KID
Function

Anonymized for Review

27.04.2022

1 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() and lapply() for plotting I will wrap them in invisible()
invis.Map <- function(f, ...) invisible(Map(f, ...))
invis.lapply <- function(x, f, ...) invisible(lapply(x, f, ...))
```

2 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("./../../KIDs")

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

# test files
file_names_test <- file_names[grepl("Test", file_names, fixed = TRUE)]

# remove test set files
file_names <- file_names[!grepl("Test", file_names, fixed = TRUE)]
```

```r
# create df
dat.valid.SRRI_list <- lapply(list(file_names, file_names_test), \(x){

                        as.data.frame(cbind("KID" = x,
                                            "SRRI" = sapply(strsplit(sapply(strsplit(x, "_", fixed = T),
                                                                            function(x) x[length(x)]), ".", fixed = T), "[",
                        })

# assign
dat.valid.SRRI <- dat.valid.SRRI_list[[1]]
dat.valid.SRRI_test <- dat.valid.SRRI_list[[2]]

## remove last two entries for each KAG because of changes to test set ##

# 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)
```

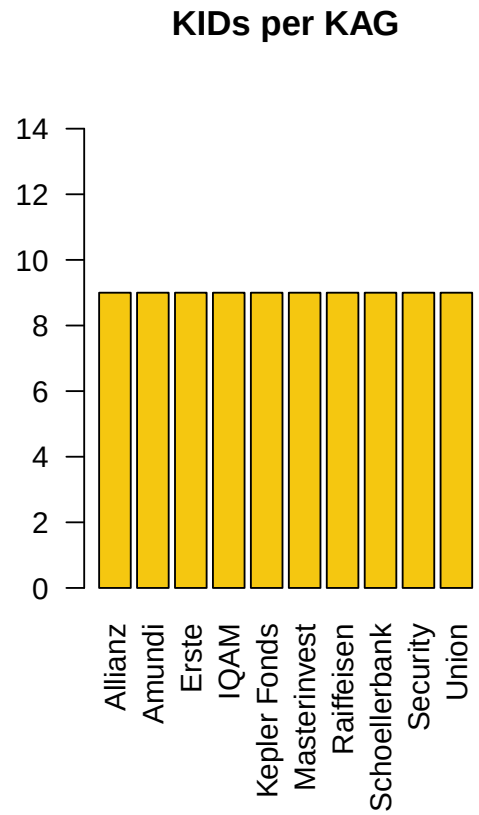
```
##       KAG             KID SRRI
## 1 Allianz ki-allakt_6.pdf    6
## 2 Allianz  ki-allap_6.pdf    6
## 3 Allianz ki-alleur_2.pdf    2
## 4 Allianz  ki-allna_6.pdf    6
## 5 Allianz ki-allost_6.pdf    6
## 6 Allianz ki-allsta_1.pdf    1
## 7 Allianz ki-allvor_3.pdf    3
```

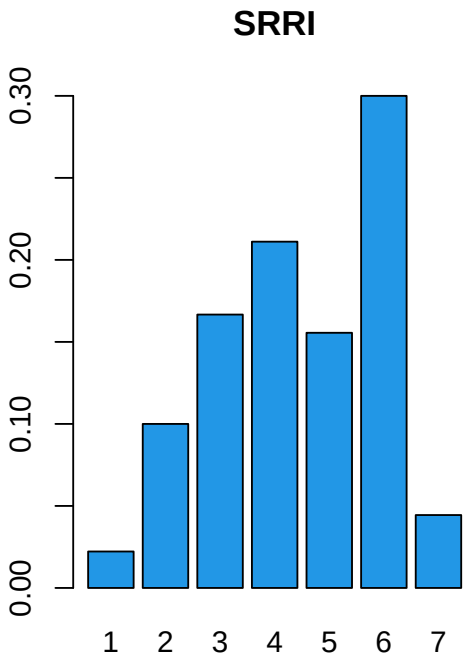
```r
# dim
dim(dat.valid.SRRI)
```

```
## [1] 90  3
```

```r
## count KIDs per KAG ##

# required dirs
dirs <- list.dirs()
dirs <- dirs[!grepl("Test", dirs, fixed = TRUE)][-c(1, 4)]

# step into first KAGs dir
setwd("Allianz")

# loop over dirs
sapply(dirs, \(x){

  # setwd
  setwd(paste0("./../", x))

  # count KIDs
  length(list.files(pattern = ".pdf"))
```

```
}) -> sample_split

# corr names
names(sample_split) <- sapply(strsplit(names(sample_split), split = "/", fixed = TRUE), "[[", 2)

# save
# pdf(file = "data_overview.pdf", height = 4)

# alignd
par(mar = c(7, 4, 4, 2) + 0.1, mfrow = c(1, 2))

# KIDs per KAG
barplot(sort(sample_split, decreasing = T), las = 2, col = 7, ylim = c(0, 15),
        main = "KIDs per KAG")

# table
tab <- table(as.numeric(dat.valid.SRRI[, "SRRI"]))

# barplot
barplot(tab / sum(tab), main = "SRRI", col = 4, xlab = "", ylim = c(0, 0.3))
```






```
# dev.off()
```

3 Performance evaluation

The SRRI is measured on an increasing ordinal scale, with 7 being associated with the highest risk and simultaneously also with the highest return. This would in theory allow for a measurement of predictive performance utilizing the abolute or squared difference between prediction and actual value. However, in this case we want to measure whether the read-out was successful or not. Hence the performance evaluation will be based on predicting correctly. Formally this means, we will use a discrete metric, i.e

$$Accuracy = \frac{\sum_{i=1}^{n} \mathbb{1}_{\{pred_i = act_i\}}}{n}.$$

Put differently, the accuracy measure is equivalent with the relative amount of correctly predicted cases.

4 Extraction utilizing pixel agglomeration

The first method which will be evaluated is based on hierarchical agglomerative clustering methods. All pixels in the color of the SRRI shade are extracted and grouped. The group associated with the highest probability of containing the SRRI is used to calculate the median of the horizontal position on the page. Then the predcited SRRI results as the SRRI value with the least horizontal absolute distance to the calculated median.

4.1 Shade Color

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

```
# set
{setwd("./../../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])

# 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")

# correct Union
list.col.KAG[["Union"]] <- "#4F4E4E"

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

4.2 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.

