KID
Function

Fabian Blasch

05.09.2021

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),
```

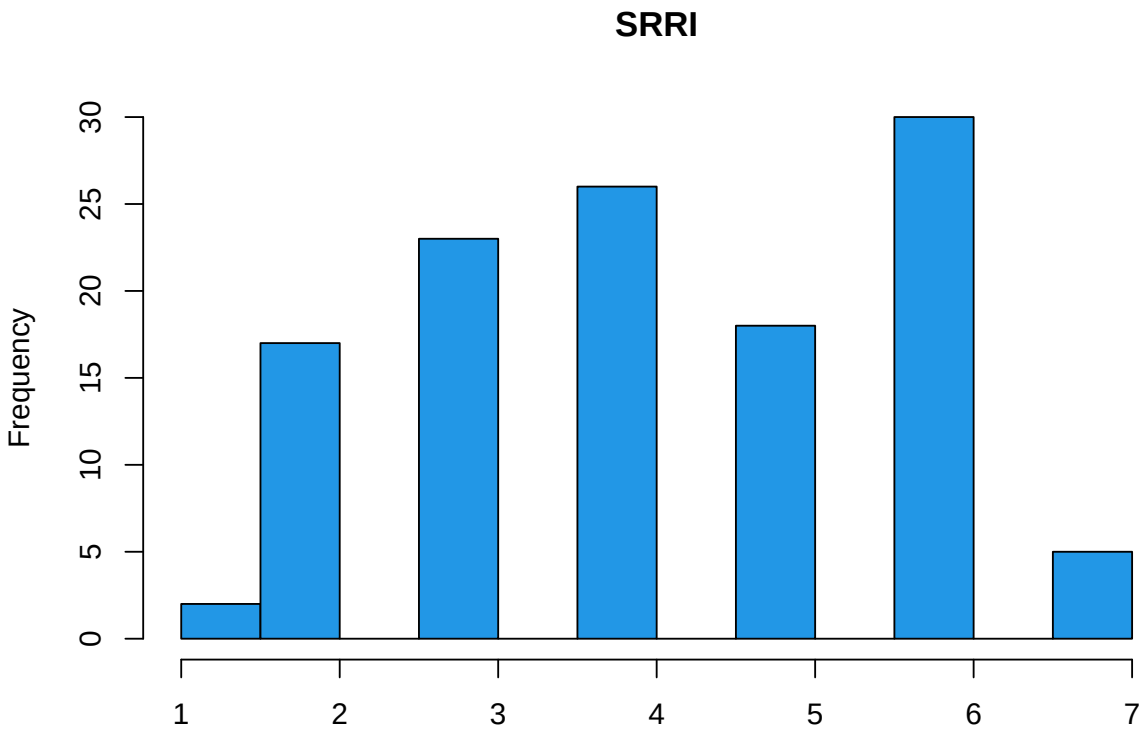
```
                                        function(x) x[length(x)]), ".", fixed = T), "[", 1)))

# 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/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
```

```
# 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])
```

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 functions
source("C:/Users/blasc/OneDrive/Documents/GitHub/KID/Code/Functions/SRRI_ext.R")
source("C:/Users/blasc/OneDrive/Documents/GitHub/KID/Code/Functions/SRRI_ext_test.R")
source("C:/Users/blasc/OneDrive/Documents/GitHub/KID/Code/Functions/SRRI_ext_fast.R")
```

