KID
Function

Fabian Blasch

28.09.2021

Packages

```
# Packages
get.package <- function(package){

  lapply(package, \(x){
    # check if packages are installed and if not install them
    if(!require(x, character.only = T)){
       install.packages(x)
    }
    # call package
    library(x, character.only = T)
  })

}

# exec
get.package(c("png", "jpeg", "tabulizer", "pdftools", "raster", "rgdal", "sp",
              "cluster", "fastcluster"))




# since I will use Map() / lapply() alot for plotting I will wrap them in invisible()
invis.Map <- function(f, ...) invisible(Map(f, ...))
invis.lapply <- function(x, f, ...) invisible(lapply(x, f, ...))
```

Actual SRRI

We can obtain the actual SRRI from the file name. Later this data will be utilized to evaluate the classification accuracy of the applied methods.

```
# set
setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs")

# files
file_names <- list.files(pattern = ".pdf", recursive = T)

# create df
dat.valid.SRRI <- as.data.frame(cbind("KID" = file_names,
                        "SRRI" = sapply(strsplit(sapply(strsplit(file_names, "_", fixed = T),
```

```r
                                        function(x) x[length(x)]), ".", fixed = T), "[", 1)))

# split first col
dat.valid.SRRI[, "KAG"] <- sapply(strsplit(dat.valid.SRRI[, 1], "/"), "[", 1)
dat.valid.SRRI[, "KID"] <- sapply(strsplit(dat.valid.SRRI[, 1], "/"), "[", 2)

# order
dat.valid.SRRI <- dat.valid.SRRI[, c(3, 1, 2)]

# glimpse
head(dat.valid.SRRI, 7)
```

```
##       KAG             KID SRRI
## 1 Allianz ki-allakt_6.pdf    6
## 2 Allianz  ki-allap_6.pdf    6
## 3 Allianz ki-alleur_2.pdf    2
## 4 Allianz  ki-allna_6.pdf    6
## 5 Allianz ki-allnar_2.pdf    2
## 6 Allianz ki-allore_3.pdf    3
## 7 Allianz ki-allost_6.pdf    6
```

```r
# dim
dim(dat.valid.SRRI)
```

```
## [1] 121   3
```

```r
# Hist
hist(as.numeric(dat.valid.SRRI[, "SRRI"]), breaks = 10, main = "SRRI", col = 4, xlab = "")
```

Shade Color

To extract the SRRI the following colors are required and need to be converted to HEX.

```
# set
{setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs/Auxiliary")

# import
dat.col.KAG <- read.table(list.files(pattern = "RGB"),
                          col.names = c("KAG", "R", "G", "B"))

# add hex
sapply(as.data.frame(t(dat.col.KAG[, -1])),
       function(x) do.call(rgb, as.list(c(x, maxColorValue = 255)))) -> HEX

# bind
dat.col.KAG <- cbind(dat.col.KAG, "HEX" = HEX)

# display
dat.col.KAG

# list of RG vectors for all KAGs
KAG_RGB <- setNames(lapply(as.data.frame(t(dat.col.KAG)[2:4, ]), function(x){
  as.numeric(x)
}), nm = dat.col.KAG[, 1])
```

```r
# correct IQAM
dat.col.KAG[5, 5] <- "#959595"

# IQAM actually features changing color shemes across their KIDs
# therefore the format haws to change from df to list as we have to match
# against multiple colors

list.col.KAG <- as.list(dat.col.KAG[order(dat.col.KAG$KAG), ncol(dat.col.KAG)]) |>
                        setNames(nm = dat.col.KAG$KAG[order(dat.col.KAG$KAG)])

# add second col to IQAM
list.col.KAG[["IQAM"]] <- c(list.col.KAG[["IQAM"]], "#949494")

# correct Schoellerbank col and add
list.col.KAG[["Schoellerbank"]] <- "#E3E3E3"
list.col.KAG[["Schoellerbank"]] <- c(list.col.KAG[["Schoellerbank"]], "#CBCBCB")

# write
# write.csv(dat.col.KAG, "KAG_COL_HEX.csv")
# saveRDS(list.col.KAG, file = "list_col_KAG.rds")
}
```

SRRI Extraction Function

Given a KID document this function aims to extract the SRRI from the standard graph (usually) located on the first of two pages.

```r
# load package
{setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/Code/Package/KIDs")
devtools::load_all()}
```

Tests

Starting with one KAG.