```
# load package
{setwd("./../Package/KIDs")
devtools::load_all()}
```

```
## i Loading KIDs
```

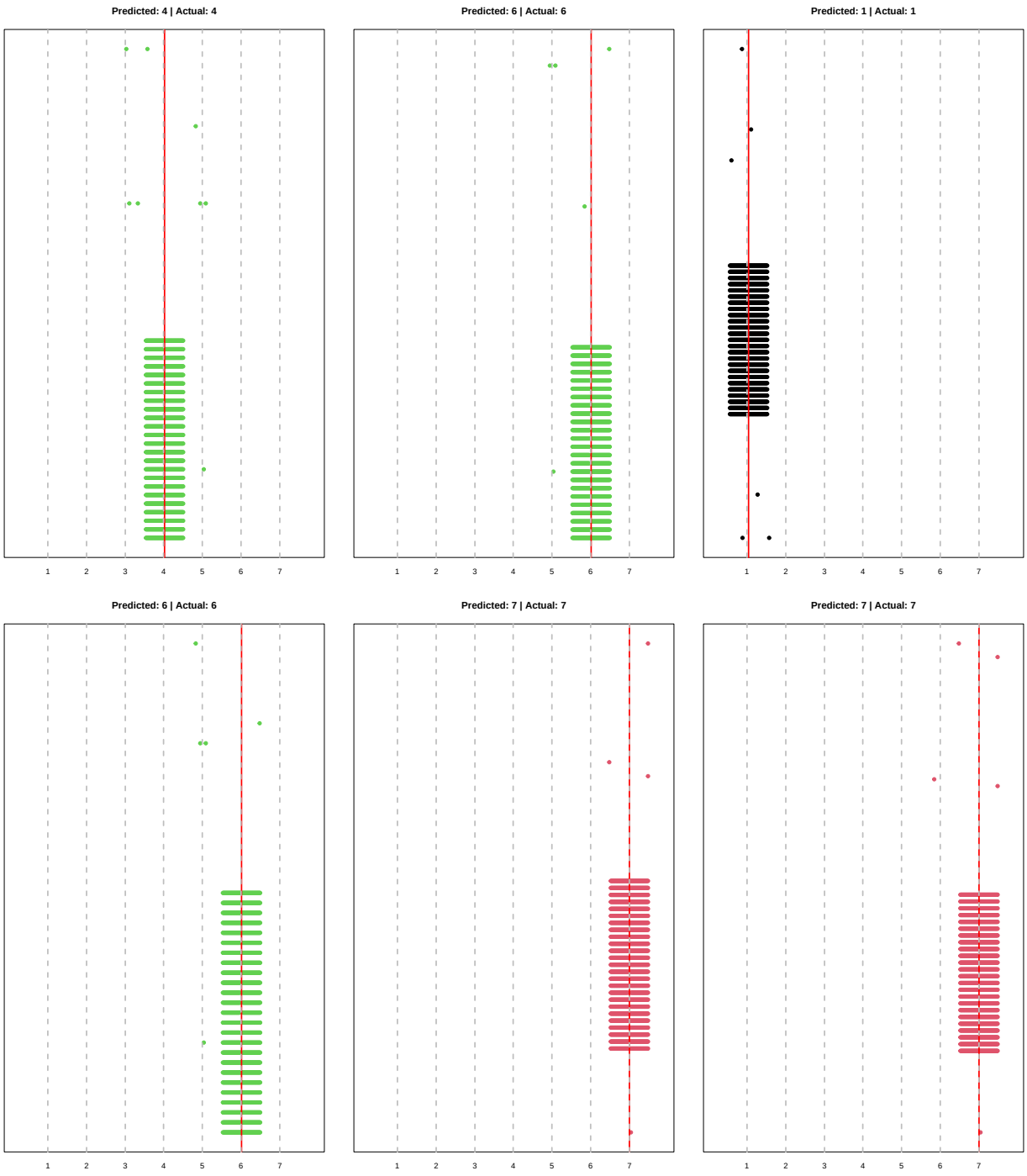
4.3 Tests

Starting with one KAG.

4.3.1 Erste

```
# setwd to file that contains KIDs
setwd("./../../KIDs")

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

# test Erste
Map(function(x){

  # change dir to respective KAG
  {setwd(x)

  # 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

# align

#pdf("Ersteextr.pdf", height = 6)
par(mfrow = c(2, 3), cex.lab = 2, mar = c(3, 1, 4, 2) + 0.1)

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

  {plot(x[, 1], x[, 2], col = x[, ncol(x)], pch = 19,
        xlim = c(1, 590), yaxt = "n", ylab = "", xlab = "", xaxt = "n")
   title(paste("Predicted:", z, "| Actual:", l))

  axis(1, at = k, labels = 1:7, tick  = FALSE)
  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)[1:6], sapply(erste.test[[1]], "[[", 4)[1:6], res[, 4][1:6], res[, 3]
   lapply(erste.test[[1]], "[[", 5)[1:6], lapply(erste.test[[1]], "[[", 5)[1:6])
```




```
#dev.off()
```

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

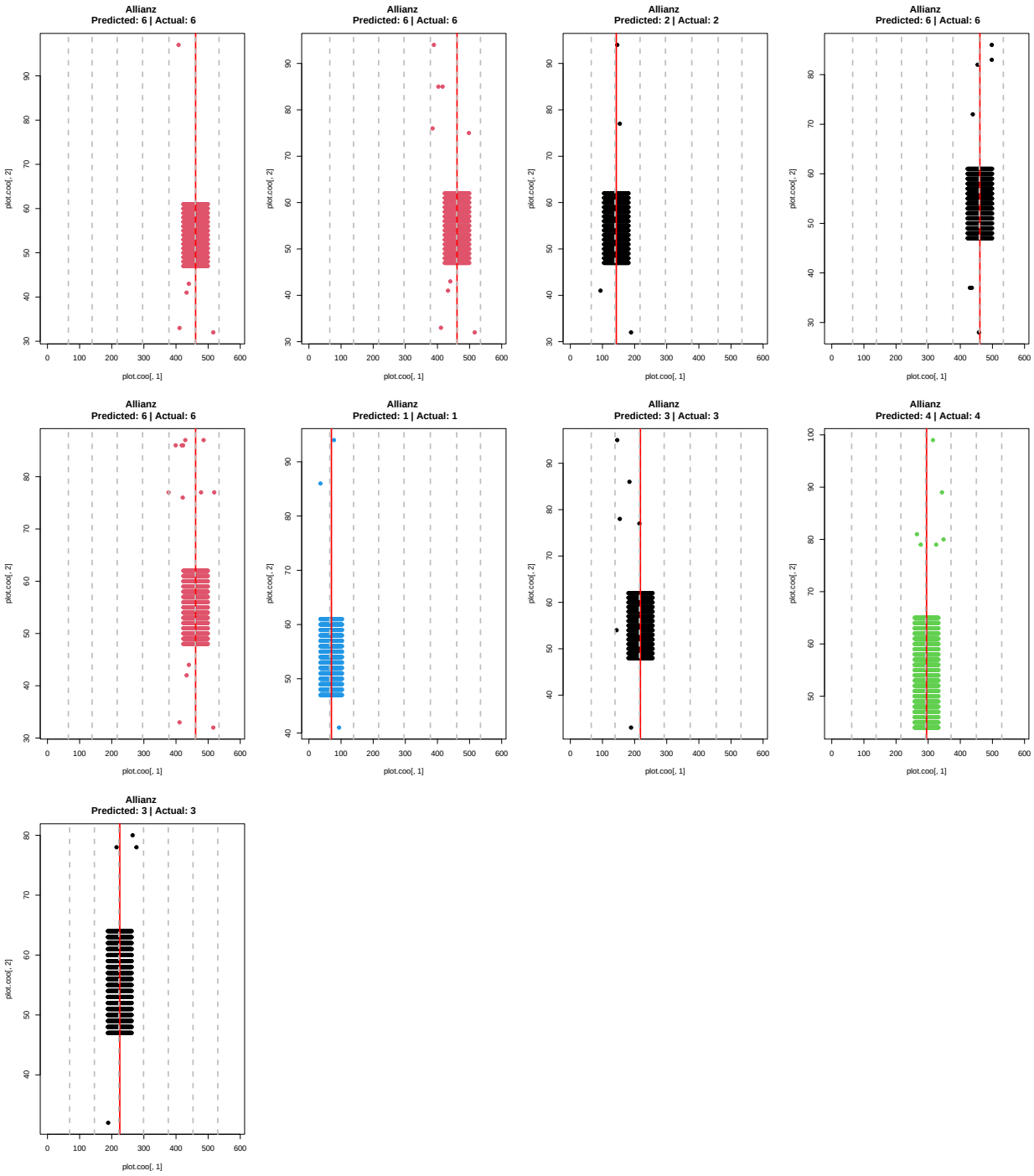
```
# setwd to file that contains the first KIDs
setwd("./../../KIDs/Allianz")

