KID
Function

Fabian Blasch

16.08.2021

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"))




# 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 SSRI

We can obtain the actual SSRI 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.SSRI <- cbind("KID" = file_names,
                        "SSRI" = sapply(strsplit(sapply(strsplit(file_names, "_", fixed = T),
```

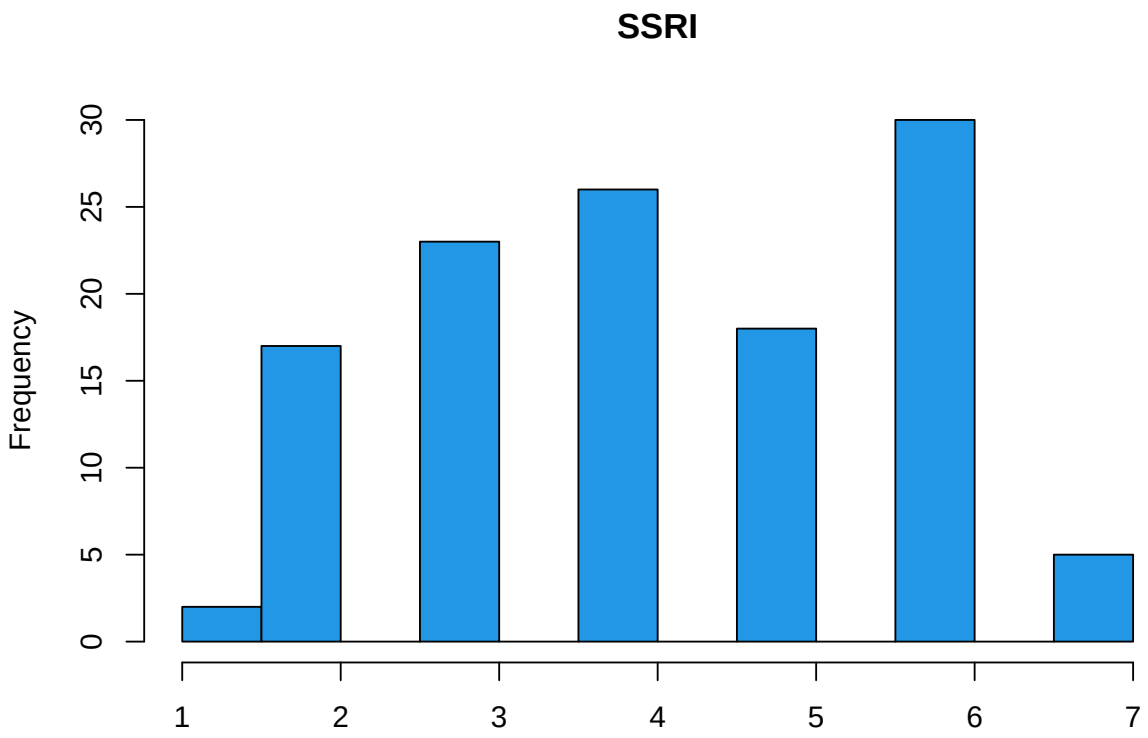
```
                                        function(x) x[length(x)]), ".", fixed = T), "[", 1))

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

```
##      KID                        SSRI
## [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"
```

```
# dim
dim(dat.valid.SSRI)
```

```
## [1] 121   2
```

```
# Hist
hist(as.numeric(dat.valid.SSRI[, 2]), breaks = 10, main = "SSRI", col = 4, xlab = "")
```



Shade Color

To extract the SSRI the following colors are required and need to be converted to hex.

```r
# 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
```

```
##               KAG   R   G   B     HEX
## V1     Raiffeisen   0  82 140 #00528C
## V2        Allianz 166 166 166 #A6A6A6
## V3         Amundi 204 210 219 #CCD2DB
## V4          Erste 166 166 166 #A6A6A6
## V5            IQAM 128 128 128 #808080
## V6         Kepler 204 204 204 #CCCCCC
## V7   Masterinvest  99 177 229 #63B1E5
## V8  Schoellerbank 217 217 217 #D9D9D9
## V9       Security 193 193 193 #C1C1C1
## V10         Union 196 197 199 #C4C5C7
```

