```
# wildLifeStrikeDataSet <- function() {
#   #setting the download parameters
#   URL <- getWData()
#   destfile <- paste(getDataDir(), "wildlife.zip", sep = "/")
#
#   method="auto"
#
#   #if the file exists then do not download again
#   if (file.exists(destfile) != TRUE)
#   {
#     download.file(URL, destfile, method)
#   } else
#   {
#     message("File exists no download required.")
#   }
#
#   destdir <- getDataDir()
#
#   #unzip the file
#   unzip(destfile, exdir = destdir)
#
#   csvfile <- paste(destdir, "/STRIKE_REPORTS (1990-1999).csv", sep="")
#
#   if (file.exists(csvfile) != TRUE)
#   {
#     setwd(getDataDir())
#     system(paste("java -jar ", getDataDir(), "/access2csv.jar ", getDataDir(), "/wildlife.accdb", sep
#     setwd(getMainDir())
#   } else
#   {
#     message("File exists no extract required.")
#   }
#
# }
#
# onTimeFlightPerformanceDataSet <- function() {
#
#   method="auto"
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#   startMonth <- getStartMonth()
#   endMonth <- getEndMonth()
#
#   for (i in startYear:endYear){
#     for (j in startMonth:endMonth){
#
#       variableName <- paste("On_Time_On_Time_Performance_", i, "_", j, sep = "")
#
#       sourceFile <- paste(variableName, ".zip", sep = "")
#       URL <- paste(getFData(), sourceFile, sep = "")
#       destinationFile <- paste(dataDir, "/", sourceFile, sep = "")
#
```

```
#       #if the file exists then do not download again
#       if (file.exists(destinationFile) != TRUE)
#       {
#         message("Downloading ", sourceFile)
#         download.file(URL, destinationFile, method)
#         Sys.sleep(0.1)
#       } else
#       {
#         message(sourceFile," file exists, no download is required.")
#       }
#
#       zippedFileName <- sourceFile
#       zippedFile <- destinationFile
#       unzippedFileName <- paste(variableName, ".csv", sep = "")
#       unzippedFile <- paste(dataDir, "/", unzippedFileName, sep = "")
#
#       #if the file exists then do not unzip it again
#       if (file.exists(unzippedFile) != TRUE)
#       {
#         message("Unzipping ", zippedFileName)
#         unzip(zippedFile, overwrite = FALSE, exdir = dataDir) #No overwrite, so if the unzip was done
#         #Clear warnings to get rid of the unzip warnings created because of the "readme.html" file, wh
#         assign("last.warning", NULL, envir = baseenv())
#       } else
#       {
#         message(unzippedFileName," file exists, no unzip is required.")
#       }
#
#       #if the variable is available, then do not reassign it
# #      if (exists(variableName) != TRUE){
# #        message("Reading ", variableName)
# #        assign(variableName, data.table(read.csv(unzippedFile, header = TRUE)), envir = .GlobalEnv)
# #      } else
# #      {
# #        message(variableName," variable exists, no assign is required.")
# #      }
#
#
#     } #end of "for (j in startMonth:endMonth)"
#   } #end of "for (i in startYear:endYear)"
#
# }
# wildLifeStrikeDataSetDataCleanup <- function() {
#
#   destdir <- getDataDir()
#
#   #Read files to data frames --> data tables
#   #if the variable is available, then do not reassign it
#   if (exists("sr_1990_1999") != TRUE){
#     message("Reading sr_1990_1999")
#     sr_1990_1999 <<- data.table(read.csv(paste(destdir,"/STRIKE_REPORTS (1990-1999).csv",sep=""), head
#     names(sr_1990_1999) <- c("INDEX_NR","OPID","OPERATOR","ATYPE","AMA","AMO","EMA","EMO","AC_CLASS",
#   } else
```

