rotation")
sca.adjust.wc$numrotat <- gtkSpinButton(min = 0, max = 100, step= 5)
sca.adjust.wc$randorder <- gtkCheckButton("Random Order")
sca.adjust.wc$max <- gtkSpinButton(min = 1, max = 6, step= 0.1)
sca.adjust.wc$min <- gtkSpinButton(min = 0.5, max = 2, step= 0.1)
```

```
sapply(c("Black", "Black2", "Blue", "Blue2", "Red", "Orange",
         "Green", "Green2", "Purple", "Pink", "Rainbow"),
       sca.adjust.wc$color$appendText)
sca.adjust.wc$color["active"]<- 0
#
sca.adjust.pc <- list()
sca.adjust.pc$title <- gtkEntry()
sca.adjust.pc$color <- gtkComboBoxNewText()
sca.adjust.pc$rad <- gtkSpinButton(min = 0.5, max = 1, step= 0.05)
sca.adjust.pc$cex <- gtkSpinButton(min = 0.5, max = 1, step= 0.05)
sca.adjust.pc$angle <- gtkSpinButton(min = 0, max = 360, step= 5)
sca.adjust.pc$threed <- gtkCheckButton("3D    Label Rad")
sca.adjust.pc$labelrad <- gtkSpinButton(min = 1, max = 2, step= 0.05)
sapply(c("Rainbow", "Heat Colors", "Terrain Colors",
         "Topo Colors", "CM Colors", "Gray Colors"),
       sca.adjust.pc$color$appendText)
sca.adjust.pc$color["active"]<- 0
#
sca.models <- list()
sca.models$codes <- rGtkDataFrame(DFs$codes)
sca.models$files <- rGtkDataFrame(DFs$files)
#
sca.filteredmodels <-list()
sca.filteredmodels$codes <- sca.models$codes$filter()
sca.filteredmodels$codes$setVisibleColumn(ncol(DFs$codes)-1)
sca.filteredmodels$files <- sca.models$files$filter()
sca.filteredmodels$files$setVisibleColumn(ncol(DFs$files)-1)
#
sca.views <- list()
sca.views$codes <- gtkTreeView(sca.filteredmodels$codes)
sca.views$files <- gtkTreeView(sca.filteredmodels$files)
sapply (sca.views, function (view){
  view$insertColumnWithAttributes (0, "Select names", gtkCellRendererText(), text = 1)
})
sapply (sca.views, function (view){
  selection <- view$getSelection()
  selection$setMode("multiple")
})
#
sca.scrolls <- list()
sca.scrolls$codes <- gtkScrolledWindow()
sca.scrolls$files <- gtkScrolledWindow()
mapply (gtkContainerAdd, object = sca.scrolls, widget = sca.views)
mapply (gtkScrolledWindowSetPolicy, sca.scrolls, "automatic", "automatic")
###----------CONTAINERS
sca.container <- gtkHBox(TRUE, 10)
sca.frame1 <- gtkFrame ("Filtering System & Data Display")
sca.frame2 <- gtkFrame ("Graphics Display")
sca.frame3 <- gtkFrame ("Type of Summary")
sca.vbox1 <- gtkVBox(FALSE, 10)
sca.vbox2 <- gtkVBox(FALSE, 10)
sca.vbox3 <- gtkVBox(FALSE, 5)
sca.vbox4 <- gtkVBox(FALSE, 5)
#
sca.hbox2 <- gtkHBox(FALSE, 5)
sca.hbox3 <- gtkHBox(FALSE, 5)
sca.grid <- gtkTable(rows = 7, columns = 2, homogeneous = FALSE)
#
sca.grid.bc <- gtkTable(rows = 6, columns = 2, homogeneous = FALSE)
sca.frame.bc <- gtkFrame ("Adjustment")
sca.grid.bc2 <- gtkTable(rows = 2, columns = 2, homogeneous = TRUE)
#
sca.grid.wc <- gtkTable(rows = 2, columns = 2, homogeneous = FALSE)
```

```
sca.frame.wc <- gtkFrame ("Scale")
sca.frame.wc2 <- gtkFrame ("Color")
sca.grid.wc2 <- gtkTable(rows = 2, columns = 2, homogeneous = TRUE)
sca.vbox.wc <- gtkVBox(FALSE, 5)
sca.hbox.wc <- gtkHBox(FALSE, 5)
#
sca.grid.pc <- gtkTable(rows = 5, columns = 2, homogeneous = FALSE)
sca.hbox.pc <- gtkHBox(FALSE, 5)
sca.grid.pc2 <- gtkTable(rows = 3, columns = 2, homogeneous = TRUE)
sca.frame.pc <- gtkFrame ("Adjustment")
###----------LAYOUT
sca.book$appendPage(sca.grid.bc, gtkLabel("Bar Graph"))
sca.book$appendPage(sca.grid.pc, gtkLabel("Pie Graph"))
sca.book$appendPage(sca.grid.wc, gtkLabel("WordCloud"))
#
sca$add(sca.container)
sca.container$packStart(sca.vbox1)
sca.container$packStart(sca.vbox2)
