

KID Function

Fabian Blasch

22.08.2021

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 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),
```

```

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)

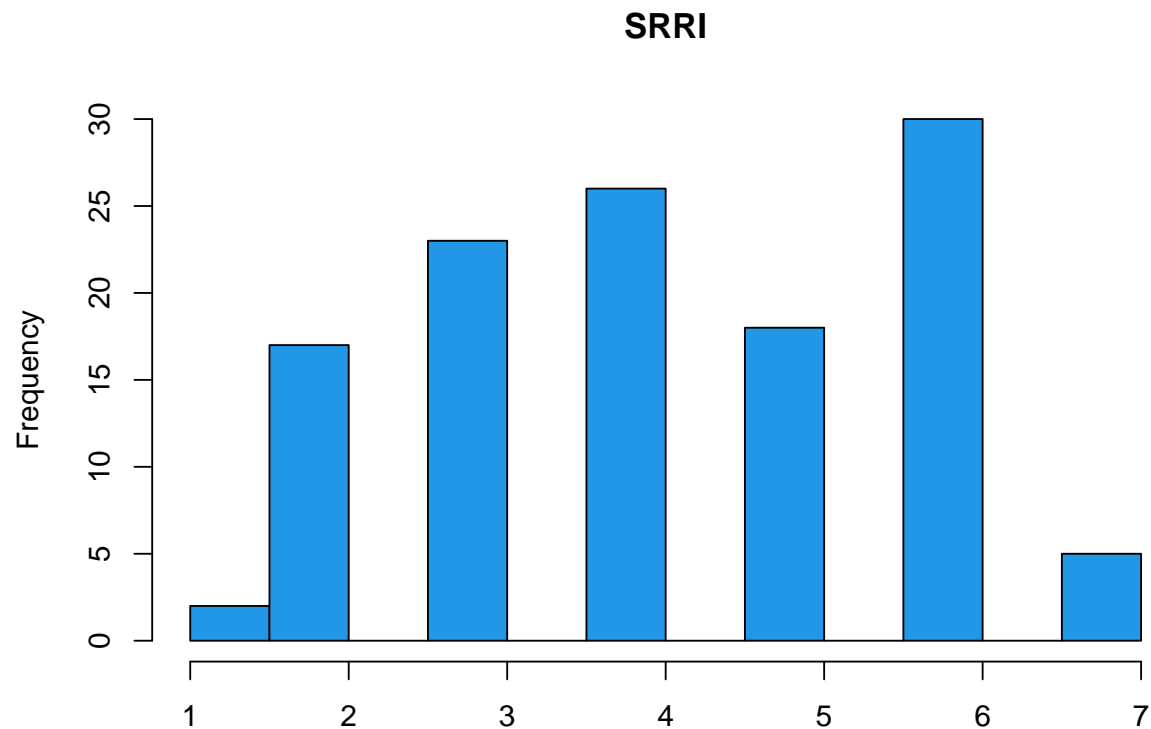
##           KAG           KID SRRI
## 1 Allianz ki-allakt_6.pdf      6
## 2 Allianz ki-allap_6.pdf      6
## 3 Allianz ki-alleur_2.pdf      2
## 4 Allianz ki-allna_6.pdf      6
## 5 Allianz ki-allnar_2.pdf      2
## 6 Allianz ki-allore_3.pdf      3
## 7 Allianz ki-allost_6.pdf      6

# dim
dim(dat.valid.SRRI)

## [1] 121   3

# 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/blasco/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
```

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.

```
# source function
source("C:/Users/blasch/OneDrive/Documents/GitHub/KID/Code/Functions/SRRI_ext.R")
```

Tests

Starting with one KAG.

Erste

```
# set wd to file that contains
setwd("C:/Users/blasch/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, y){

  # set
  {setwd("C:/Users/blasch/OneDrive/Documents/GitHub/KID/KIDS")
    setwd(x)}

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

  # FUN over all .pdfs
  lapply(file_nom, function(z){
    SRRI_ext(doc = z, col = y)
  })

}, dirs[3], col[3, 2]) -> erste.test

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

```

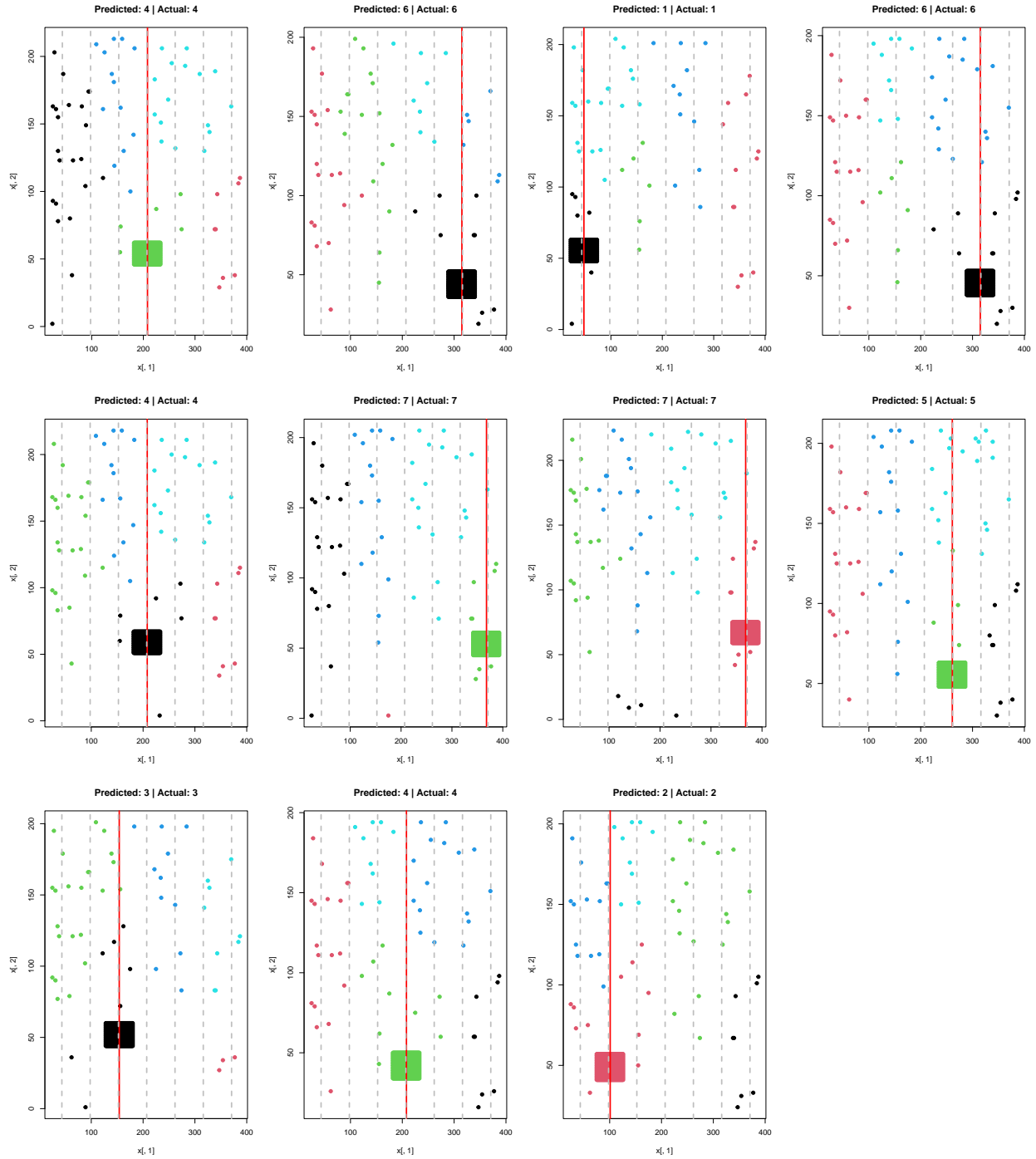
par(mfrow = c(3, 4))

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

  {plot(x[, 1], x[, 2], col = x[, ncol(x)], pch = 19, main = paste("Predicted:", z, "| Actual:", l))
  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))