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

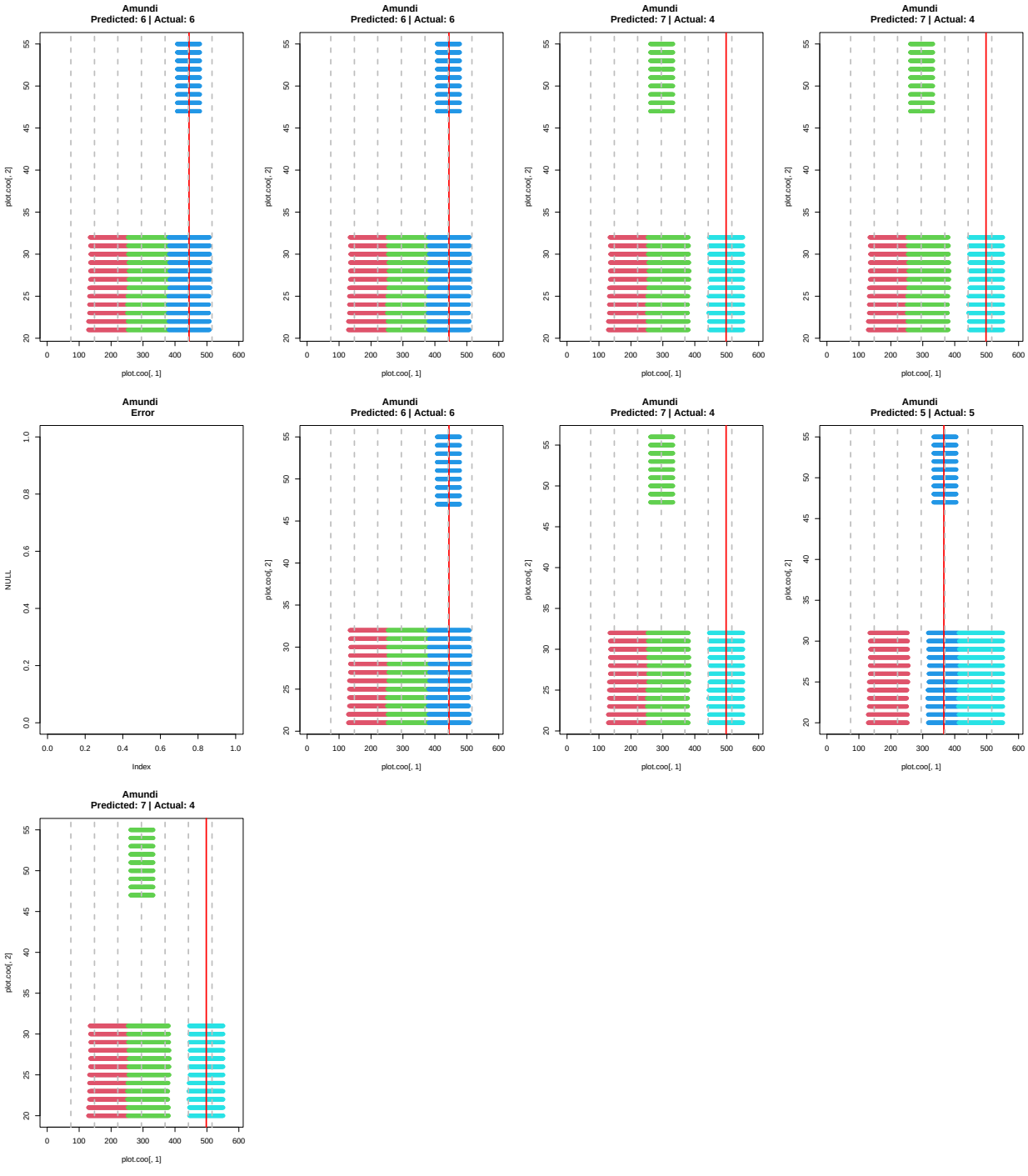
```
  # Map over dirs
  Map(function(x, y){

    # set
    {setwd(paste0("./../", 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 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(paste0("./../", 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
```

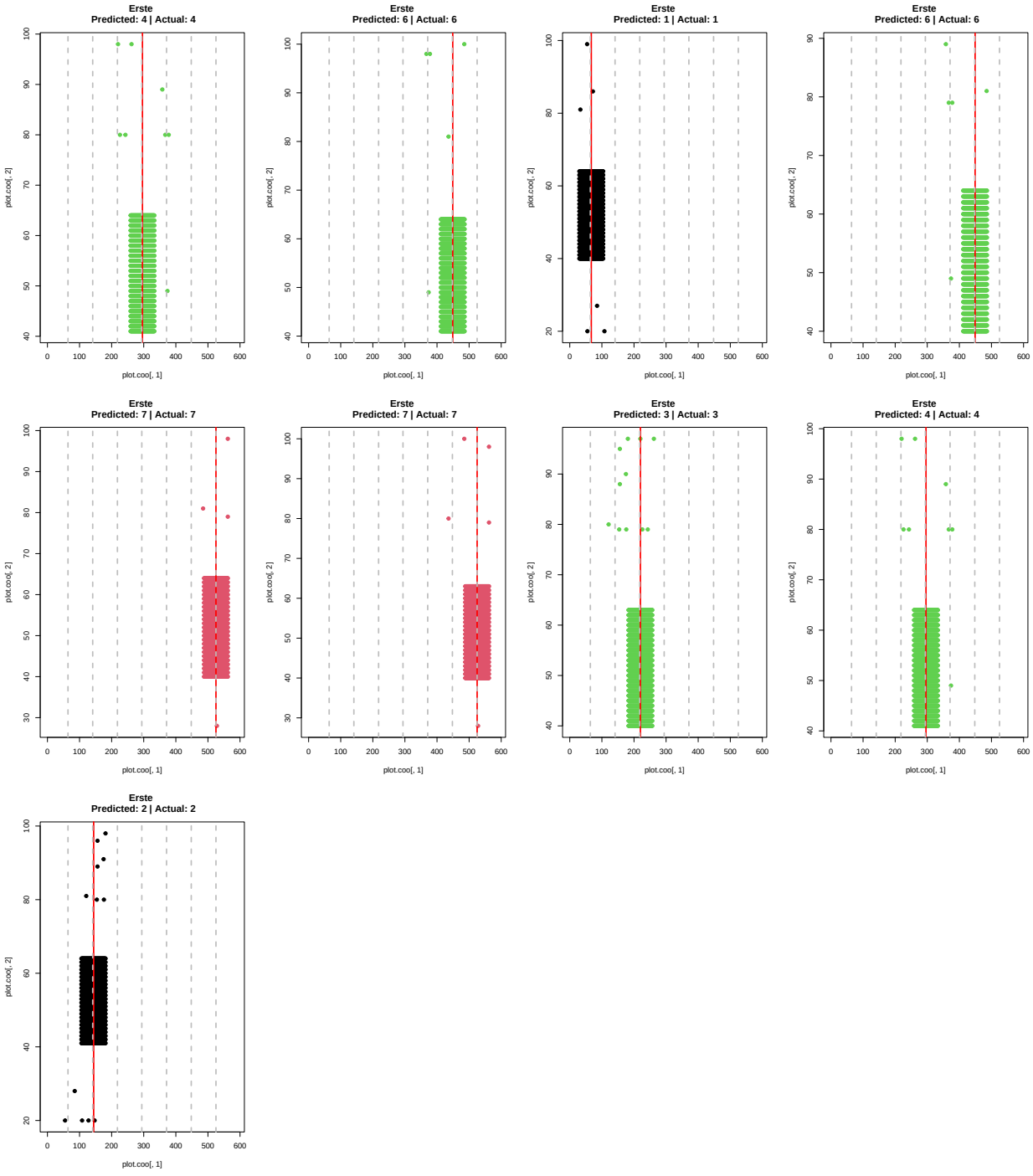
Now that we have identified all KIDs for which the extraction failed, we can proceed to see if the classification was correct for the remaining kids.