#
sca.vbox1$packStart(sca.frame1, expand = TRUE, fill = TRUE)
sca.vbox1$packStart(sca.vbox3, expand = FALSE, fill = FALSE)
sca.vbox1$packStart(sca.hbox2, expand = FALSE, fill = FALSE)
#
sca.frame1$add(sca.grid)
sca.grid$attach(gtkLabel("Codes"), left.attach =0,1 , top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
sca.grid$attach(sca.boxes$codes, left.attach = 0,1, top.attach = 1,2,
            xoptions = c("expand", "fill"), yoptions = "")
sca.grid$attach(sca.scrolls$codes, left.attach = 0,1, top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
sca.grid$attach(sca.selectall$button1, left.attach = 0,1, top.attach = 3,4,
            xoptions = c("expand", "fill"), yoptions = "")
sca.grid$attach(gtkLabel("Files"), left.attach =1,2 , top.attach = 0,1,
            xoptions = c("expand", "fill"), yoptions = "")
sca.grid$attach(sca.boxes$files, left.attach = 1,2, top.attach = 1,2,
            xoptions = c("expand", "fill"), yoptions = "")
sca.grid$attach(sca.scrolls$files, left.attach = 1,2, top.attach = 2,3,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
sca.grid$attach(sca.selectall$button2, left.attach = 1,2, top.attach = 3,4,
            xoptions = c("expand", "fill"), yoptions = "")
sca.grid$attach(sca.frame3, left.attach = 0,1, top.attach = 4,5,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
sca.radiobox <- gtkVBox(FALSE, 5)
sca.frame3$add(sca.radiobox)
sapply(sca.radiogp, gtkBoxPackStart, object=sca.radiobox)
sca.grid$attach(sca.table$print, left.attach = 0,1, top.attach = 5,6,
            xoptions = c("expand", "fill"), yoptions = "")
sca.grid$attach(sca.table$export, left.attach = 0,1, top.attach = 6,7,
            xoptions = c("expand", "fill"), yoptions = "")
sca.grid$attach(sca.scroll, left.attach = 1,2, top.attach = 4,7,
            xoptions = c("expand", "fill"), yoptions = c("expand", "fill"))
sca.grid$setColSpacing(0,5)
sca.grid$setRowSpacing(1,5)
sca.grid$setRowSpacing(2,5)
sca.grid$setRowSpacing(3,10)
sca.grid$setRowSpacing(4,5)
sca.grid$setRowSpacing(5,5)
sca.grid$setRowSpacing(6,5)
#
sapply(sca.general, sca.hbox2$packStart, expand = TRUE)
sca.hbox2$packStart(gtkLabel(""),expand=TRUE, fill = TRUE)
#
sca.vbox2$packStart(sca.frame2, expand = TRUE, fill = TRUE)
```

```
sca.frame2$add(sca.vbox4)
sca.vbox4$packStart(sca.device, expand = TRUE, fill = TRUE)
sca.vbox4$packStart(sca.book, expand = FALSE, fill = FALSE)
sca.vbox4$packStart(sca.hbox3, expand=FALSE, fill=FALSE)
#
sca.hbox3$packStart(gtkLabel(""), expand = TRUE, fill = TRUE)
sapply(sca.graph, sca.hbox3$packEnd, expand = TRUE)
###----------ADJUST LAYOUT
sca.grid.bc$attach(gtkLabel("Title"), left.attach =0,2 , top.attach = 0,1,
                   xoptions = "", yoptions = "")
sca.grid.bc$attach(sca.adjust.bc$title, left.attach =0,2 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.bc$attach(gtkLabel("Color"), left.attach =0,1 , top.attach = 2,3,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.bc$attach(sca.adjust.bc$color, left.attach =0,1 , top.attach = 3,4,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.bc$attach(sca.adjust.bc$horiz, left.attach =0,1 , top.attach = 4,5,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.bc$attach(sca.frame.bc, left.attach =1,2 , top.attach = 2,5,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.bc$setColSpacing(0,5)
#
sca.frame.bc$add(sca.grid.bc2)