Erste

```r
# set wd to file that contains
setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs")

# safe dirs
dirs <- list.dirs()[-c(1, 4)] # remove hardcode later

# colors
col <- dat.col.KAG[order(dat.col.KAG[, "KAG"]), c("KAG", "HEX")]
col[5, 1] <- "Kepler Fonds"

# test Erste
Map(function(x){

  # set
  {setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs")
   setwd(x)
```

```r
  # ,pdfs
  file_nom <- list.files(pattern = ".pdf")}

   # FUN over all .pdfs
  lapply(file_nom, function(z){
    SRRI_ext_loc(doc = z, col = dat.col.KAG[4, 5])
  })


}, dirs[3]) -> erste.test

# extracted SRRI
cbind(dat.valid.SRRI[dat.valid.SRRI[, "KAG"] == "Erste", ],
      "Extracted" = sapply(erste.test[[1]], "[[", 2)) -> res

par(mfrow = c(3, 4))

# plot
invis.Map(function(x, y, z, l, k, m){

  {plot(x[, 1], x[, 2], col = x[, ncol(x)], pch = 19, main = paste("Predicted:", z, "| Actual:", l),
        xlim = c(1, 590))
  abline(v = y, col = "red", lwd = 2)
  lapply(k, function(x) abline(v = x, col = "grey", lwd = 2, lty = 2))}

}, lapply(erste.test[[1]], "[[", 3), sapply(erste.test[[1]], "[[", 4), res[, 4], res[, 3],
   lapply(erste.test[[1]], "[[", 5), lapply(erste.test[[1]], "[[", 5))
```

In the case of Erste the SRRI extraction works perfectly. Now the remaining KAGs will be examined.

```
# store Errors
utils::capture.output(

  # Map over dirs
  Map(function(x, y){

    # set
```

```r
    {setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs")
     setwd(x)

    # ,pdfs
    file_nom <- list.files(pattern = ".pdf")}

     # lapply over all .pdfs
    lapply(file_nom, function(z){

      # extract and error handle
      try(SRRI_ext_loc(doc = z, col = y, tol = 50), silent = F)

    })

  }, dirs, list.col.KAG) -> test

, type = "message")
```

```
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
```

```
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in median.default(ext.loc[[1]][, cut.off.point]) :
##   need numeric data
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_loc(doc = z, col = y, tol = 50) :
##   Error: No pixels of given color detected.
```

```
## [1] "Zusätzlich: Warnmeldungen:"
## [2] "1: In in_dir(input_dir(), evaluate(code, envir = env, new_device = FALSE,  :"
## [3] "  You changed the working directory to C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs (probal
```

```
## [4] "2: In in_dir(input_dir(), evaluate(code, envir = env, new_device = FALSE,  :"
## [5] "  You changed the working directory to C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs/Auxili
## [6] "3: In in_dir(input_dir(), evaluate(code, envir = env, new_device = FALSE,  :"
## [7] "  You changed the working directory to C:/Users/blasc/OneDrive/Documents/GitHub/KID/Code/Packag
## [8] "4: In in_dir(input_dir(), evaluate(code, envir = env, new_device = FALSE,  :"
## [9] "  You changed the working directory to C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs/Erste
```

```r
# error index
lapply(test, function(x){

  # error ind
  which(sapply(x, class) == "try-error")

}) -> err.tmp

# retrieve error throwing funds with ind
do.call(rbind, Map(function(x, y, z){

  if(length(y) > 0){

    {setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs")
     setwd(z)

    # .pdfs
    file_nom <- list.files(pattern = ".pdf")}

    # subset
    cbind(rep(z, length(y)),
          file_nom[y],
          sapply(x[y], "[", 1))

  } else {
   cbind(NA, NA, "No errros.")
  }

}, test, err.tmp, dirs)) -> dat.err
```