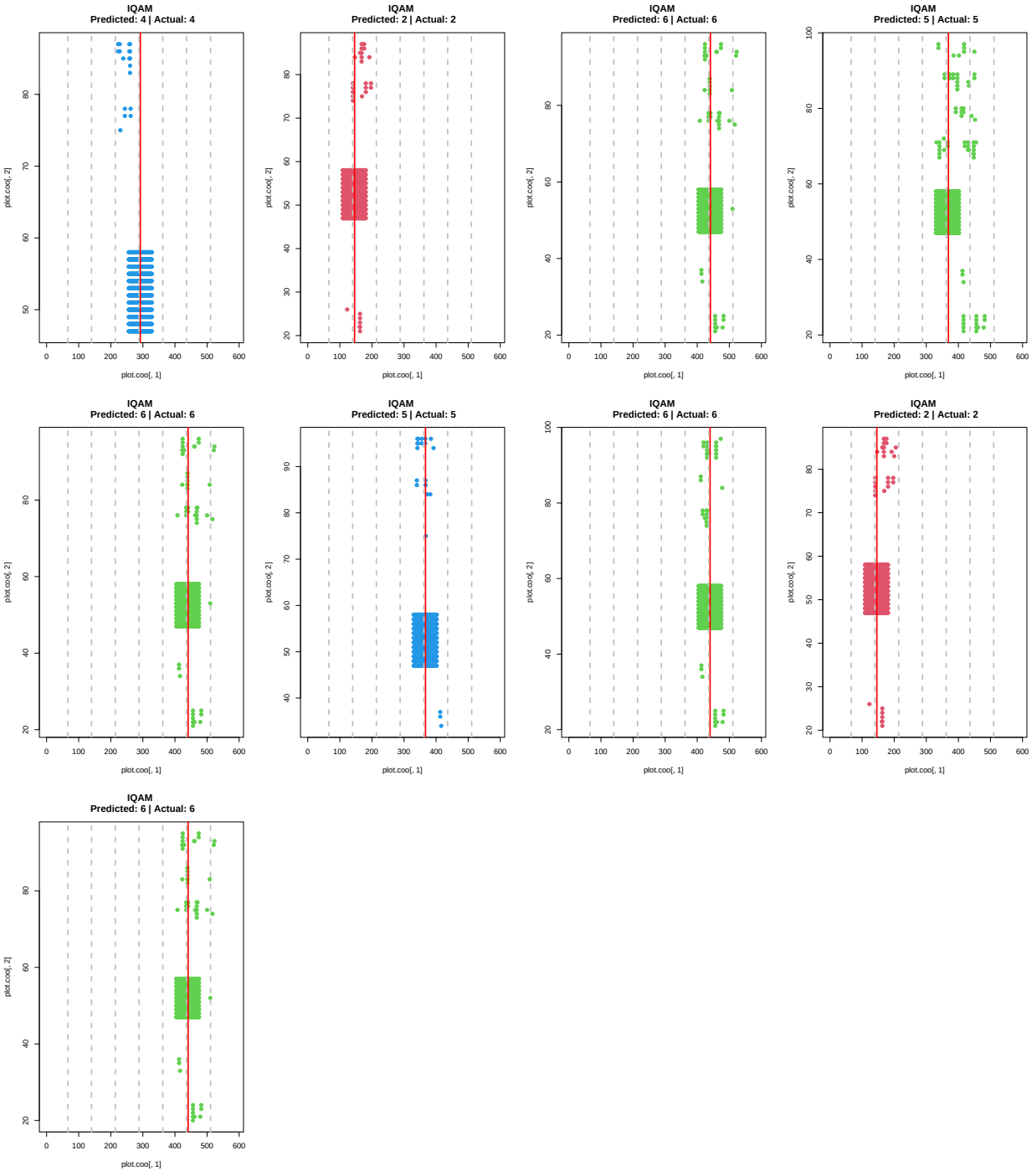
```r
# Plot
Map(function(x, y){

  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)

}, 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))
```