sca.grid.bc2$attach(gtkLabel("Margin"), left.attach =0,1 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.bc2$attach(gtkLabel("Cex"), left.attach =0,1 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.bc2$attach(sca.adjust.bc$margin, left.attach =1,2 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.bc2$attach(sca.adjust.bc$cex, left.attach =1,2 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
#
sca.grid.wc$attach(sca.frame.wc, left.attach =0,1 , top.attach = 0,1,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.wc$attach(sca.frame.wc2, left.attach =1,2 , top.attach = 0,1,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.wc$attach(sca.adjust.wc$randorder, left.attach =0,1 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.wc$attach(sca.hbox.wc, left.attach =1,2 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.wc$setColSpacing(0,5)
#
sca.hbox.wc$packStart(sca.adjust.wc$rotat, expand=FALSE, fill = FALSE)
sca.hbox.wc$packStart(sca.adjust.wc$numrotat, expand=FALSE, fill = FALSE)
#
sca.frame.wc2$add(sca.vbox.wc)
sca.vbox.wc$packStart(sca.adjust.wc$color, expand=TRUE, fill = TRUE)
sca.vbox.wc$packStart(sca.adjust.wc$randcolor, expand=FALSE, fill = FALSE)
#
sca.frame.wc$add(sca.grid.wc2)
sca.grid.wc2$attach(gtkLabel("Min"), left.attach =0,1 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.wc2$attach(gtkLabel("Max"), left.attach =0,1 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.wc2$attach(sca.adjust.wc$min, left.attach =1,2 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.wc2$attach(sca.adjust.wc$max, left.attach =1,2 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
#
sca.grid.pc$attach(gtkLabel("Title"), left.attach =0,2 , top.attach = 0,1,
                   xoptions = "", yoptions = "")
sca.grid.pc$attach(sca.adjust.pc$title, left.attach =0,2 , top.attach = 1,2,
                   xoptions = c("expand", "fill"), yoptions = "")
```

```
sca.grid.pc$attach(gtkLabel("Color"), left.attach =0,1 , top.attach = 2,3,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc$attach(sca.adjust.pc$color, left.attach =0,1 , top.attach = 3,4,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc$attach(sca.hbox.pc, left.attach =0,1 , top.attach = 4,5,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc$attach(sca.frame.pc, left.attach =1,2 , top.attach = 2,5,
                   xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc$setColSpacing(0,5)
#
sca.hbox.pc$packStart(sca.adjust.pc$threed,expand=FALSE,fill=FALSE)
sca.hbox.pc$packStart(sca.adjust.pc$labelrad,expand=FALSE,fill=FALSE)
#
sca.frame.pc$add(sca.grid.pc2)
sca.grid.pc2$attach(gtkLabel("Radius"), left.attach =0,1 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc2$attach(gtkLabel("Cex"), left.attach =0,1 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc2$attach(gtkLabel("Init. Angle"), left.attach =0,1 , top.attach = 2,3,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc2$attach(sca.adjust.pc$rad, left.attach =1,2 , top.attach = 0,1,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc2$attach(sca.adjust.pc$cex, left.attach =1,2 , top.attach = 1,2,
                    xoptions = c("expand", "fill"), yoptions = "")
sca.grid.pc2$attach(sca.adjust.pc$angle, left.attach =1,2 , top.attach = 2,3,
                    xoptions = c("expand", "fill"), yoptions = "")
###----------------SIGNAL CONNECTIONS
gSignalConnect(sca.boxes$codes, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF["type"]
  DF["visible"] <- grepl(pattern, values)
}, data= sca.filteredmodels$codes)
gSignalConnect(sca.boxes$files, "changed", function(combo, user.data){
  pattern <- combo$getActiveText()
  DF <- user.data$getModel()
  values <- DF["type"]