SSRI Extraction Function

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

```r
# doc ... path to file
# col ... HEX code of color shade

SSRI.ext <- function(doc, col){

  ## FIND PAGE ##

  # convert pdf to text and identify line of interest
  pdf.text <- strsplit(pdftools::pdf_text(doc), "\n")

  # obtain relative line on page
  sapply(pdf.text, function(y){
    # id
    tmp1 <- grep("Risiko- und Ertragsprofil", y) / length(y)

    # return
    if(length(tmp1) == 0){
      return(NA)
    } else {
      return(tmp1)
    }
```

```
}) -> pos.vec

# ERROR if both pages yield a value different to NA
if(sum(is.na(pos.vec)) > 1) stop("Error: Could not uniquely idenfify position of SSRI.")

##  BITMAP  ##

# identify Page
page.SSRI <- which(!is.na(pos.vec))

# generate bitmap
bit.map <- pdftools::pdf_render_page(doc, page = page.SSRI, dpi = 50)

# subset array
ind.page.len <- round(dim(bit.map)[3] * (pos.vec[page.SSRI] - 0.1))

# return
bit.map.sub <- bit.map[ , , -c(ind.page.len:1)]

## COLOR ##

# split HEX
col.split <- unlist(strsplit(gsub("(.{2})", "\\1 ",
                             unlist(strsplit(dat.col.KAG[4, 5], "#"))[[2]]), " "))

# convert to lower case
col.split <- tolower(col.split)

## COORDINATES ##

# Shade
coo <- which(bit.map.sub[1,,] == col.split[1] & bit.map.sub[2,,] == col.split[2] &
             bit.map.sub[3,,] == col.split[3], arr.ind = T)

# stopif no pixels of desired color detected
if(nrow(coo) < 1) stop("Error: No pixels of given color detected.")

# margin
coob <- which(bit.map[1,,] == "00" & bit.map[2,,] == "00" & bit.map[3,,] == "00",
              arr.ind = T)

# lsm / rsm
lsm <- min(coob[, 1])
rsm <- max(coob[, 1])

# scale
int_leng <- (rsm - lsm) / 7

# midpoints
scale <- setNames(cumsum(c(lsm + int_leng / 2, rep(int_leng, 6))), 1:7)

## CLASSIFICATIOn ##
```

```
  # p = 5, method = "average"

  # get grouping
  grps <- agnes(coo, method = "average", diss = F)

  # restrict amnt of groups
  grps <- cutree(grps, k = 5)

  # bind
  dat.grps <- as.data.frame(cbind(coo, grps))

  ## IDENTIFY CLUSTER ##

  # identify cluster with minimum sum of variance
  which.min(rowSums(aggregate(dat.grps[, 1:2], by = dat.grps[, 3, drop = F],
                              var))) -> rect.grp
  # median
  med.rect.grp <- median(dat.grps[, 1][which(dat.grps[, 3] == rect.grp)])

  # return minimum absolute difference
  dif <- abs(med.rect.grp - scale)

  # which
  SSRI <- which.min(dif)

  # return
  list(dif,
       SSRI)
}

# TEST
SSRI.ext(doc = "C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs/Erste/kid-eb-147-t2957-at_de-de_en_4.p
                col = dat.col.KAG[4, 5])
```

```
## [[1]]
##         1         2         3         4         5         6         7
## 164.71429 110.14286  55.57143   1.00000  53.57143 108.14286 162.71429
##
## [[2]]
## 4
## 4
```

Result of First Test

One Pixel of the predicted scale, it does not get much better than that. Lets hope this works close to as well for all the other documents.