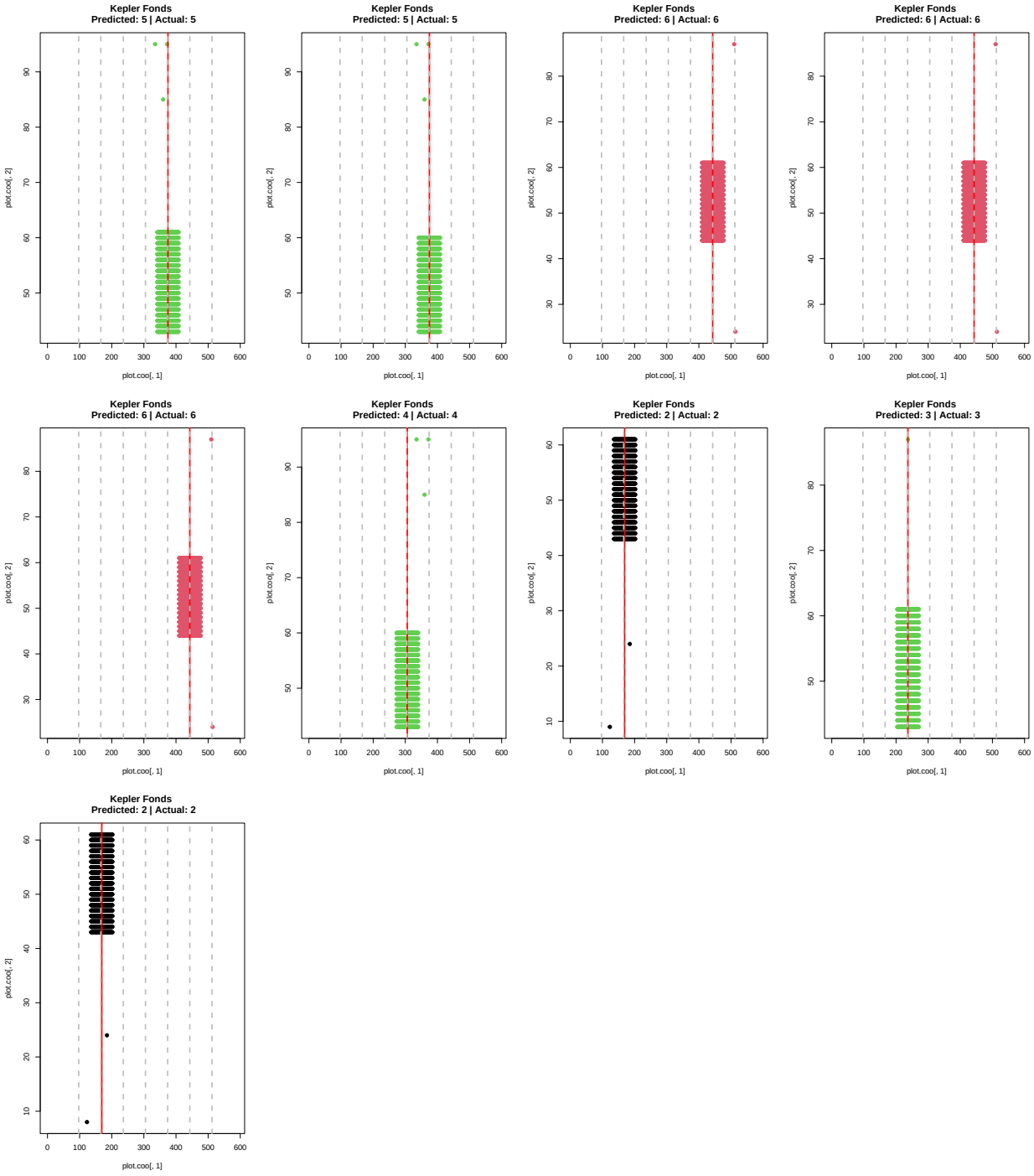
```
      }

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

}, test, tef)
```