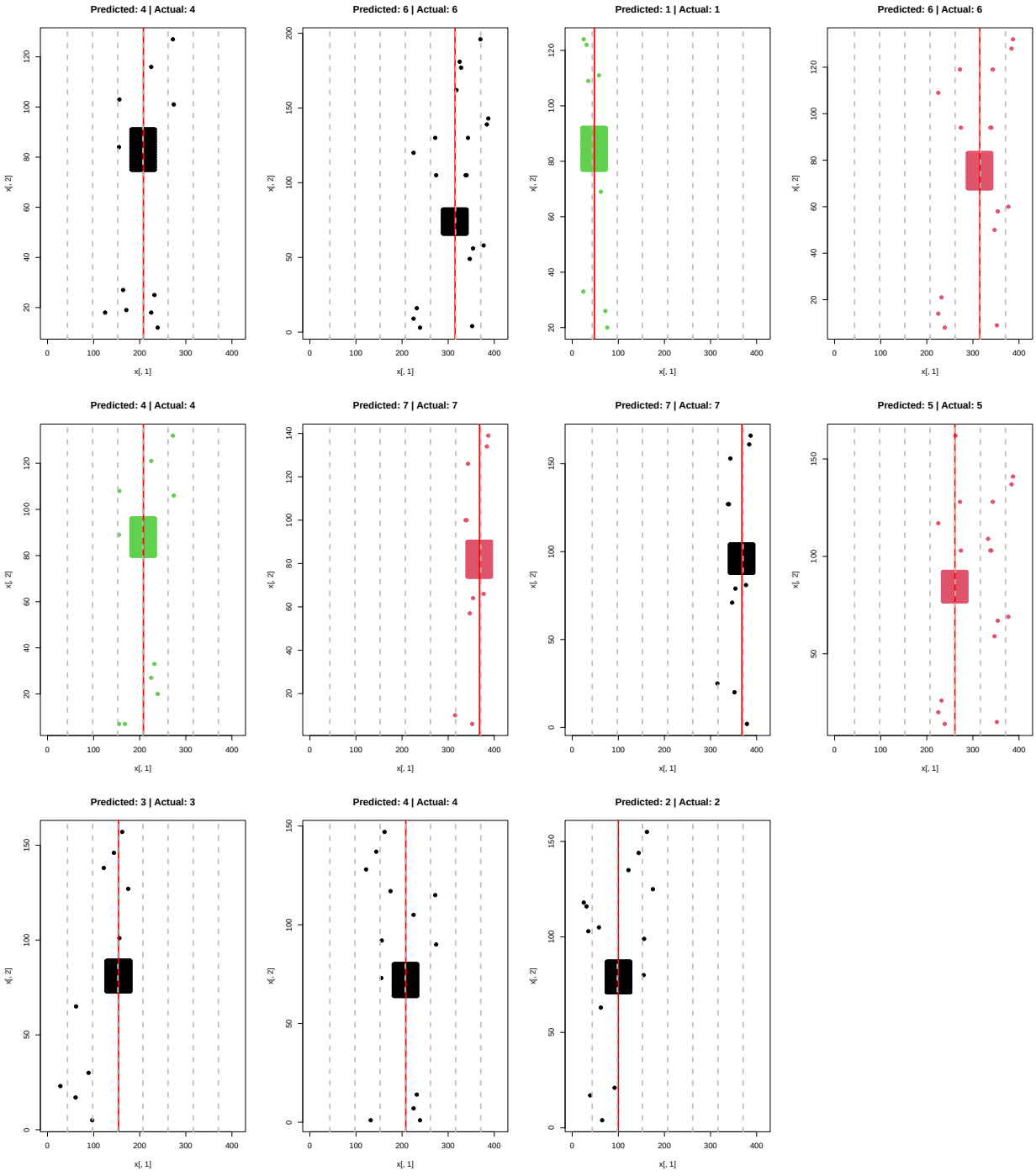
Tests

Starting with one KAG.

Erste

```
# 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)

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

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


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

```r
# 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, 413))
  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_fast(doc = z, col = y), silent = F)

    })

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

, type = "message")
```

```
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
## Error in SRRI_ext_fast(doc = z, col = y) :
##   Error: No pixels of given color detected.
```

```
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
## Error in if (is.na(rel.pos)) stop("Error: Could not detect SRRI.") :
##   Argument hat Länge 0
```

```
##    [1] "Zusätzlich: Es gab 28 Warnungen (Anzeige mit warnings())"
##    [2] "Zusätzlich: Warnmeldungen:"
##    [3] "1: In min(coob[, 1]) :"
##    [4] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##    [5] "2: In max(coob[, 1]) :"
##    [6] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##    [7] "Zusätzlich: Warnmeldungen:"
##    [8] "1: In min(coob[, 1]) :"
##    [9] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##   [10] "2: In max(coob[, 1]) :"
##   [11] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##   [12] "Zusätzlich: Warnmeldungen:"
##   [13] "1: In min(coob[, 1]) :"
##   [14] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##   [15] "2: In max(coob[, 1]) :"
##   [16] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##   [17] "Zusätzlich: Warnmeldungen:"
##   [18] "1: In min(coob[, 1]) :"
##   [19] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##   [20] "2: In max(coob[, 1]) :"
##   [21] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##   [22] "Zusätzlich: Warnmeldungen:"
##   [23] "1: In min(coob[, 1]) :"
##   [24] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##   [25] "2: In max(coob[, 1]) :"
##   [26] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##   [27] "Zusätzlich: Warnmeldungen:"
```