```
#   {
#     message("sr_1990_1999 variable exists, no assign is required.")
#   }
#
#   if (exists("sr_2000_2009") != TRUE){
#     message("Reading sr_2000_2009")
#     sr_2000_2009 <<- data.table(read.csv(paste(destdir,"/STRIKE_REPORTS (2000-2009).csv",sep=""), hea
#     names(sr_2000_2009) <- c("INDEX_NR","OPID","OPERATOR","ATYPE","AMA","AMO","EMA","EMO","AC_CLASS",
#   } else
#   {
#     message("sr_2000_2009 variable exists, no assign is required.")
#   }
#
#   if (exists("sr_2010_Current") != TRUE){
#     message("Reading sr_2010_Current")
#     sr_2010_Current <<- data.table(read.csv(paste(destdir,"/STRIKE_REPORTS (2010-Current).csv",sep="")
#     names(sr_2010_Current) <- c("INDEX_NR","OPID","OPERATOR","ATYPE","AMA","AMO","EMA","EMO","AC_CLAS
#   } else
#   {
#     message("sr_2010_Current variable exists, no assign is required.")
#   }
#
#
#   #STRIKE_REPORTS_BASH --> only military, not required
#   #srb_1990_Current <- data.table(read.csv(paste(destdir,"/STRIKE_REPORTS_BASH (1990-Current).csv",sep
#   #names(srb_1990_Current) <- c("INDEX_NR","OPID","OPERATOR","ATYPE","AMA","AMO","EMA","EMO","AC_CLAS
# }

#
# onTimeFlightPerformanceDataSetDataCleanup <- function() {
#   dataDir <- getDataDir()
#   startYear <- getStartYear()
#   endYear <- getEndYear()
#   startMonth <- getStartMonth()
#   endMonth <- getEndMonth()
#
#   for (i in startYear:endYear){
#     for (j in startMonth:endMonth){
#
#       variableName <- paste("On_Time_On_Time_Performance_", i, "_", j, sep = "")
#
#       unzippedFileName <- paste(variableName, ".csv", sep = "")
#       unzippedFile <- paste(dataDir, "/", unzippedFileName, sep = "")
#
#       #if the variable is available, then do not reassign it
# #      if (exists(variableName) != TRUE){
# #        message("Reading ", variableName)
# #        assign(variableName, data.table(read.csv(unzippedFile, header = TRUE)), envir = .GlobalEnv)
# #      } else
# #      {
# #        message(variableName," variable exists, no assign is required.")
# #      }
#
#       #TODO:
```

```
#       #- remove unused and/or duplicated columns
#       #- annual data set merge
#       #- save separate data file for each year --> ?? size
#
#
#     } #end of "for (j in startMonth:endMonth)"
#   } #end of "for (i in startYear:endYear)"
#
# }
# #Project functions
#
# #Function name: loadLibraries
# #Input: none
# #Output: none
# #Main use: load the required libraries for the project, if library is not installed, than installs it
#
# loadLibraries <- function() {
#   if (!require(installr)) {install.packages("installr"); require(installr)}
#   if (!require(RODBC)) {install.packages("RODBC"); require(RODBC)}
#   if (!require(knitr)) {install.packages("knitr"); require(knitr)}
#   if (!require(data.table)) {install.packages("data.table"); require(data.table)}
#   if (!require(dplyr)) {install.packages("dplyr"); require(dplyr)}
#   if (!require(dtplyr)) {install.packages("dtplyr"); require(dtplyr)}
#   if (!require(ggplot2)) {install.packages("ggplot2"); require(ggplot2)}
#   if (!require(ReporteRs)) {install.packages("ReporteRs"); require(ReporteRs)}
#   if (!require(yaml)) {install.packages("yaml"); require(yaml)}
#
#   #update R
#   updateR(TRUE)
#
#   #update MiKTeX packages
#   #system("mpm --update --quiet")
#
#   #require(grid)
#   #require(lattice)
#   #require(ggplot2movies)
#   #require(latticeExtra)
# }
#
#
#
# #Function name: versionDetails
# #Input: none
# #Output: The version details of the environment used for the project
# #Main use: with just one function call have the possibility to provide the environment details into t
#
# versionDetails <- function() {
#
#   cat(paste(
#     "R Studio version 1.0.143\n\n",
#     version$version.string, " ", version$`svn rev`,"\n\n",
#     "Package versions:\n",
#     "- RODBC version ", packageVersion("RODBC"),"\n",
```