  DF["visible"] <- grepl(pattern, values)
}, data= sca.filteredmodels$files)
gSignalConnect(sca.selectall$button1, "clicked", function (button){
  sca.views$codes$getSelection()$SelectAll()
})
gSignalConnect(sca.selectall$button2, "clicked", function (button){
  sca.views$files$getSelection()$SelectAll()
})
#
gSignalConnect(sca.general$close, "clicked", function(button){
  sca$destroy()
})
gSignalConnect(sca.general$help, "clicked", function(button){
  InstructionsWindow(active=5)
})
#
gSignalConnect(sca.table$export, "clicked", function (button){
  gtkTreeViewExportModel(sca.view, ".csv", "SummaryOfCodings")
})
gSignalConnect(sca.table$print, "clicked", function(button){
  cBoxValues <- list(sca.boxes$codes$getActiveText(),
                     sca.boxes$files$getActiveText())
  ind1 <- sca.views$codes$selectedIndices()
  ind2<- sca.views$files$selectedIndices()
  for (i in (1:length(sca.radiogp))){
    if(sca.radiobox[[i]]["active"]){
```

```
      variable <- sca.radiobox[[i]]$getLabel()
    }
  }
  if(ind1==0 || ind2==0){
    dialog <- gtkMessageDialog(parent = sca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    CodeValues <- sca.models$codes[ind1,1]
    FileValues <- sca.models$files[ind2,1]
    ###Clear the Main View
    gtkTreeViewClearColumns(sca.view)
    ###Make the table
    DFTable <- SummaryOfCodings(filter= cBoxValues, cid=CodeValues,
                                fid = FileValues, variable = variable)
    TableModel <- rGtkDataFrame(DFTable)
    sca.view$setModel(TableModel)
    cell_renderer1 <- gtkCellRendererText()
    cell_renderer1["background"] <-"gray92"
    mapply(sca.view$insertColumnWithAttributes,
           position = -1,
           title= "Coding",
           cell =list(cell_renderer1),
           text = 1-1)
    cell_renderer2 <- gtkCellRendererText()
    mapply(sca.view$insertColumnWithAttributes,
           position = -1,
           title= "Value",
           cell =list(cell_renderer2),
           text = 2-1)
  }
})
#
gSignalConnect(sca.graph$clear, "clicked", function (button){
  frame()
})
gSignalConnect(sca.graph$plot, "clicked", function(button){
  cBoxValues <- list(sca.boxes$codes$getActiveText(),
                     sca.boxes$files$getActiveText())
  ind1 <- sca.views$codes$selectedIndices()
  ind2<- sca.views$files$selectedIndices()
  for (i in (1:length(sca.radiogp))){
    if(sca.radiobox[[i]]["active"]){
      variable <- sca.radiobox[[i]]$getLabel()
    }
  }
  if(ind1==0 || ind2==0){
    dialog <- gtkMessageDialog(parent = sca,
                               flags = "destroy-with-parent",
                               type = "warning", button = "ok",
                               "No values selected from some tables")
    response <- dialog$run()
    if (response == GtkResponseType["ok"]){
      dialog$destroy()}
  }else {
    CodeValues <- sca.models$codes[ind1,1]
    FileValues <- sca.models$files[ind2,1]
    DFTable <- SummaryOfCodings(filter= cBoxValues, cid=CodeValues,
                                fid = FileValues, variable = variable)
```

```
if(sca.book$getCurrentPage()==0){
  title <- sca.adjust.bc$title$getText()
  cex <- sca.adjust.bc$cex$getValue()
  margin <- sca.adjust.bc$margin$getValue()
  horiz <- ifelse(sca.adjust.bc$horiz$active, 1,0)
  color <- sca.adjust.bc$color$getActiveText()
  num <- nrow(DFTable)
  #
  if(color=="Gray"){col= "gray"}
  if(color=="White"){col= "white"}
  if(color=="Black"){col= "black"}
  if(color=="Red"){col= "red"}
  if(color=="Blue"){col= "blue"}