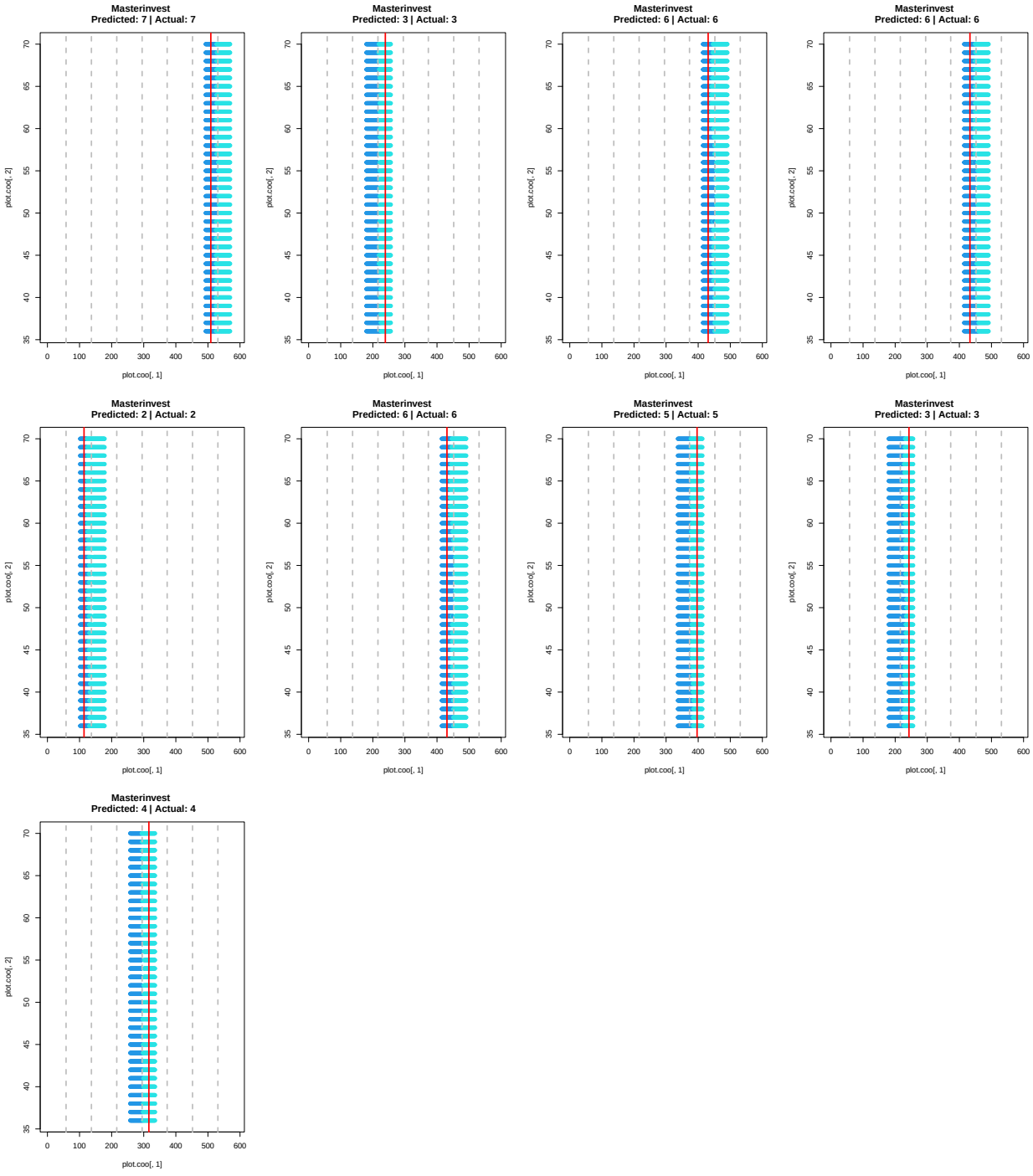


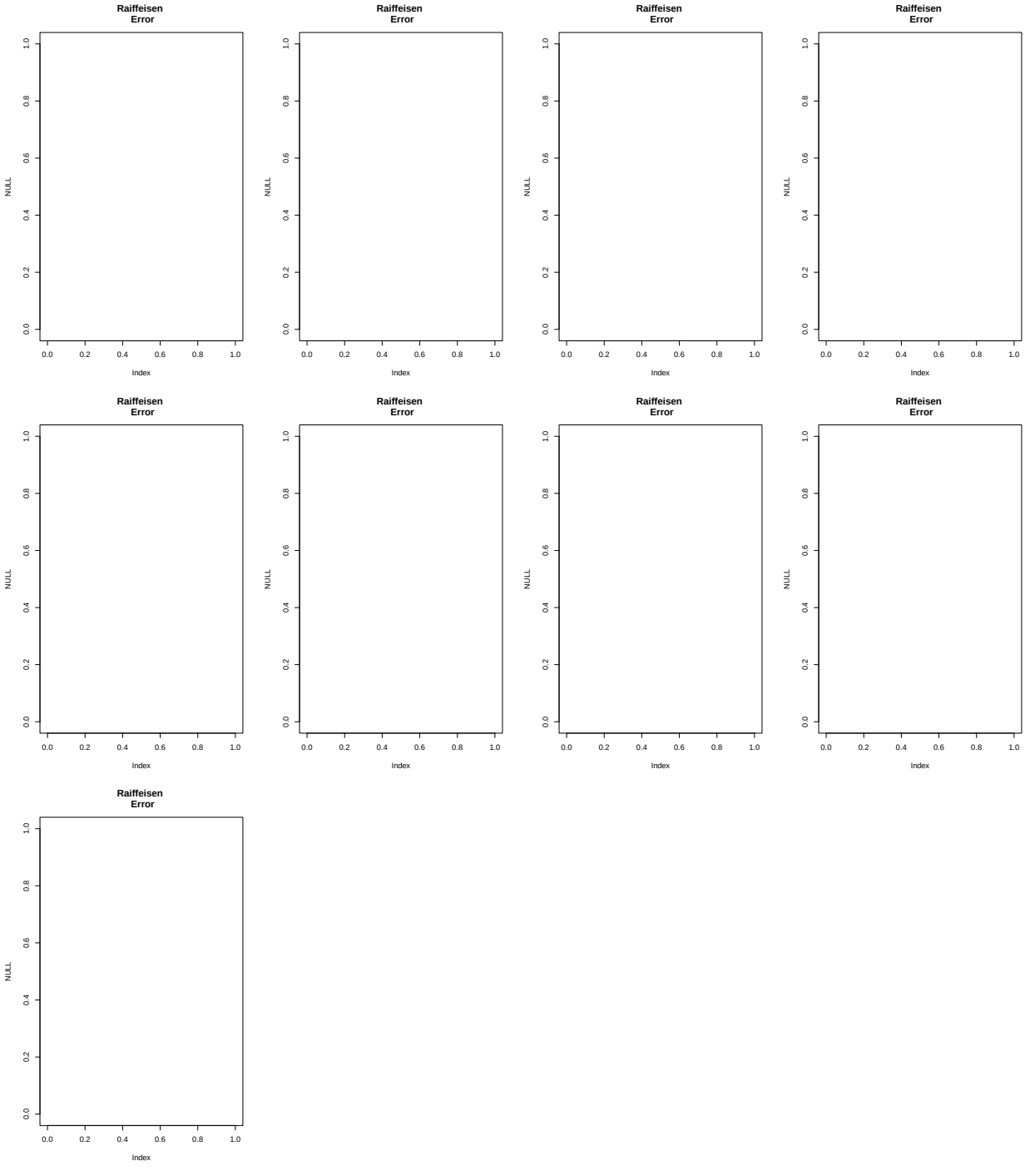

Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 7 | Actual: 4
Amundi
Predicted: 7 | Actual: 4
Amundi
Error
Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 7 | Actual: 4
Amundi
Predicted: 5 | Actual: 5
Amundi
Predicted: 7 | Actual: 4
plot.coo[, 1]
plot.coo[, 2]
Index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

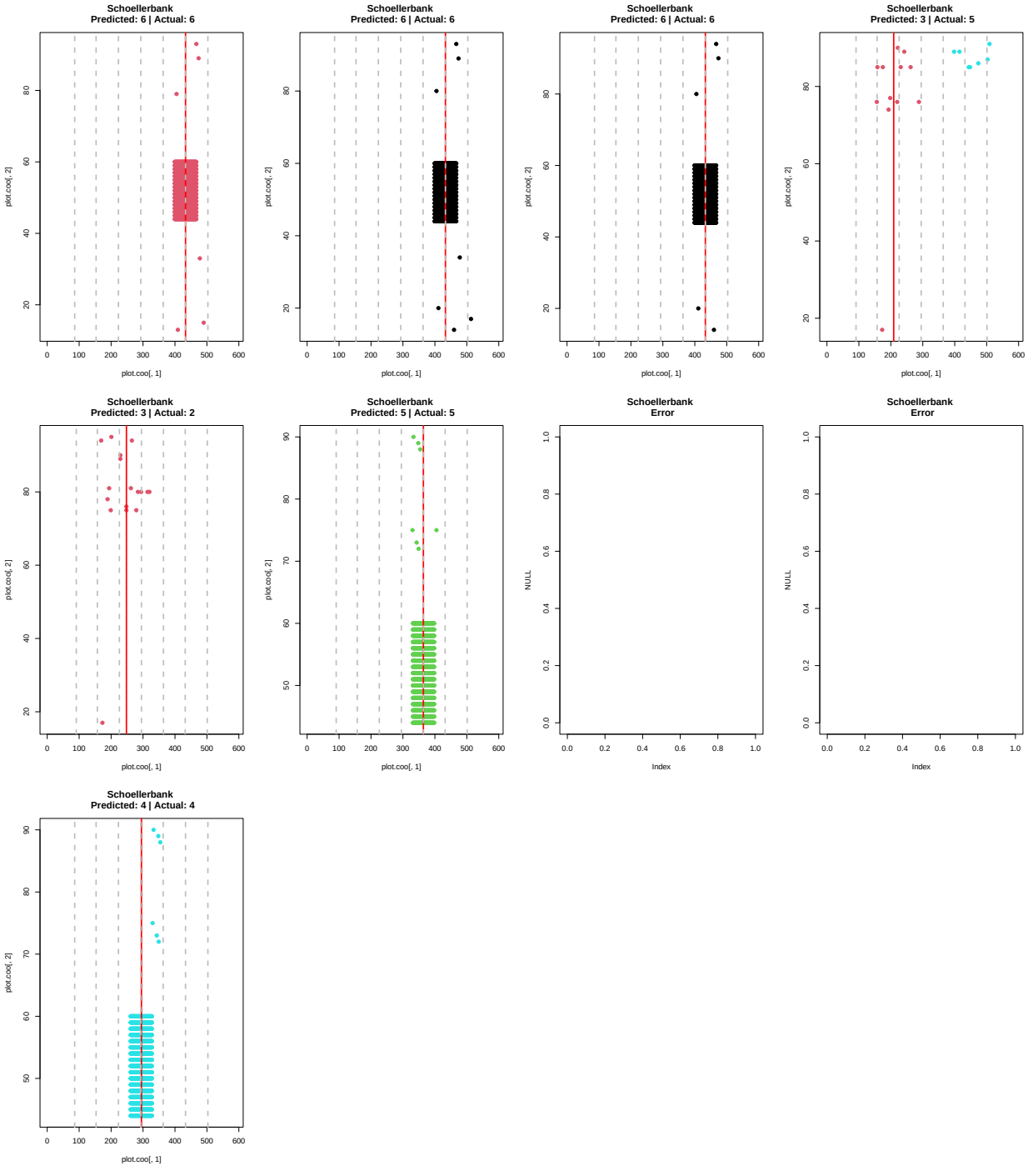

Erste
Predicted: 4 | Actual: 4
Erste
Predicted: 6 | Actual: 6
Erste
Predicted: 1 | Actual: 1
Erste
Predicted: 6 | Actual: 6
Erste
Predicted: 7 | Actual: 7
Erste
Predicted: 7 | Actual: 7
Erste
Predicted: 3 | Actual: 3
Erste
Predicted: 4 | Actual: 4
Erste
Predicted: 2 | Actual: 2
plot.coo[, 1]
plot.coo[,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