```
#     "- knitr version ", packageVersion("knitr"),"\n",
#     "- data.table version ", packageVersion("data.table"),"\n",
#     "- dplyr version ", packageVersion("dplyr"),"\n",
#     "- dtplyr version ", packageVersion("dtplyr"),"\n",
#     "- ReporteRs version ", packageVersion("ReporteRs"),"\n",
#     "- ReporteRsjars version ", packageVersion("ReporteRsjars"),"\n",
#     "- installr version ", packageVersion("installr"),"\n",
#     "- stringr version ", packageVersion("stringr"),"\n",
#     "- ggplot2 version ", packageVersion("ggplot2"),"\n",
#     "- yaml version ", packageVersion("yaml"),"\n\n",
#     "Base package versions:\n",
#     "- stats version ", packageVersion("stats"),"\n",
#     "- graphics version ", packageVersion("graphics"),"\n",
#     "- grDevices version ", packageVersion("grDevices"),"\n",
#     "- utils version ", packageVersion("utils"),"\n",
#     "- datasets version ", packageVersion("datasets"),"\n",
#     "- methods version ", packageVersion("methods"),"\n",
#     "- base version ", packageVersion("base"),sep=""))
#
# }
#
# #Function name: versionDetailsMiKTeX
# #Input: none
# #Output: The version details of the environment used for the project
# #Main use: with just one function call have the possibility to provide the environment details into t
#
# versionDetailsMiKTeX <- function() {
#
#   cat(system("mpm --version", intern = TRUE), sep = '\n')
#
# }
#
#
# #Function name: versionDetailsMiKTeXPackages
# #Input: none
# #Output: The version details of the environment used for the project
# #Main use: with just one function call have the possibility to provide the environment details into t
#
# versionDetailsMiKTeXPackages <- function() {
#
#   cat(system("mpm --list", intern = TRUE), sep = '\n')
#
# }
#
#
# #Function name: readConfigFile
# #Input: none
# #Output: none
# #Main use: reads the config file to a global variable to use in the session
#
# readConfigFile <- function(a) {
#   if (a == TRUE){
#     config <<- yaml.load_file("91-Config.yaml")
#   }
```

```
#   else {
#     config <<- yaml.load_file("../91-Config.yaml")
#   }
# }
#
# #Function name: getMainDir
# #Input: none
# #Output: the main directory from the config file
# #Main use: call it whenever you need to get the main directory - might not be the same as the result
#
# getMainDir <- function() {
#   return(config$directories$maindir)
# }
#
#
# #Function name: getBackupDir
# #Input: none
# #Output: the name of the backup directory from the config file, plus the timestamp directory created
# #Main use: call it whenever you need to get the backup directory
#
# getBackupDir <- function() {
#   backupdir <- config$directories$backupdir
#   subdir <- Sys.Date()
#   returnvalue <- file.path(backupdir, subdir)
#
#   if (!file.exists(returnvalue)){
#     dir.create(returnvalue)
#     dir.create(file.path(returnvalue, "Documents"))
#   }
#
#   return(returnvalue)
#
# }
#
#
# #Function name: getDocDir
# #Input: none
# #Output: the Documents directory from the config file
# #Main use: call it whenever you need to get the Documents directory
#
# getDocDir <- function() {
#   return(config$directories$documents)
# }
#
# #Function name: getDocInputDir
# #Input: none
# #Output: the Documents directory from the config file
# #Main use: call it whenever you need to get the Documents directory
#
# getDocInputDir <- function() {
#   return(config$directories$documentinput)
# }
#
# #Function name: getDocOutputDir
```

```