```
##  [28] "1: In min(coob[, 1]) :"
##  [29] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [30] "2: In max(coob[, 1]) :"
##  [31] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [32] "Zusätzlich: Warnmeldungen:"
##  [33] "1: In min(coob[, 1]) :"
##  [34] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [35] "2: In max(coob[, 1]) :"
##  [36] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [37] "Zusätzlich: Warnmeldungen:"
##  [38] "1: In min(coob[, 1]) :"
##  [39] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [40] "2: In max(coob[, 1]) :"
##  [41] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [42] "Zusätzlich: Warnmeldungen:"
##  [43] "1: In min(coob[, 1]) :"
##  [44] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [45] "2: In max(coob[, 1]) :"
##  [46] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [47] "Zusätzlich: Warnmeldungen:"
##  [48] "1: In min(coob[, 1]) :"
##  [49] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [50] "2: In max(coob[, 1]) :"
##  [51] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [52] "Zusätzlich: Warnmeldungen:"
##  [53] "1: In min(coob[, 1]) :"
##  [54] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [55] "2: In max(coob[, 1]) :"
##  [56] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [57] "Zusätzlich: Warnmeldungen:"
##  [58] "1: In min(coob[, 1]) :"
##  [59] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [60] "2: In max(coob[, 1]) :"
##  [61] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [62] "Zusätzlich: Warnmeldungen:"
##  [63] "1: In min(coob[, 1]) :"
##  [64] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [65] "2: In max(coob[, 1]) :"
##  [66] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [67] "Zusätzlich: Warnmeldungen:"
##  [68] "1: In min(coob[, 1]) :"
##  [69] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [70] "2: In max(coob[, 1]) :"
##  [71] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [72] "Zusätzlich: Warnmeldungen:"
##  [73] "1: In min(coob[, 1]) :"
##  [74] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [75] "2: In max(coob[, 1]) :"
##  [76] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [77] "Zusätzlich: Warnmeldungen:"
##  [78] "1: In min(coob[, 1]) :"
##  [79] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [80] "2: In max(coob[, 1]) :"
##  [81] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
```