```



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
```

```

{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(doc = z, col = y), silent = F)

})

}, dirs, col[, 2]) -> test

, type = "message")

## Error in SRRI_ext(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in pos.vec[page.SRRI] - off :
##   nicht-numerisches Argument für binären Operator
## Error in SRRI_ext(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext(doc = z, col = y) :

```



```

# 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 erros.")
  }

}), 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.

```

# Plot
Map(function(x, y){

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

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

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

```

```

par(mfrow = c(3, 4))

# plot

# over KAGs
invisible(Map(function(m, n){

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

  # over KIDS
  invisible(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,
           main = paste(fund, "\n", "Predicted:", pred, "| Actual:", act))

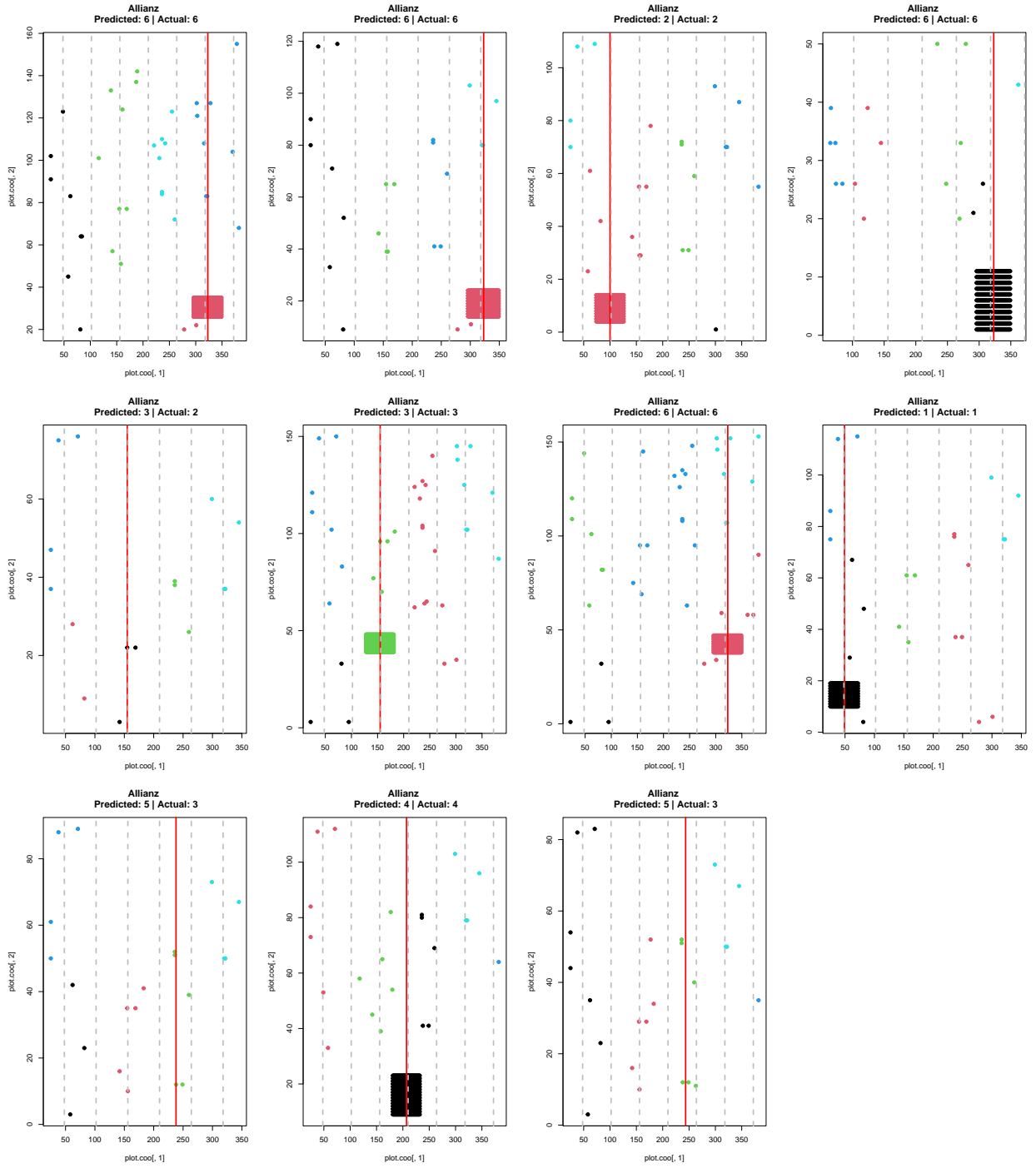
      # median
      abline(v = med, col = "red", lty = 1, lwd = 2)

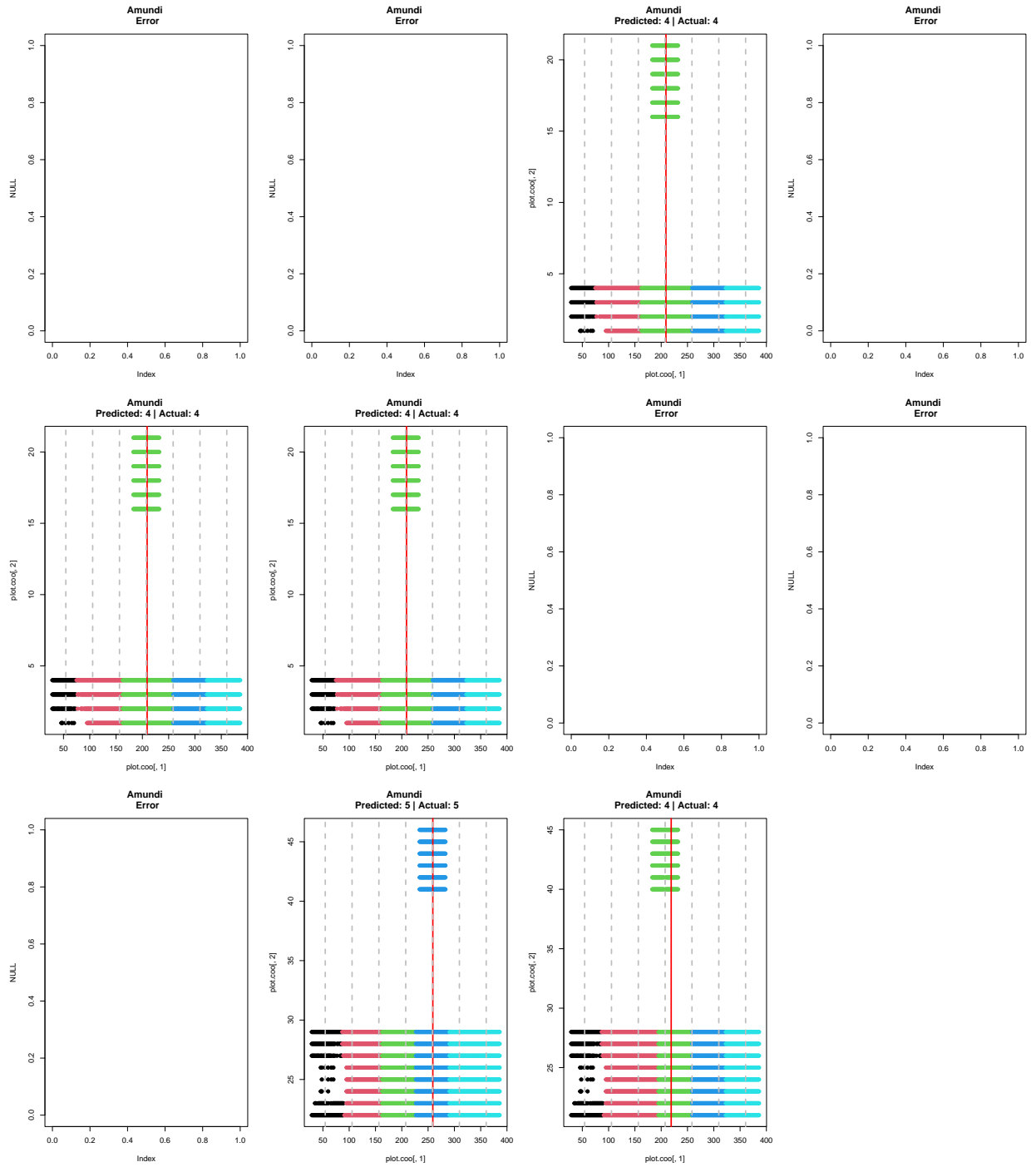
      # Scale
      lapply(scal, function(s) abline(v = s, col = "grey", lwd = 2, lty = 2))
    }

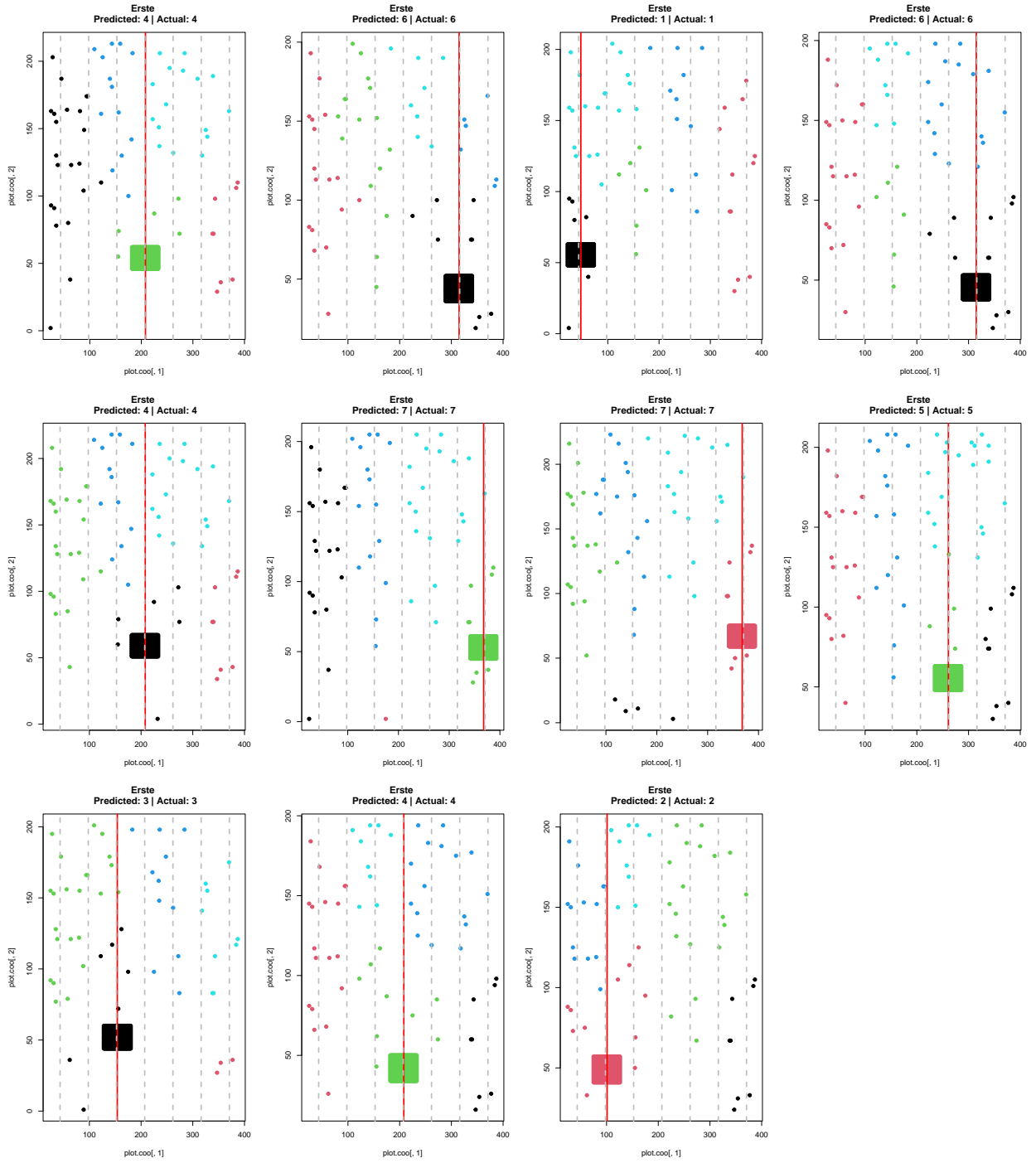
  }, m, n[, 4], n[, 3], n[, 1])

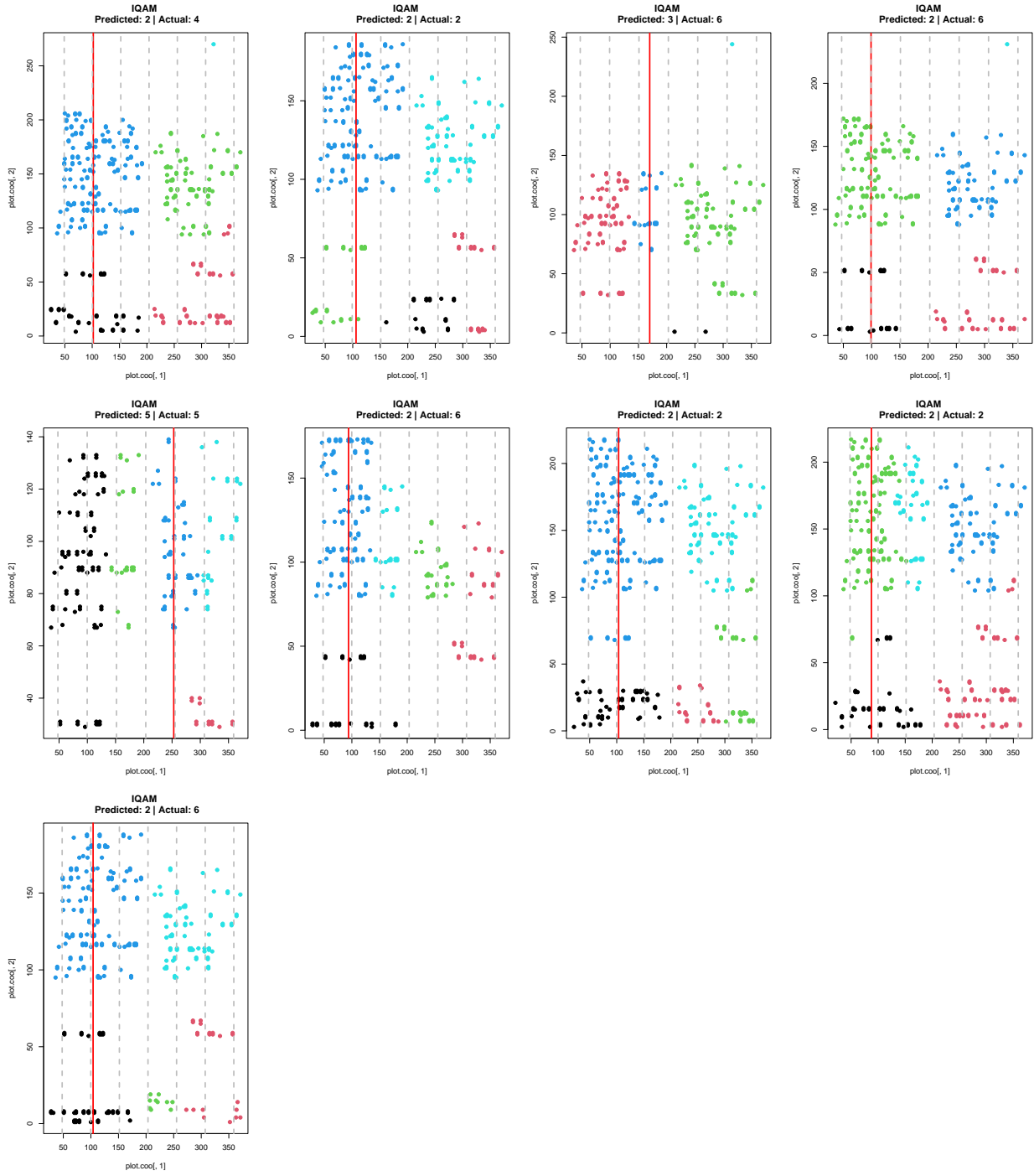
}, test, tef)

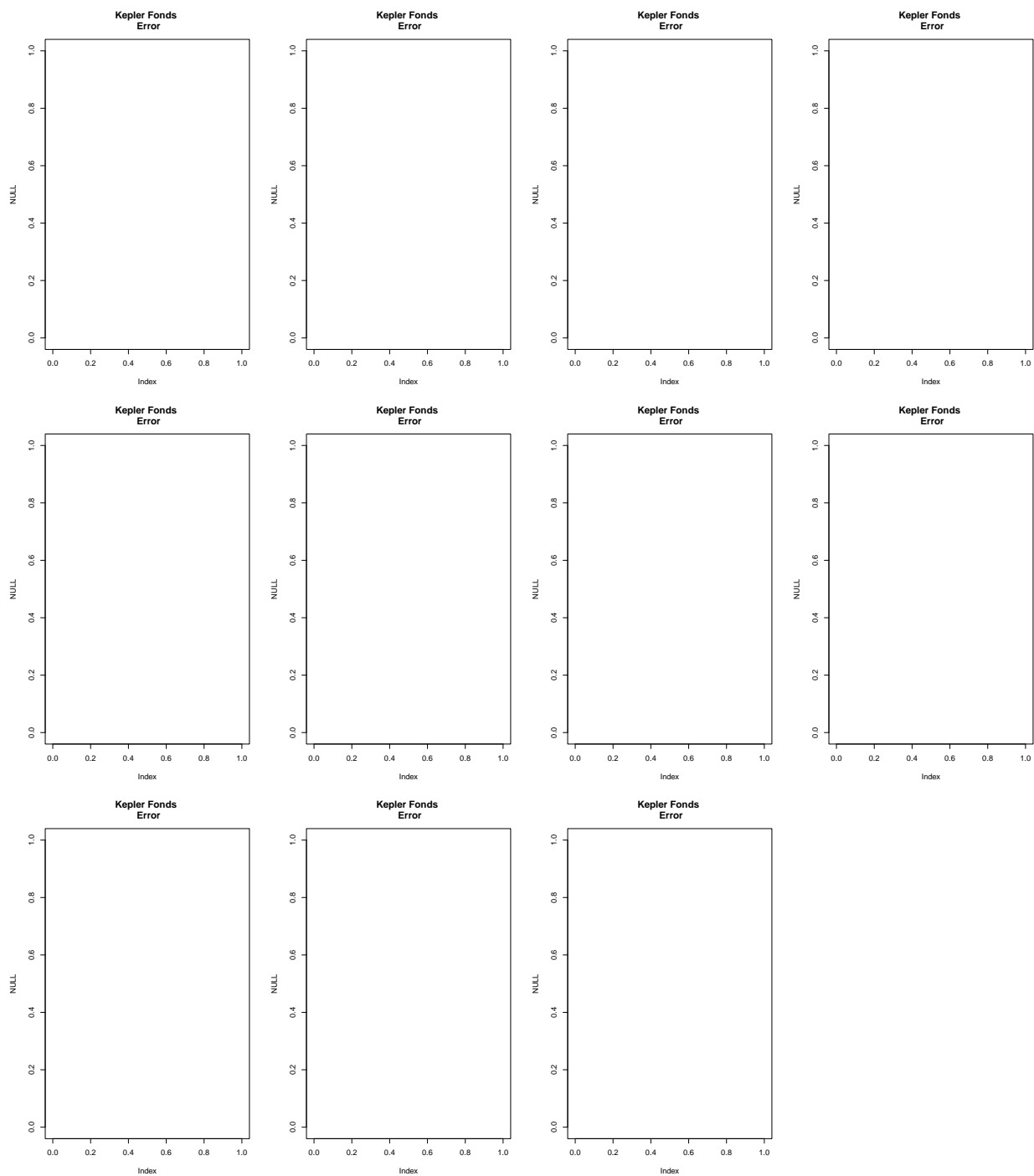
```