Now that we have identified all KIDs for which the extraction failed, we can proceed to see if the classification was correct for the remaining kids.

```r
# Plot
Map(function(x, y){

  sapply(y, function(z){
    # cond
    if(class(z) == "try-error"){
      return(NA)
    } else {
      z[[2]]
    }
  }) -> tmp

  # match
  cbind(dat.valid.SRRI[dat.valid.SRRI[, "KAG"] == x, ],
        "Extracted" = tmp)
```

```r
}, col[, 1], test) -> tef

# plot

# over KAGs
invis.Map(function(m, n){

  # arrange
  par(mfrow = c(ceiling(length(m) / 4), 4))

  # over KIDs
  invis.Map(function(x, y, z, k){

    if(class(x) == "try-error"){

      # plot empty for KIDs that remain unclassified for now
      plot(NULL, xlim = c(0, 1), ylim = c(0, 1), main = paste(k, "\n", "Error"))

    } else {
      # build tmp vars for plotting
      plot.coo <- x[[3]]
      med <- x[[4]]
      scal <- x[[5]]
      pred <- y
      act <- z
      fund <- k

      # plot
      plot(plot.coo[, 1], plot.coo[, 2], col = plot.coo[, ncol(plot.coo)], pch = 19,
           xlim = c(1, 590),
           main = paste(fund, "\n", "Predicted:", pred, "| Actual:", act))

      # median
      abline(v = med, col = "red", lty = 1, lwd = 2)

      # Scale
      lapply(scal, function(s) abline(v = s, col = "grey", lwd = 2, lty = 2))
      }

    },m , n[, 4], n[, 3], n[, 1])

}, test, tef)
```

Allianz
Predicted: 6 | Actual: 6
Allianz
Predicted: 6 | Actual: 6
Allianz
Predicted: 2 | Actual: 2
Allianz
Predicted: 6 | Actual: 6
Allianz
Predicted: 2 | Actual: 2
Allianz
Predicted: 3 | Actual: 3
Allianz
Predicted: 6 | Actual: 6
Allianz
Predicted: 1 | Actual: 1
Allianz
Predicted: 3 | Actual: 3
Allianz
Predicted: 4 | Actual: 4
Allianz
Predicted: 3 | Actual: 3
plot.coo[, 1]
plot.coo[, 2]

Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 4 | Actual: 4
Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 4 | Actual: 4
Amundi
Predicted: 4 | Actual: 4
Amundi
Error
Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 4 | Actual: 4
Amundi
Predicted: 5 | Actual: 5
Amundi
Predicted: 4 | Actual: 4
plot.coo[, 1]
plot.coo[, 2]
Index
NULL

Erste
Predicted: 4 | Actual: 4
Erste
Predicted: 6 | Actual: 6
Erste
Predicted: 1 | Actual: 1
Erste
Predicted: 6 | Actual: 6
Erste
Predicted: 4 | Actual: 4
Erste
Predicted: 7 | Actual: 7
Erste
Predicted: 7 | Actual: 7
Erste
Predicted: 5 | Actual: 5
Erste
Predicted: 3 | Actual: 3
Erste
Predicted: 4 | Actual: 4
Erste
Predicted: 2 | Actual: 2
plot.coo[, 1]
plot.coo[, 2]