  if(color=="Cyan"){col= "cyan"}
  if(color=="Green"){col= "green"}
  if(color=="Dark Green"){col= "dark green"}
  if(color=="Purple"){col= "purple"}
  if(color=="Black Palette"){col= palette(gray(seq(0,.9,len = num)))}
  if(color=="Red Palette"){col= colors()[c(552:556)]}
  if(color=="Blue Palette"){col= colors()[c(129:132)]}
  if(color=="Green Palette"){col= colors()[c(255:258)]}
  if(color=="Purple Palette"){col= colors()[c(462:466)]}
  if(color=="Rainbow"){col= rainbow(num)}
  if(color=="Heat Colors"){col= heat.colors(num)}
  if(color=="Terrain Colors"){col = terrain.colors(num)}
  if(color=="Topo Colors"){col = topo.colors(num)}
  if(color=="CM Colors"){col = cm.colors(num)}
  if(horiz==0){
    par(mar=c(margin, 2.5, 2, 0.5) + 0.1)#Modify first number
    barplot(height=DFTable$value, names.arg = DFTable$coding ,
            main=title, ylab="", xlab="", axis.lty = 1, las=2, col=col,
            cex.axis= cex, cex.names = cex, cex.main= cex+0.15)
  }
  if(horiz==1){
    par(mar=c(2.5, margin, 2, 1) + 0.1)##Modify second number
    barplot(height=DFTable$value, names.arg = DFTable$coding,
            main=title,ylab="", xlab="", axis.lty = 1, las=1, col=col,
            cex.axis= cex, cex.names = cex, cex.main= cex+0.15, horiz = 1)
  }
}
if(sca.book$getCurrentPage()==1){
  title <- sca.adjust.pc$title$getText()
  rad <- sca.adjust.pc$rad$getValue()
  cex <- sca.adjust.pc$cex$getValue()
  angle <- sca.adjust.pc$angle$getValue()
  labelrad <- sca.adjust.pc$labelrad$getValue()
  threed <- ifelse(sca.adjust.pc$threed$active, 1,0)
  values <- DFTable$value[DFTable$value!=0]
  lbls= DFTable$coding[DFTable$value!=0] #solo nombres
  pct <- round(values/sum(values)*100)
  lbls <- paste(lbls, pct)
  lbls <- paste(lbls,"%",sep="")#nombres+%
  color <- sca.adjust.pc$color$getActiveText()
  num <- nrow(DFTable)
  #
  if(color=="Rainbow"){col= rainbow(num)}
  if(color=="Heat Colors"){col= heat.colors(num)}
  if(color=="Terrain Colors"){col = terrain.colors(num)}
  if(color=="Topo Colors"){col = topo.colors(num)}
  if(color=="CM Colors"){col = cm.colors(num)}
  if(color=="Gray Colors"){col = palette(gray(seq(0,.9,len = num)))}
  if(length(values)!=0){
    if(threed==0){
```

```
        par(mar=c(0, 0,1, 0) + 0.1)
        pie(values, labels=lbls, main=title,cex=cex, radius=rad,
            init.angle = angle, cex.main=1, col=col)
       }
       if(threed==1){
        par(mar=c(0, 0, 1, 0) + 0.1)
        pos<-pie3D(values, main=title,
                explode = 0.1, labelcex = cex, radius = rad,
                start=angle*pi/180, cex.main= 1, col=col)
        pie3D.labels(labels = lbls, radialpos = pos,
                  labelrad=labelrad, labelcex=cex)
       }
      } else {frame()}

    }
    if(sca.book$getCurrentPage()==2){
      max <- sca.adjust.wc$max$getValue()
      min <- sca.adjust.wc$min$getValue()
      randorder <- ifelse(sca.adjust.wc$randorder$active, TRUE,FALSE)
      randcolor <- ifelse(sca.adjust.wc$randcolor$active, TRUE,FALSE)
      rotat <- ifelse(sca.adjust.wc$rotat$active, TRUE,FALSE)
      numrotat <- sca.adjust.wc$numrotat$getValue()
      values <- DFTable$value[DFTable$value!=0]
      words <-DFTable$coding[DFTable$value!=0]
      color <- sca.adjust.wc$color$getActiveText()
      num <- nrow(DFTable)
      #
      if(color=="Black"){col= "black"}
      if(color=="Black2"){col = palette(gray((seq(0,.9,len = num))))}
      if(color=="Blue"){col= colors()[c(129:132)]}
      if(color=="Blue2"){col= colors()[c(70:73)]}