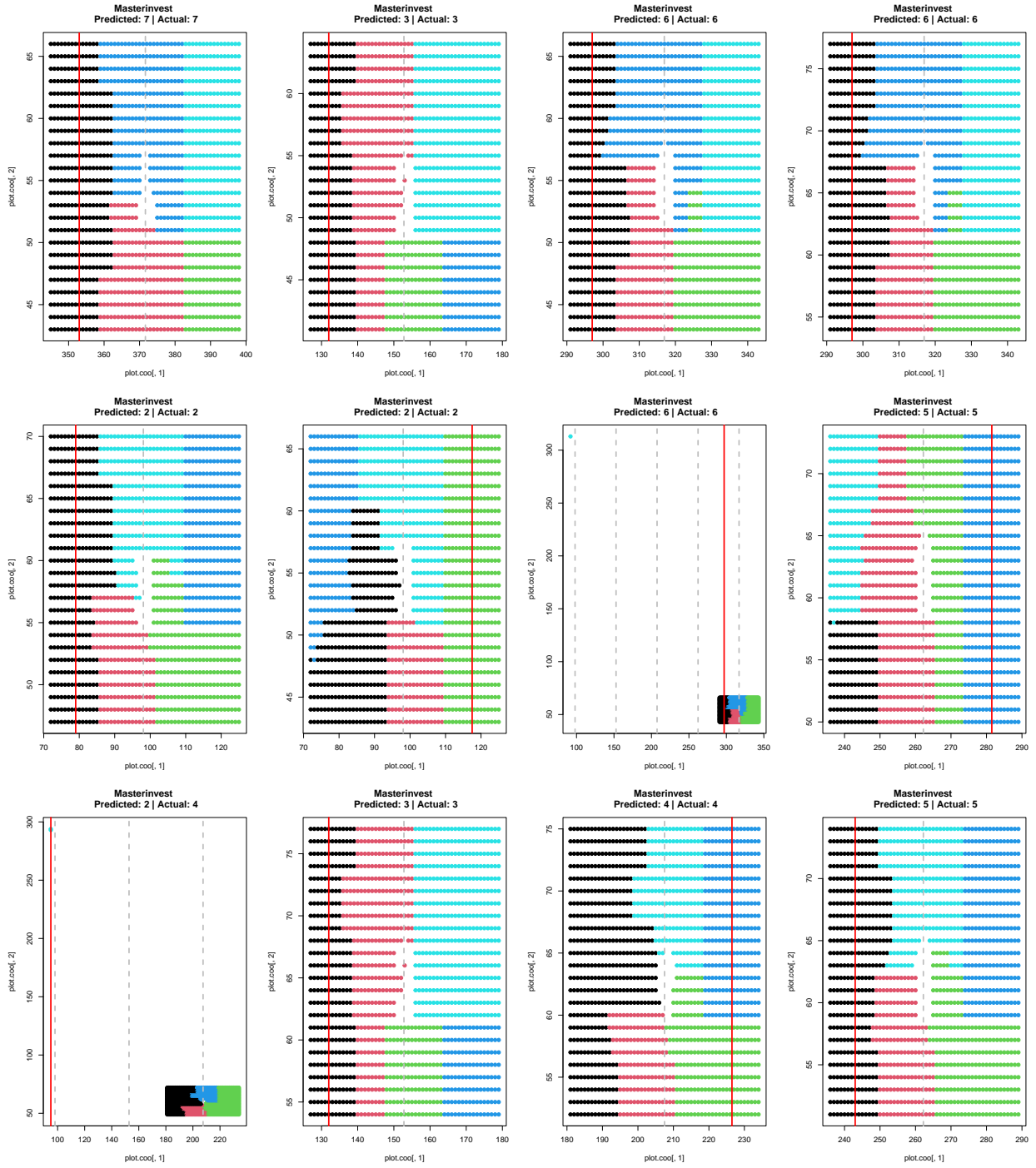


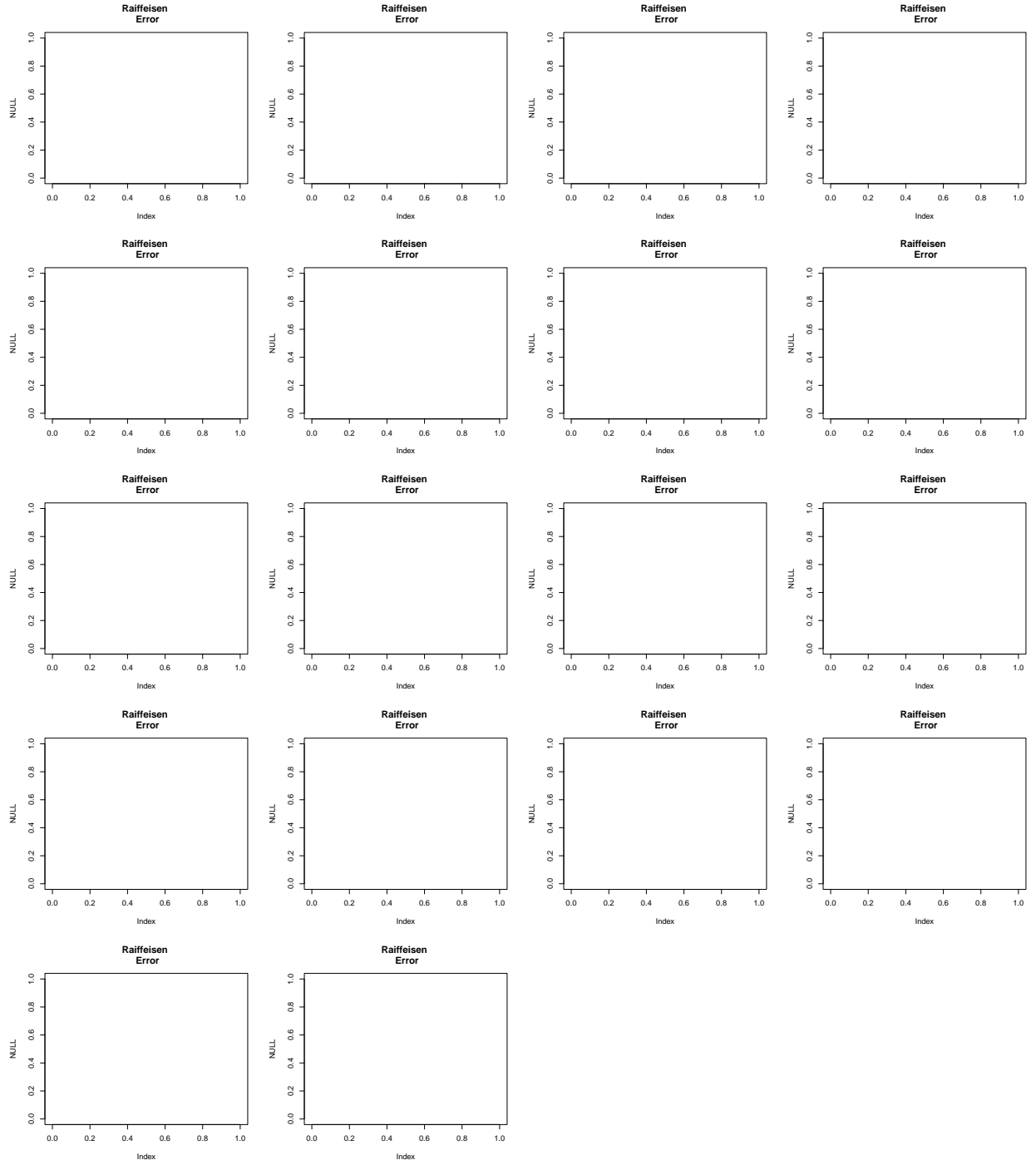


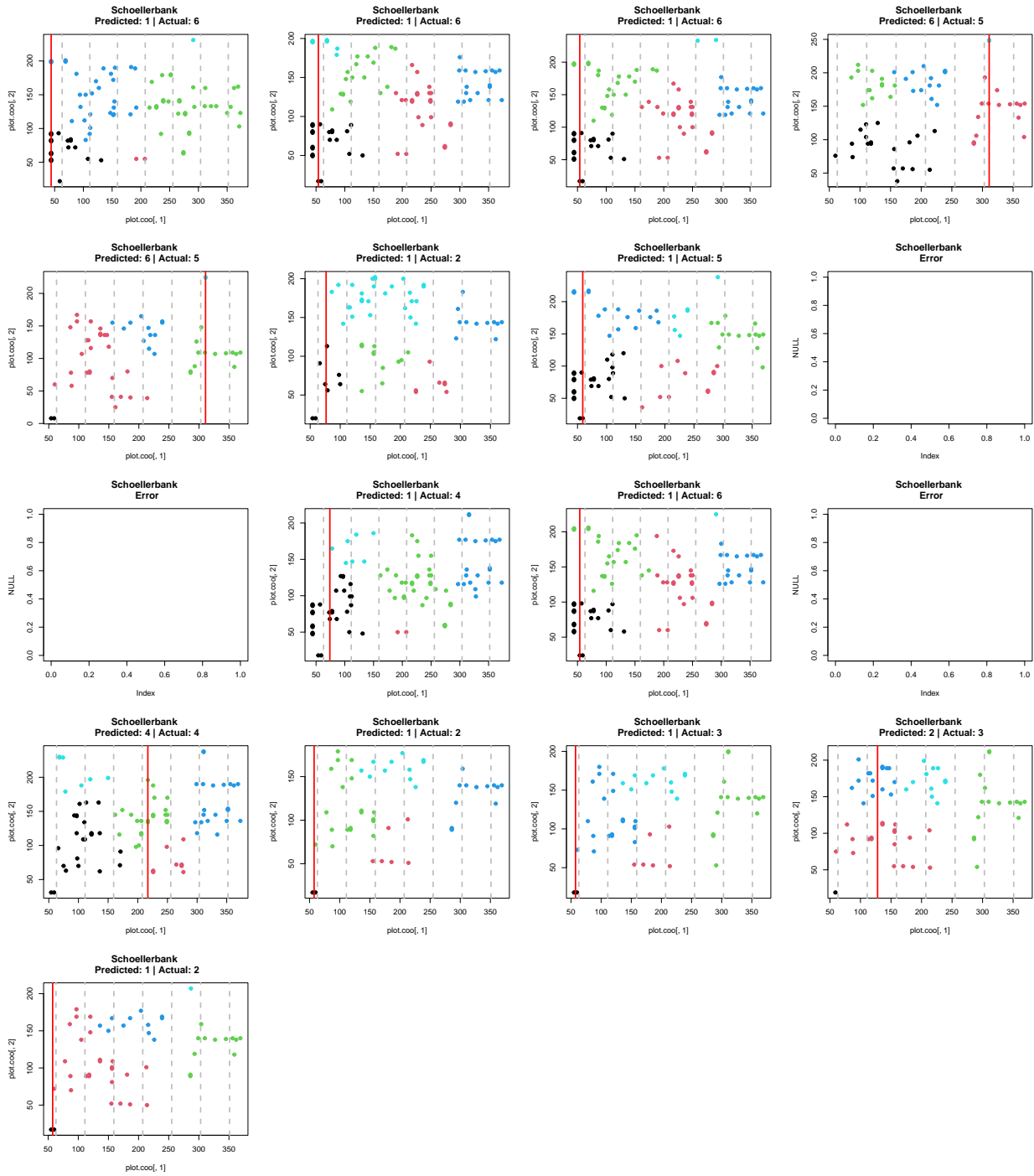


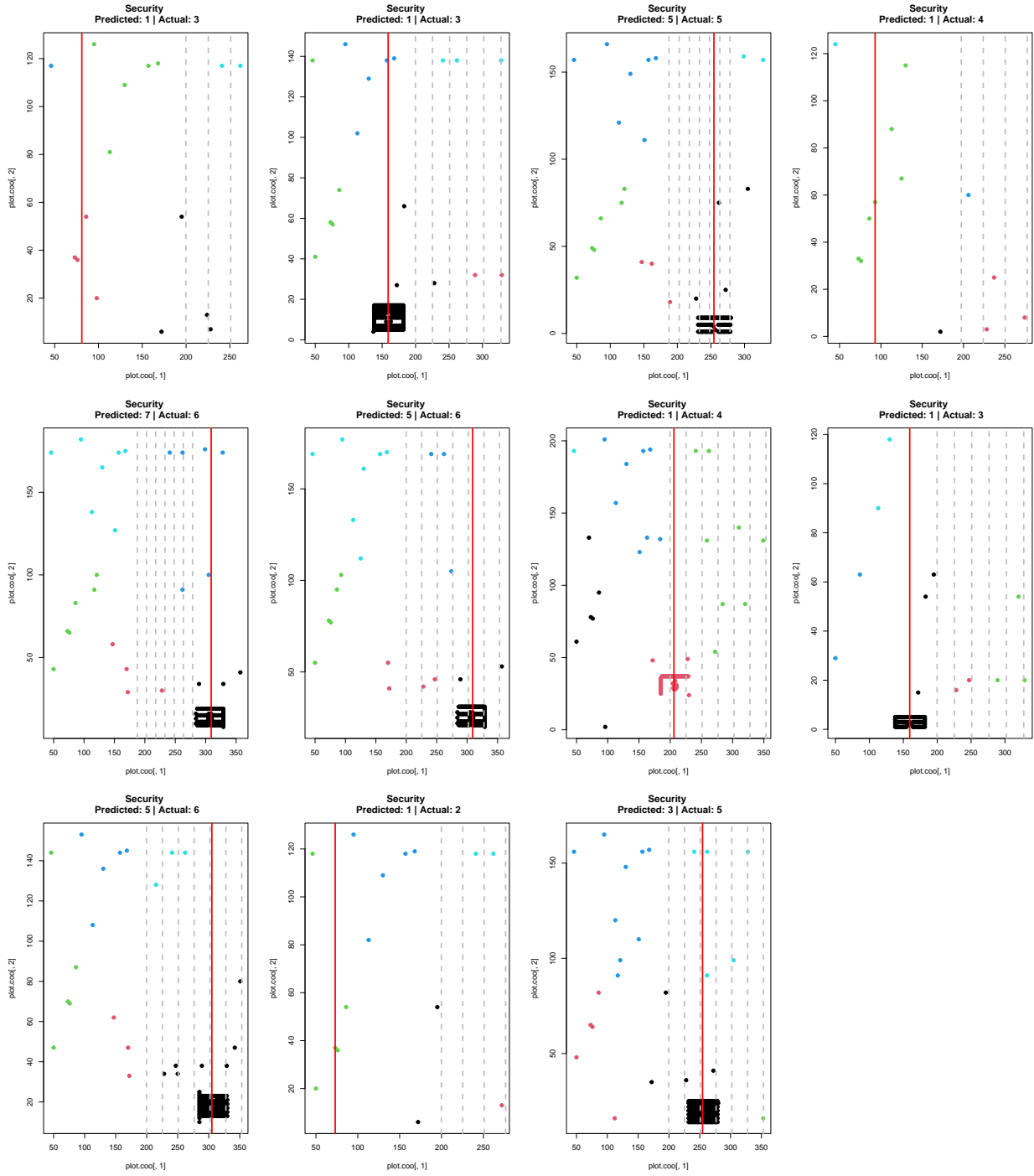


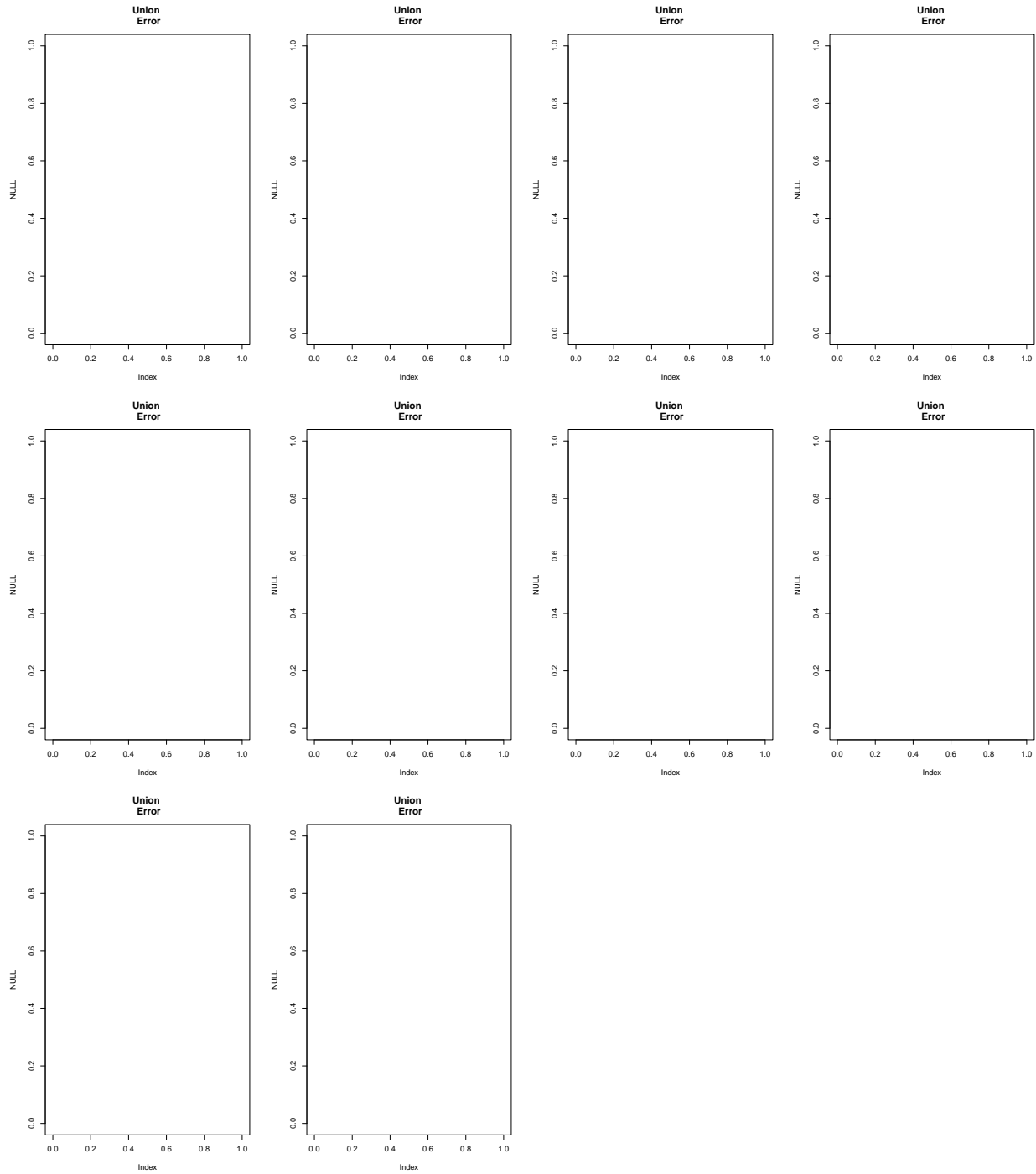












By KAG

- Alianz
 - Read out worked quite well for all cases in which the rectangle was identified, unfortunately it seems like in three cases this was not the case
 - Check cutoff
 - Check color
- Amundi
 - The import of Amundi's PDFs is associated with two Errors: Could not detect the SRRI text and

- could not detect default color. Both are not immediately visually apparent when looking at the PDFs, further examination is required
- Seemingly a lot of noise in the target color from right above the SRRI graph. This is not a severe problem as the noise is generated by a bar that does not change over the width of the page , i.e all PDFs that were read in without an error where classified correctly.
 - Erste
 - Worked brilliantly!
 - IQAM
 - No box detected in given color
 - Check HEX code.
 - Kepler Fonds
 - requires separate run specified error source is unknown
 - Masterinvest
 - Very heterogenic results, some PDFs display extreme amount of noise and no visually detectable box.
 - check color
 - check lsm / rsm
 - Raiffeisen
 - Check color
 - Check last three PDFs for text detection issues
 - Schoellerbank
 - No box detected, check color
 - three PDFs threw and error all because of text detection.
 - Security
 - Scale is completely off (generally think about using all pixels for margin not only black)
 - rectangle is missing in some files
 - check cutoff
 - check color
 - Union
 - check SRRI text detection