```
##  [82] "Zusätzlich: Warnmeldungen:"
##  [83] "1: In min(coob[, 1]) :"
##  [84] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [85] "2: In max(coob[, 1]) :"
##  [86] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [87] "Zusätzlich: Warnmeldungen:"
##  [88] "1: In min(coob[, 1]) :"
##  [89] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [90] "2: In max(coob[, 1]) :"
##  [91] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [92] "Zusätzlich: Warnmeldungen:"
##  [93] "1: In min(coob[, 1]) :"
##  [94] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
##  [95] "2: In max(coob[, 1]) :"
##  [96] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
##  [97] "Zusätzlich: Warnmeldungen:"
##  [98] "1: In min(coob[, 1]) :"
##  [99] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
## [100] "2: In max(coob[, 1]) :"
## [101] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
## [102] "Zusätzlich: Warnmeldungen:"
## [103] "1: In min(coob[, 1]) :"
## [104] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
## [105] "2: In max(coob[, 1]) :"
## [106] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
## [107] "Zusätzlich: Warnmeldungen:"
## [108] "1: In min(coob[, 1]) :"
## [109] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
## [110] "2: In max(coob[, 1]) :"
## [111] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
## [112] "Zusätzlich: Warnmeldungen:"
## [113] "1: In min(coob[, 1]) :"
## [114] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
## [115] "2: In max(coob[, 1]) :"
## [116] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
## [117] "Zusätzlich: Warnmeldungen:"
## [118] "1: In min(coob[, 1]) :"
## [119] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
## [120] "2: In max(coob[, 1]) :"
## [121] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
## [122] "Zusätzlich: Warnmeldungen:"
## [123] "1: In min(coob[, 1]) :"
## [124] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
## [125] "2: In max(coob[, 1]) :"
## [126] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
## [127] "Zusätzlich: Warnmeldungen:"
## [128] "1: In min(coob[, 1]) :"
## [129] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
## [130] "2: In max(coob[, 1]) :"
## [131] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
## [132] "Zusätzlich: Warnmeldungen:"
## [133] "1: In min(coob[, 1]) :"
## [134] "  kein nicht-fehlendes Argument für min; gebe Inf zurück"
## [135] "2: In max(coob[, 1]) :"
```

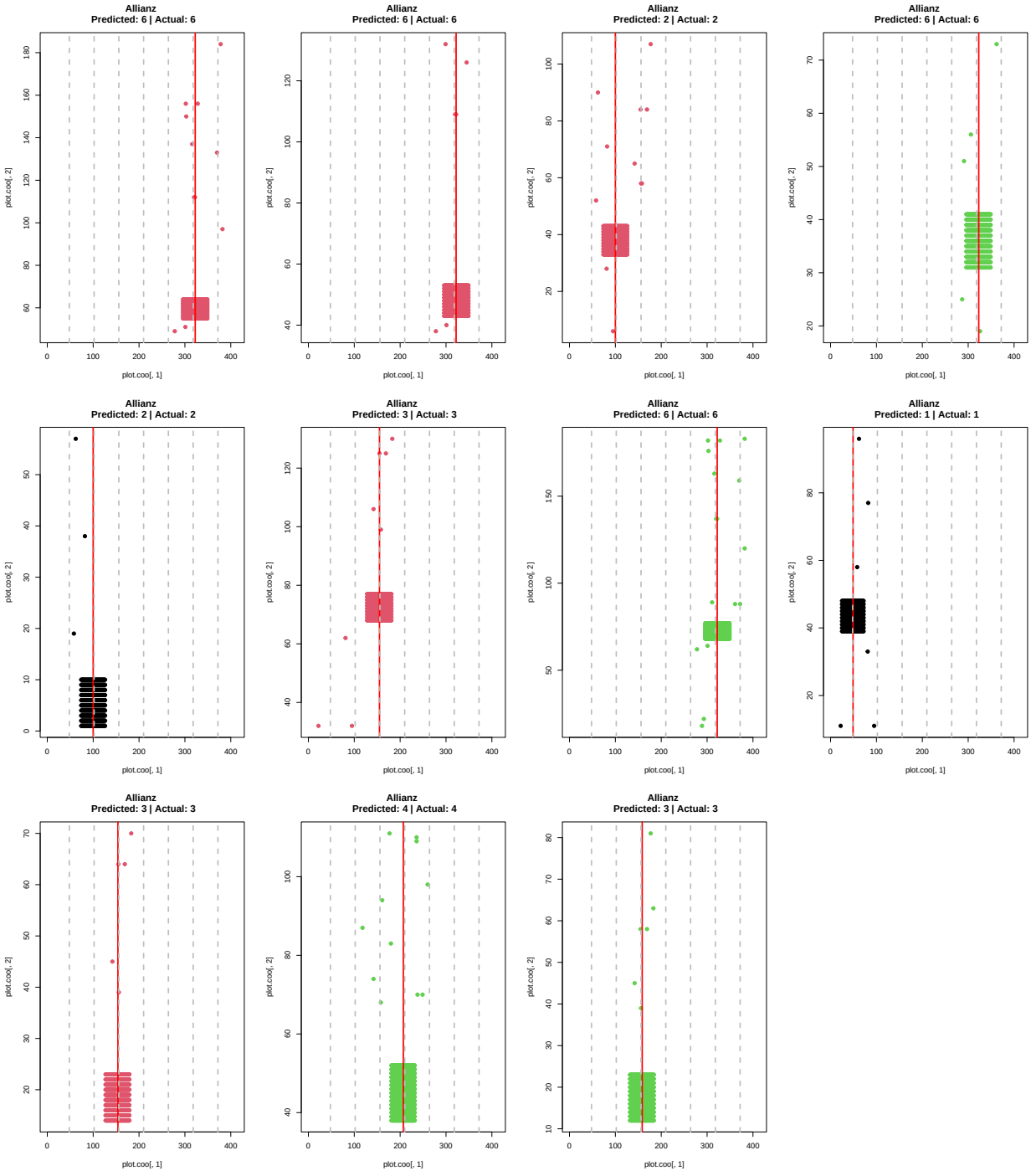
```
## [136] "  kein nicht-fehlendes Argument für max; gebe -Inf zurück"
```

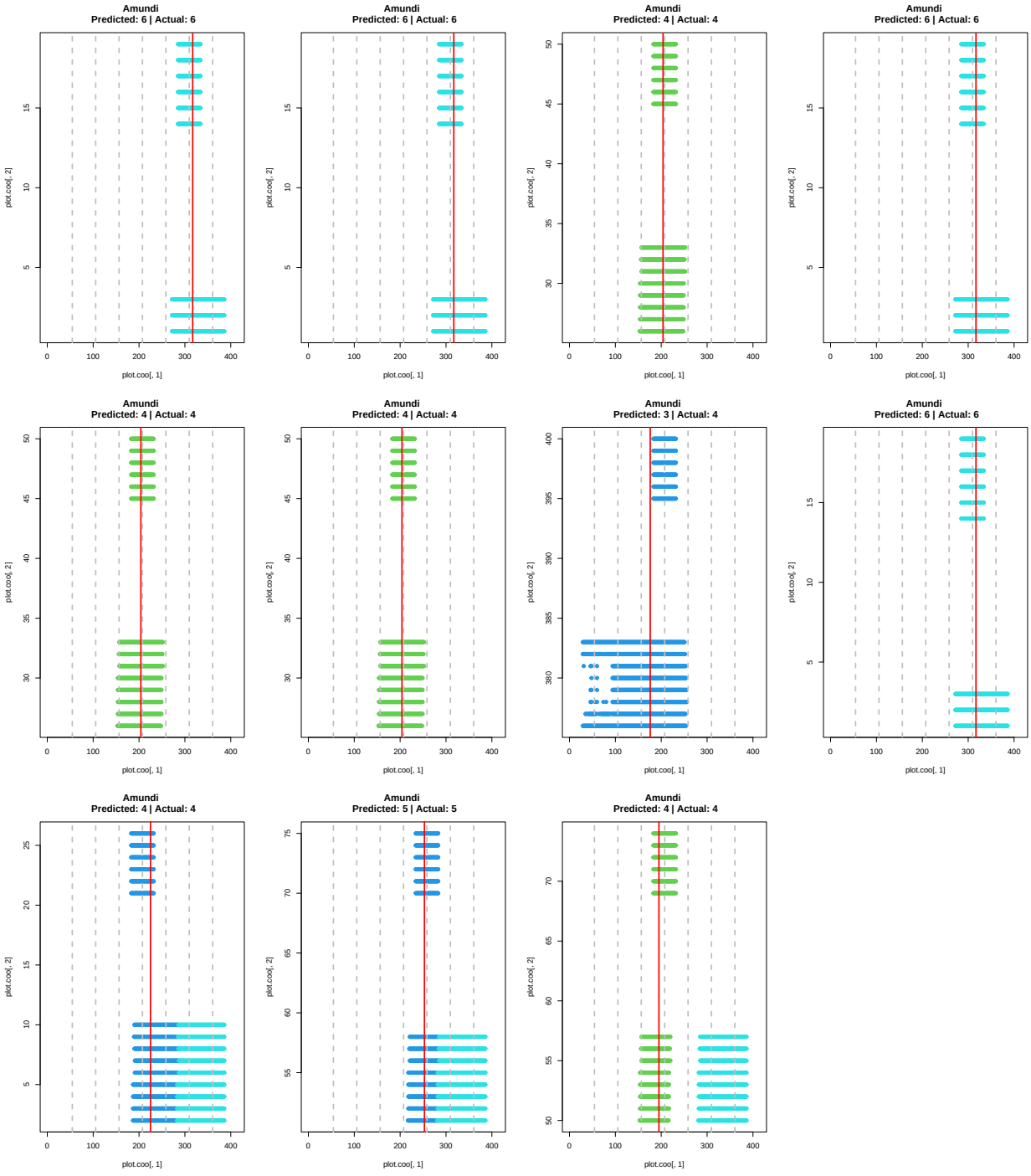
```
# error index
lapply(test, function(x){