IQAM
Predicted: 4 | Actual: 4
IQAM
Predicted: 2 | Actual: 2
IQAM
Predicted: 6 | Actual: 6
IQAM
Predicted: 6 | Actual: 6
IQAM
Predicted: 5 | Actual: 5
IQAM
Predicted: 6 | Actual: 6
IQAM
Predicted: 2 | Actual: 2
IQAM
Predicted: 2 | Actual: 2
IQAM
Predicted: 6 | Actual: 6
plot.coo[, 1]
plot.coo[, 2]

Kepler Fonds
Predicted: 5 | Actual: 5
Kepler Fonds
Predicted: 5 | Actual: 5
Kepler Fonds
Predicted: 2 | Actual: 2
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 4 | Actual: 4
Kepler Fonds
Predicted: 4 | Actual: 4
Kepler Fonds
Predicted: 2 | Actual: 2
Kepler Fonds
Predicted: 3 | Actual: 3
Kepler Fonds
Predicted: 2 | Actual: 2
plot.coo[, 1]
plot.coo[, 2]

Masterinvest
Predicted: 7 | Actual: 7
Masterinvest
Predicted: 3 | Actual: 3
Masterinvest
Predicted: 6 | Actual: 6
Masterinvest
Predicted: 6 | Actual: 6
Masterinvest
Predicted: 2 | Actual: 2
Masterinvest
Predicted: 2 | Actual: 2
Masterinvest
Predicted: 6 | Actual: 6
Masterinvest
Predicted: 5 | Actual: 5
Masterinvest
Predicted: 4 | Actual: 4
Masterinvest
Predicted: 3 | Actual: 3
Masterinvest
Predicted: 4 | Actual: 4
Masterinvest
Predicted: 5 | Actual: 5
plot.coo[, 1]
plot.coo[, 2]

Raiffeisen
Error
NULL
Index
0.0
0.2
0.4
0.6
0.8
1.0

Schoellerbank
Predicted: 6 | Actual: 6
Schoellerbank
Predicted: 6 | Actual: 6
Schoellerbank
Predicted: 6 | Actual: 6
Schoellerbank
Predicted: 5 | Actual: 5
Schoellerbank
Predicted: 3 | Actual: 5
Schoellerbank
Predicted: 3 | Actual: 2
Schoellerbank
Predicted: 5 | Actual: 5
Schoellerbank
Error
Schoellerbank
Error
Schoellerbank
Predicted: 4 | Actual: 4
Schoellerbank
Predicted: 6 | Actual: 6
Schoellerbank
Error
Schoellerbank
Predicted: 7 | Actual: 4
Schoellerbank
Predicted: 2 | Actual: 2
Schoellerbank
Predicted: 3 | Actual: 3
Schoellerbank
Predicted: 3 | Actual: 3
Schoellerbank
Predicted: 3 | Actual: 2
plot.coo[, 1]
plot.coo[, 2]
Index
NULL

Security
Error
NULL
Index
0.0
0.2
0.4
0.6
0.8
1.0

Problem description by KAG

- Allianz
  - Works for the given sample.
- Amundi
  - Works for every KID excluding one that is a scanned pdf. Unfortunately, all Amundi files have 2 bars in the same color as the SRRI shade,# accordingly the weight of SRRI shade is reduced and the median no longer lies in the middle of the SRRI shading. For now all scanned PDFs will return a warning. Later one could implement an algorithm that detects whether bars cover the

entire page and accordingly remove those bitmap entries.

- Erste
  - Works for the given sample.
- IQAM
  - Works for the given sample.
- Kepler Fonds
  - Works for the given sample.
- Masterinvest
  - Works for he given sample.
- Raiffeisen
  - pdf.data readout does not contain the full scale
  - Check color
  - Check last three PDFs for text detection issues
- Schoellerbank
  - No box detected, check color
  - three PDFs threw an error, all because of text detection.
- Security
  - pdf.data readout does not contain the full scale
  - Scale is completely off
  - rectangle is missing in some files
  - check cutoff
  - check color
- Union
  - check SRRI text detection