      if(color=="Red"){col= colors()[c(552:556)]}
      if(color=="Orange"){col= colors()[c(90,94)]}
      if(color=="Green"){col= colors()[c(254:258)]}
      if(color=="Green2"){col= colors()[c(610:614)]}
      if(color=="Purple"){col= colors()[c(462:466)]}
      if(color=="Pink"){col= colors()[c(116:120)]}
      if(color=="Rainbow"){col= rainbow(1000)}


      if(length(values)!=0){
       if(rotat==TRUE){
        wordcloud(words, values,
                scale= c(max, min), random.order = randorder, min.freq=1,
                random.color = randcolor, rot.per = numrotat/100, colors=col)
       }
       if(rotat==FALSE){
        wordcloud(words, values,
                scale= c(max, min), random.order = randorder,
                random.color = randcolor, min.freq=1, colors=col)
       }
      } else {frame()}
    }
   }
  })
  ###---------------END
  sca$show()
  sca$move(gdkScreenWidth()*1/3,gdkScreenHeight()*1/4)
}
```

- **utilities.R**

```
###---------------AUXILAR FUNCTIONS-------------------###
```

```
gtkTreeViewSelectedIndices <- function (object){
  model <- object$getModel()
  paths <- object$getSelection()$getSelectedRows()$retval
  path_strings <- sapply(paths, function(i){
    model$convertPathToChildPath(i)$toString()
  } )
  if(length(path_strings))
    as.numeric(path_strings)+1
  else
    0
}
gtkTreeViewClearColumns <- function(object){
  column<- object$getColumn(0)
  if (is.null(column)==FALSE){
    for (i in (1:ncol(object$getModel()))){
      column <- object$getColumn(0)
      object$RemoveColumn(column)
    }
  }
}
gtkTreeViewExportModel <- function(object, extension, x) {
  title <- paste("Export in ", extension, sep="")
  name <- paste(x, extension, sep="")
  pattern <- paste("*", extension, sep="")
  if(extension==".csv") filter <- "CSV files"
  if(extension==".txt") filter <- "txt files"
  dialog <- gtkFileChooserDialog(title = title,
                                 parent=NULL, action = "save",
                                 "gtk-save" , GtkResponseType["ok"],
                                 "gtk-cancel" , GtkResponseType["cancel"],
                                 show = FALSE)
  dialog$SetDoOverwriteConfirmation(TRUE)
  dialog$SetCreateFolders(TRUE)
  dialog$SetCurrentName(name=name)
  fileFilter <- gtkFileFilter()
  fileFilter$setName (filter)
  fileFilter$addPattern (pattern)
  dialog$addFilter(fileFilter)
  gSignalConnect (dialog, "response" ,
                  f = function (dialog, response, data) {
                    if (response == GtkResponseType["ok"]) {
                      filename <- dialog$getFilename()
                      if(extension==".csv"){
                        write.table(object$getModel(), file = filename, sep = ";",
                                    row.names = FALSE)
                      }
                      if (extension==".txt"){
                        write.table(object$getModel(), file = filename, sep = "\t", quote=FALSE,
                                    row.names = FALSE)
                      }
                    }
                    dialog$destroy()
                  } )
  dialog$show()
}
gtkTextViewExportBuffer <- function(object, x) {
  name <- paste(x, ".txt", sep="")
  dialog <- gtkFileChooserDialog(title = "Export in txt",
                                 parent=NULL, action = "save",
                                 "gtk-save" , GtkResponseType["ok"],
                                 "gtk-cancel" , GtkResponseType["cancel"],
                                 show = FALSE)