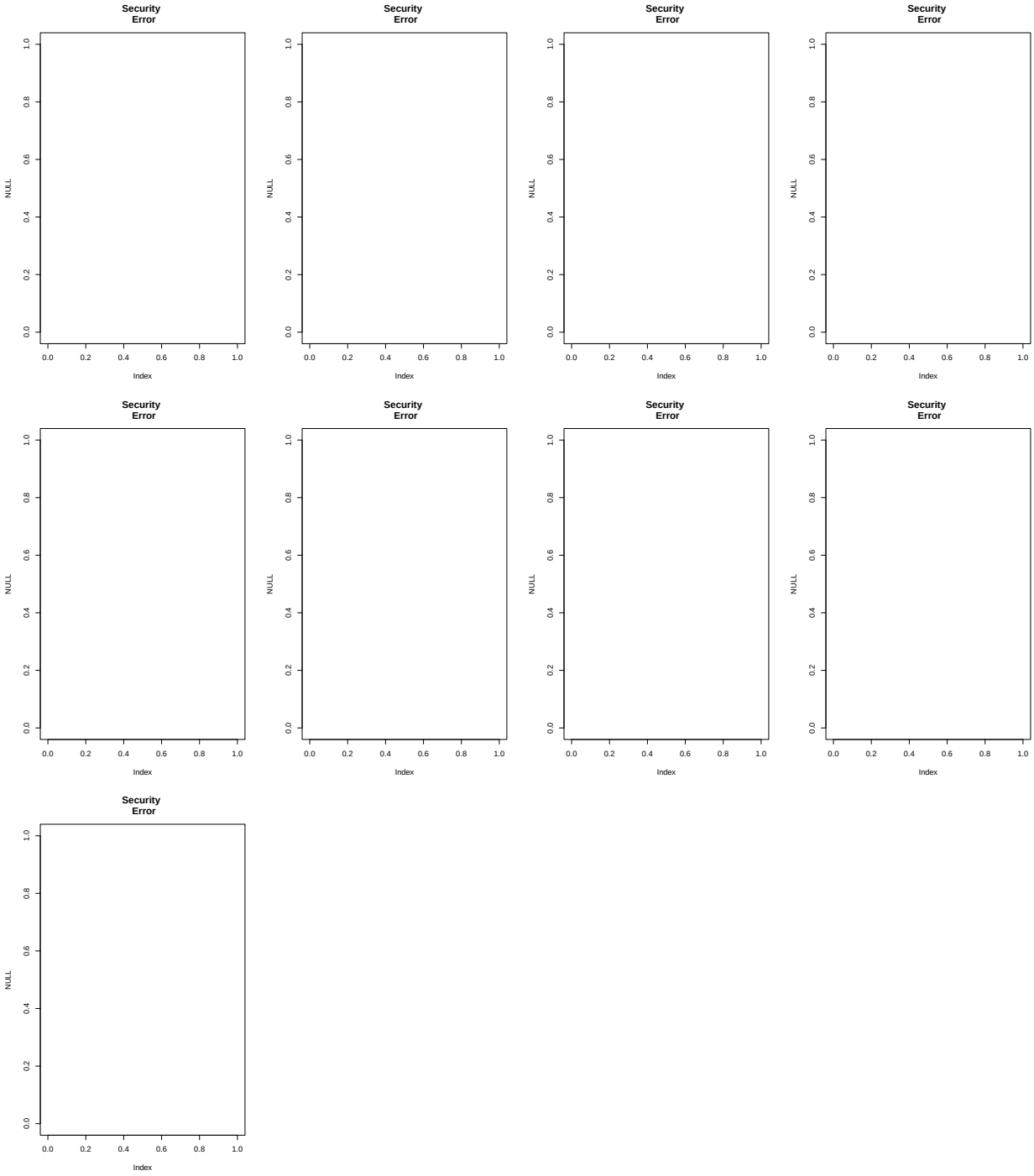

IQAM
Predicted: 4 | Actual: 4
IQAM
Predicted: 2 | Actual: 2
IQAM
Predicted: 6 | Actual: 6
IQAM
Predicted: 5 | Actual: 5
IQAM
Predicted: 6 | Actual: 6
IQAM
Predicted: 5 | Actual: 5
IQAM
Predicted: 6 | Actual: 6
IQAM
Predicted: 2 | Actual: 2
IQAM
Predicted: 6 | Actual: 6
plot.coo[, 1]
plot.coo[, 2]