# #Input: none
# #Output: the Documents directory from the config file
# #Main use: call it whenever you need to get the Documents directory
#
# getDocOutputDir <- function() {
#   return(config$directories$documentoutput)
# }
#
#
# #Function name: getDataDir
# #Input: none
# #Output: the DataSets directory from the config file
# #Main use: call it whenever you need to get the DataSets directory
#
# getDataDir <- function() {
#   return(config$directories$datasets)
# }
#
#
# #Function name: getStartYear
# #Input: none
# #Output: start year
# #Main use: call it whenever you need to get the start year of the data set to work with
#
# getStartYear <- function() {
#   return(config$years$startyear)
# }
#
#
# #Function name: getEndYear
# #Input: none
# #Output: start year
# #Main use: call it whenever you need to get the end year of the data set to work with
#
# getEndYear <- function() {
#   return(config$years$endyear)
# }
#
#
# #Function name: getStartMonth
# #Input: none
# #Output: start year
# #Main use: call it whenever you need to get the start month of the data set to work with
#
# getStartMonth <- function() {
#   return(config$months$startmonth)
# }
#
#
# #Function name: getEndMonth
# #Input: none
# #Output: start year
# #Main use: call it whenever you need to get the end month of the data set to work with
#
```

```
# getEndMonth <- function() {
#   return(config$months$endmonth)
# }
#
#
# #Function name: backupFiles
# #Input: none
# #Output: the name of the backup directory from the config file, plus the timestamp directory created a
# #Main use: call it whenever you need to get the backup directory
#
# backupFiles <- function() {
#
#   #Main directoery files
#   filesMain <- list.files(getMainDir(), full.names = TRUE)
#   file.copy(filesMain, getBackupDir(), overwrite = TRUE)
#
#   #Documents folder
#   filesDocuments <- list.files(getDocDir(), full.names = TRUE)
#   file.copy(filesDocuments, file.path(getBackupDir(), "Documents"), overwrite = TRUE)
#
# }
#
#
# #Function name: getWData
# #Input: none
# #Output: the Wildlife Data Set url from the config file
# #Main use: call it whenever you need to get the url
#
# getWData <- function() {
#   return(config$sources$wildlife)
# }
#
#
# #Function name: getFData
# #Input: none
# #Output: the Flight Data Set url from the config file
# #Main use: call it whenever you need to get the url
#
# getFData <- function() {
#   return(config$sources$flightdata)
# }
#
#
#
# #Function name: removeDataSetVariables
# #Input: none
# #Output: none
# #Main use: call it whenever you need to cleanup the Data Set variables
#
# removeDataSetVariables <- function() {
#
#   #rm(list = ls(pattern = "On_Time_On_Time_Performance*", envir = .GlobalEnv), envir = .GlobalEnv)
#   #rm(list = ls(pattern = "sr_*", envir = .GlobalEnv), envir = .GlobalEnv)
#
```

```
# }
#
# #Function name: loadSourceCodeFunctions
# #Input:
# #Output:
# #Main use:
#
# loadSourceCodeFunctions <- function() {
#
#   source("01-WildLiveStrikeDataSetDataPreparation.R")
#   source("02-OnTimeFlightPerformanceDataSetDataPreparation.R")
#   source("03-WildLiveStrikeDataSetDataCleanup.R")
#
# }
#
# #Function name: addWatermark
# #Input: plot
# #Output: plot with watermark
# #Main use: adds watermark to the plot
#
# addWatermark <- function(p) {
#   labelText <- "Final Paper - Gabor Horvath"
#   temp <- ggplot_build(p)
#   x_pos <- mean(temp$panel$ranges[[1]]$x.range)
#   y_pos <- mean(temp$panel$ranges[[1]]$y.range)
#   x_pos1 <- mean(c(temp$panel$ranges[[1]]$x.range[1],x_pos))
#   y_pos1 <- mean(c(temp$panel$ranges[[1]]$y.range[2],y_pos))
#   x_pos2 <- mean(c(temp$panel$ranges[[1]]$x.range[2],x_pos))
#   y_pos2 <- mean(c(temp$panel$ranges[[1]]$y.range[2],y_pos))
#   x_pos3 <- mean(c(temp$panel$ranges[[1]]$x.range[1],x_pos))