  dialog$SetDoOverwriteConfirmation(TRUE)
```

```
  dialog$SetCreateFolders(TRUE)
  dialog$SetCurrentName(name=name)
  fileFilter <- gtkFileFilter()
  fileFilter$setName ("txt files")
  fileFilter$addPattern ("*txt")
  dialog$addFilter(fileFilter)
  gSignalConnect (dialog, "response" ,
                  f = function (dialog, response, data) {
                    if (response == GtkResponseType["ok"]) {
                      filename <- dialog$getFilename()
                      TextBounds<- object$getBounds()
                      write(TextBounds$start$getText(TextBounds$end),
                            file = filename, sep = "\n")
                    }
                    dialog$destroy()
                  } )
  dialog$show()
}
gtkExportNet <- function(object, type) {
  if(type=="nodes"){
    title <- "Export Nodes"
    name <- "Nodes.csv"
  }
  if(type=="links"){
    title <- "Export Links"
    name <- "Links.csv"
  }
  filter <- "CSV files"
  dialog <- gtkFileChooserDialog(title = title,
                                 parent=NULL, action = "save",
                                 "gtk-save" , GtkResponseType["ok"],
                                 "gtk-cancel" , GtkResponseType["cancel"],
                                 show = FALSE)
  dialog$SetDoOverwriteConfirmation(TRUE)
  dialog$SetCreateFolders(TRUE)
  dialog$SetCurrentName(name=name)
  fileFilter <- gtkFileFilter()
  fileFilter$setName (filter)
  fileFilter$addPattern ("*.csv")
  dialog$addFilter(fileFilter)
  gSignalConnect (dialog, "response" ,
                  f = function (dialog, response, data) {
                    if (response == GtkResponseType["ok"]) {
                      filename <- dialog$getFilename()
                      write.table(object, file = filename, sep = ",",row.names = FALSE)
                    }
                    dialog$destroy()
                  } )
  dialog$show()
}
```

- **zzz.R**

```
.onAttach <- function(...) {
  if (interactive()) {
    packageStartupMessage("\nUse 'RQDAExt()' to start the programme.\n")
    RQDAExt()
  }
}

.onUnload <- function(...){
  cat("RQDAExt has been unloaded.\n")
}
```

ANEXO IV – Archivos de datos

Este anexo contiene los archivos de datos de carpeta \inst correspondiente a toda aquella información adiconal que será instalada en el directorio principal del paquete cuando sea instalado.

- ## Carpeta \icon

- ## Carpeta \extdata
  - ### help1

GENERAL

Welcome to the RQDAExt: R-based Qualitative Data Analysis Extension. This GUI has been designed for being used with the GUI provided by 'RQDA' (HUANG, Ronggui. (2014). RQDA: R-based Qualitative Data Analysis. R package version 0.2-7. URL <http://rqda.r-forge.r-project.org/>. It complements and extends the package 'RQDA' with new options for the analysis and graphics with this GUI.

Steps:
1. Check that you have already done a project with RQDA. If not, do it.
2. Open a .rqda file with RQDAExt.
3. Choose one of the different analysis that RQDAExt can do:
   - Code Vs Code Analysis.
   - Code Vs File Analysis.
   - Coding Analysis.
   - Summary of Codings Analysis.
   - Attribute Analysis.

  - ### help2

CODE VS CODE ANALYSIS

The Code Vs Code Analysis allows to visualize the concurrences between different codings in order to find and establish possible relationships between them. The results can be shown with a table or a network chart.

Steps:
1. Choose the elements of the combo boxes. They indicate how you can filter the row codes, the code columns and the files.