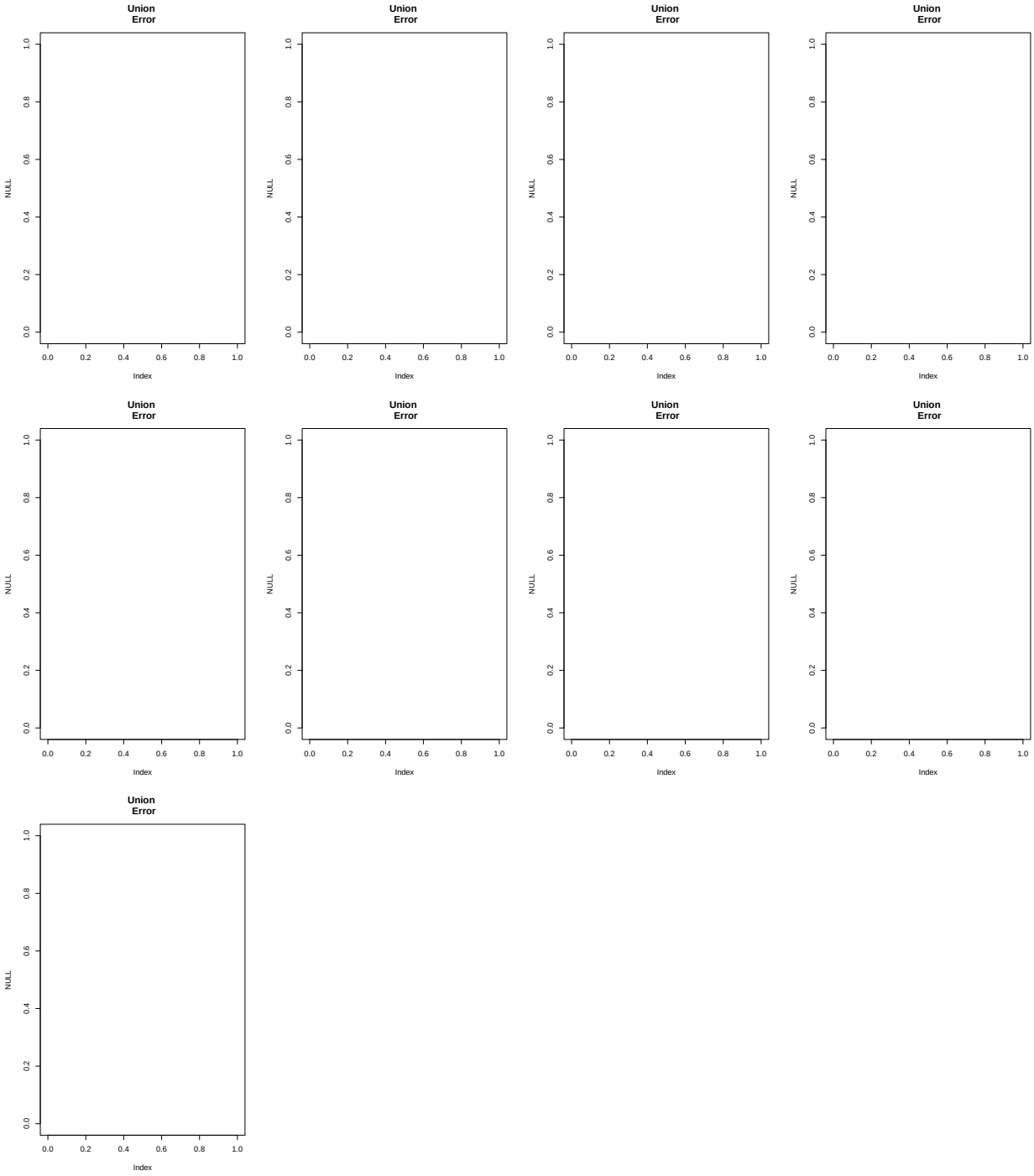

Kepler Fonds
Predicted: 5 | Actual: 5
Kepler Fonds
Predicted: 5 | Actual: 5
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 4 | Actual: 4
Kepler Fonds
Predicted: 2 | Actual: 2
Kepler Fonds
Predicted: 3 | Actual: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


Security
Error
NULL
Index
0.0
0.2
0.4
0.6
0.8
1.0


Union
Error
NULL
Index
0.0
0.2
0.4
0.6
0.8
1.0

4.4 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
  - 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
  - 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.

5 Naive Extraction

This approach is quite a bit is simpler and less error-prone than the previous one. Additionally, it does not require a color specification to extract the SRRI given a PDF. The idea is to cut out the rectangle around each scale entry to then identify if the color is different to the background of the remaining pdf.

```
# setwd to file that contains KIDs
setwd("./../../KIDs/Allianz")

# tover KAGs
mapply(\(x){

  # change dir to respective KAG
  {setwd(paste0("./../", x))

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

  # FUN over all .pdfs
  sapply(file_nom, \(z){

    # extract
    SRRI_ext_rec(doc = z, pass = TRUE) |> try()

  }) |> (\(y) sapply(y, \(t) ifelse(length(t) > 0, t, NA)))() # integer(0) to NA

}, dirs) -> pred_naiv

# performance
pred_un <- pred_naiv |> unlist() |> unname()

# NAs
pred_un |> is.na() |> mean()
```

```
## [1] 0.2333333
```

```
# Performance for all non NAs
(pred_un == as.numeric(dat.valid.SRRI[, "SRRI"])) |> mean(na.rm = TRUE)
```

```
## [1] 1
```

5.1 Improvement of SRRI_ext_loc()

So far only the average link for 5 final clusters was thoroughly tested. Accordingly, now a tuning grid will be generated to see if the performance can be improved by changing some of the input parameters.

```
# input parameter grid
# tgrid <- expand.grid("tol" = c(20, seq(30, 60, 15)), "p" = seq(4, 8, 2), "method" = c("single", "aver

# also alter
tgrid <- expand.grid("tol" = 20, "p" = seq(4, 8, 2),
                     "method" = c("single", "average", "complete"), "co" = c(0.05, 0.1, 0.2, 0.3))


## ran once ##
```

```r
  # wd
  # setwd to file that contains KIDs
  #  setwd("./../../KIDs/Allianz")

  # storage
  # lst_store <- list()

  # loop over tgrid rows
  # for (i in 1:nrow(tgrid)){
  #
  #   mapply(\(x, y){
  #
  #     # change dir to respective KAG
  #     {setwd(paste0("./../", x))
  #
  #     # pdfs
  #     file_nom <- list.files(pattern = ".pdf")}
  #
  #     # FUN over all .pdfs
  #     lapply(file_nom, \(z){
  #
  #       # extract
  #       SRRI_ext_loc(doc = z, col = y, tol = tgrid[i, "tol"], p = tgrid[i, "p"], method = tgrid[i, "m
  #
  #     })
  #
  #   }, dirs, list.col.KAG) -> lst_store[[i]]
  # }

# save
# saveRDS(lst_store, "./../../KIDs/Auxiliary/SRRI_ext_loc_perf.rds")

# read
lst_store <- readRDS("./../../KIDs/Auxiliary/SRRI_ext_loc_perf.rds")

# loop over params
sapply(lst_store, \(x){

  # use coercion to remove Error messages
  per_vec <- as.numeric(x) |> suppressWarnings()

  # performance
  per <- (per_vec == as.numeric(dat.valid.SRRI[, "SRRI"])) |> mean(na.rm = TRUE)

  # nas
  nas <- per_vec |> is.na() |> mean()

  # ret
  cbind(per, nas)

}) |> t() -> per_SRRI_ext_loc

# names
```

```
colnames(per_SRRI_ext_loc) <- c("Per", "NAs")

# bind to tgrid to see corresponing paramter performance
para_per <- cbind(tgrid, per_SRRI_ext_loc)

# order
para_per_ordered <- para_per[order(para_per[, "Per"], para_per[, "NAs"], decreasing = c(TRUE, FALSE)), ]
```

5.2 Out of Sample Performance

5.2.1 Naive Approach

```
# setwd to file that contains KIDs
setwd("./../../KIDs/Allianz/Test")

# tover KAGs
mapply(\(x){

  # change dir to respective KAG
  {setwd(paste0("./../../", x, "/Test"))

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

  # FUN over all .pdfs
  sapply(file_nom, \(z){

    # extract
    SRRI_ext_rec(doc = z) |> try()

  }) |> (\(y) sapply(y, \(t) ifelse(length(t) > 0, t, NA)))() # integer(0) to NA

}, dirs) -> pred_naiv_test

# performance
pred_un_t <- pred_naiv_test |> unlist() |> unname()

# NAs
pred_un_t |> is.na() |> mean()
```

```
## [1] 0.2
```

```
# Performance for all non NAs
(pred_un_t == as.numeric(dat.valid.SRRI_test[, "SRRI"])) |> mean(na.rm = TRUE)
```

```
## [1] 1
```

5.2.2 Cluster Approach

```
# setwd to file that contains KIDs
setwd("./../../KIDs/Allianz/Test")

# over dirs amd colors
mapply(\(x, y){
```

```
  # change dir to respective KAG
  {setwd(paste0("./../../", x, "/Test"))

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

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

    # extract
    SRRI_ext_loc(doc = z, col = y, tol = 20, p = 4, method = "average")[[2]] |> try()

  })

}, dirs, list.col.KAG) -> testres
```

```
## 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 = 20, p = 4, method = "average") :
##   Error: No pixels of given color detected.
## Error in fastcluster::hclust(dist(coo, method = "euclidean"), method = method) :
##   N must be at least 2.
```

```
# performance
pred_un_ct <- testres |> unlist() |> unname()

# NAs
pred_un_ct |> as.numeric() |> is.na() |> mean()
```

```
## Warning in mean(is.na(as.numeric(pred_un_ct))): NAs durch Umwandlung erzeugt
```

```
## [1] 0.3
```

```
# Performance for all non NAs
(pred_un_ct == as.numeric(dat.valid.SRRI_test[, "SRRI"])) |> mean(na.rm = TRUE)
```

```
## [1] 0.7
```

The accuracy for documents that are not partly or completely scanned is 100%. This is definitely a performance increase when comparing to the agglomerative classification.

5.3 Workaround for scanned cases