#   y_pos3 <- mean(c(temp$panel$ranges[[1]]$y.range[1],y_pos))
#   x_pos4 <- mean(c(temp$panel$ranges[[1]]$x.range[2],x_pos))
#   y_pos4 <- mean(c(temp$panel$ranges[[1]]$y.range[1],y_pos))
#   watermarked = p +
#     annotate("text", x = x_pos, y = y_pos, label = labelText, size = 12, col="black", fontface = "bold
#     annotate("text", x = x_pos1, y = y_pos1, label = labelText, size = 7, col="black", fontface = "bol
#     annotate("text", x = x_pos2, y = y_pos2, label = labelText, size = 7, col="black", fontface = "bol
#     annotate("text", x = x_pos3, y = y_pos3, label = labelText, size = 7, col="black", fontface = "bol
#     annotate("text", x = x_pos4, y = y_pos4, label = labelText, size = 7, col="black", fontface = "bol
#   return(watermarked)
# }
#
#
# #Function name: addFlippedWatermark
# #Input: plot
# #Output: plot with watermark when x and y are flipped using coord_flip()
# #Main use: adds watermark to the plot
#
# addFlippedWatermark <- function(p) {
#   labelText <- "Final Paper - Gabor Horvath"
#   temp <- ggplot_build(p)
#   x_pos <- mean(temp$panel$ranges[[1]]$x.range)
#   y_pos <- mean(temp$panel$ranges[[1]]$y.range)
#   x_pos1 <- mean(c(temp$panel$ranges[[1]]$x.range[1],x_pos))
```

```
#   y_pos1 <- mean(c(temp$panel$ranges[[1]]$y.range[2],y_pos))
#   x_pos2 <- mean(c(temp$panel$ranges[[1]]$x.range[2],x_pos))
#   y_pos2 <- mean(c(temp$panel$ranges[[1]]$y.range[2],y_pos))
#   x_pos3 <- mean(c(temp$panel$ranges[[1]]$x.range[1],x_pos))
#   y_pos3 <- mean(c(temp$panel$ranges[[1]]$y.range[1],y_pos))
#   x_pos4 <- mean(c(temp$panel$ranges[[1]]$x.range[2],x_pos))
#   y_pos4 <- mean(c(temp$panel$ranges[[1]]$y.range[1],y_pos))
#   watermarked = p +
#     annotate("text", y = x_pos, x = y_pos, label = labelText, size = 12, col="black", fontface = "bold
#     annotate("text", y = x_pos1, x = y_pos1, label = labelText, size = 7, col="black", fontface = "bol
#     annotate("text", y = x_pos2, x = y_pos2, label = labelText, size = 7, col="black", fontface = "bol
#     annotate("text", y = x_pos3, x = y_pos3, label = labelText, size = 7, col="black", fontface = "bol
#     annotate("text", y = x_pos4, x = y_pos4, label = labelText, size = 7, col="black", fontface = "bol
#   return(watermarked)
# }
#
#
# #Function name: addOneWatermark
# #Input: plot
# #Output: plot with watermark
# #Main use: adds watermark to the plot
#
# addOneWatermark <- function(p) {
#   labelText <- "Final Paper - Gabor Horvath"
#   temp <- ggplot_build(p)
#   x_pos <- mean(temp$panel$ranges[[1]]$x.range)
#   y_pos <- mean(temp$panel$ranges[[1]]$y.range)
#   watermarked = p +
#     annotate("text", x = x_pos, y = y_pos, label = labelText, size = 5, col="black", fontface = "bold
#   return(watermarked)
# }
#
#
# #Function name: addTSWatermark
# #Input: plot
# #Output: plot with watermark
# #Main use: adds watermark to the plot
#
# addTSWatermark <- function(p, d) {
#   labelText <- "Final Paper - Gabor Horvath"
#   temp <- ggplot_build(p)
#   y_pos <- mean(temp$panel$ranges[[1]]$y.range)
#   watermarked = p +
#     annotate("text", x = d, y = y_pos, label = labelText, size = 12, col="black", fontface = "bold", a
#   return(watermarked)
# }
#
#
# #Function name: startJPG
# #Input: string - file name
# #Output: N/A
# #Main use: change the default values for the plots
#
# startJPG <- function(s) {
```

```
#   jpeg(
#     s, #File name, no directory!
#     width = 800, #width of the plot in px (should be the same as the height)
#     height = 800, #height of the plot in px (should be the same as the width)
#     quality = 99, #image quality in percentage, smaller value = higher compression
#     res = 72 #nominal resolution in ppi (pixels per inch)
#   )
# }
```