2. Select the elements of the tables. These will be the codings that will be represented in the table or the graphics. Multiple choice or select all option are available.
3. Choose one of these types of relation between codings from the last combo box:
   - Inclusion.
   - Overlap.
   - Exact.
   - Proximity.
4. If you want to display the table of concurrences, go to the first tab of the notebook, click the button "Print" and wait few seconds.
5. If you want to display graphics, go to the second tab of the notebook, select between the two types of chart and click on "Plot". Probably you will have to adjust some parameters of the graphic displayed so every time that you adjust something you will have to click on “Plot” again.
6. Export your results.

Note: In the network charts try to unselect the codings (in the filtering system) without edges for a correct visualization.
Note 2: You can change the size of the window with aim of do zoom in the view.

- **help3**

CODE VS FILE ANALYSIS

The Code Vs File Analysis allows to visualize the concurrences between different codings and files/cases in order to find and establish possible relationships between them. The results can be shown with a table or a network chart.

Steps:
1. Choose the elements of the combo boxes. They indicate how you can filter the codes and the files.
2. Select the elements of the tables. These will be the codings and the files that will be represented in the table or the graphics. Multiple choice or select all option are available.
3. If you want to display the table of concurrences, go to the first tab of the notebook, click the button "Print" and wait few seconds.
4. 5. If you want to display graphics, go to the second tab of the notebook, select between the two types of chart and click on "Plot". Probably you will have to adjust some parameters of the graphic displayed so every time that you adjust something you will have to click on "Plot" again.
6. Export your results.

Note: In the network charts try to unselect the codings or files (in the filtering system) without edges for a correct visualization.
Note 2: You can change the size of the window with aim of do zoom in the view.

- **help4**

CODING ANALYSIS

The Coding Analysis allows to find the codings from several files with an AND or OR conjunction, in order to study the content of them.

Steps:
1. Choose the elements of the combo boxes. They indicate how you can filter the codes and the files.
2. Select the elements of the tables. These will be the codings that will appear in the text and the files where they are from. Multiple choice or select all option are available.
3. Choose one of the logical conjunctions between the codings:
    - AND.
    - OR.
4. Click on "Print" button to get the codings.
5. Exports the codings.

Note: If only one code is selected, chose one of the logical conjunctions indifferently.

- **help5**

SUMMARY OF CODING ANALYSIS

The Summary of Coding Analysis allows to visualize the appearance frequencies of several codings in order to do find and establish the more used concepts. The results can be shown with a table or a statistical chart.

Steps.
1. Choose the elements of the combo boxes. They indicate how you can filter the codes and the files.
2. Select the elements of the tables. These will be the codings and the files used to count their frequencies. Multiple choice or select all option are available.
3. Choose one of the types of summary:
    - Number of Codings. Number of coding for each code.
    - Average Length. Average number of characters in codings for each code.
    - Number of Files. Number of files coded for each code.
    - Codings for Each File. Number of codings for each file.
4. If you want to display a table with the frequencies, click the button "Print".
5. If you want to display graphics, choose the type of the graphic and click on "Plot". Probably you will have to adjust some parameters of the graphic displayed so every time that you adjust something you will have to click on "Plot" again.
6. Export your results.

- **help6**

ATTRIBUTE ANALYSIS

The Attribute Analysis allows two things. Search the attributes from one file or case in order to do a fast query or sort the files or cases from their attributes.

Steps for the first analysis:
1. Choose one element of the combo boxes. It indicates whether the query will be from a files or a cases.
2. Select only one element from the table.
3. Click on "Print" button to consult the attributes.

Steps for the second analysis:
1. Choose one element of the combo boxes. It indicates whether the sorting will be from a files or a cases.
2. Select only one attribute from the radio box.
3. Click on "Search" button to obtain the sorting.
4. Export your results.

- **NEWS file**

Version 0.1
* Initial release

- **LICENCE file**

YEAR: 2016
COPYRIGHT HOLDER: Gerard Pueyo
ORGANIZATION: Universitat Politecnica de Catalunya, Tarrasa