  # 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
```

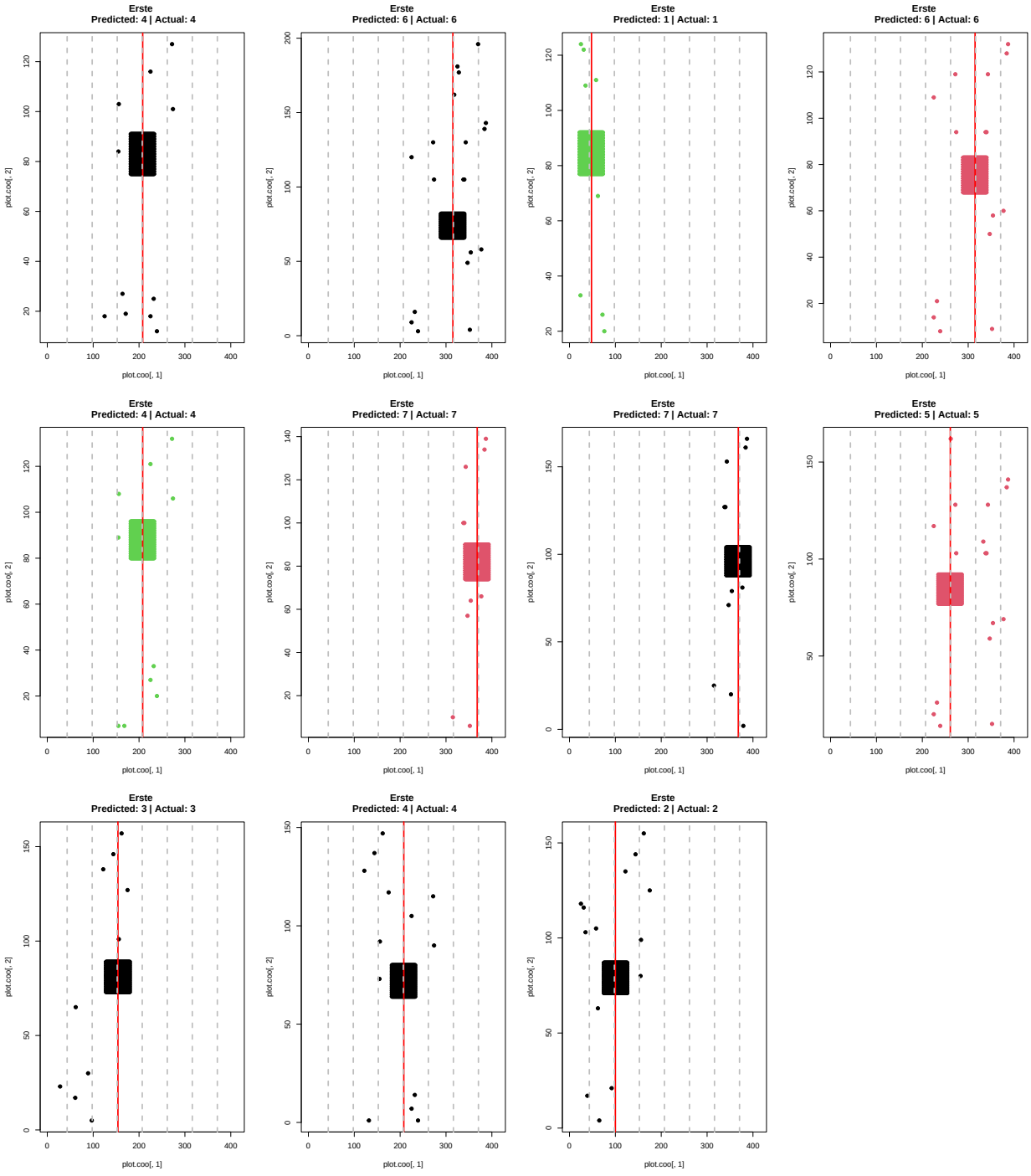
Now that we have identified all KIDs for which the extraction failed, we can proceed to see if the classification was correct for the remaining kids.

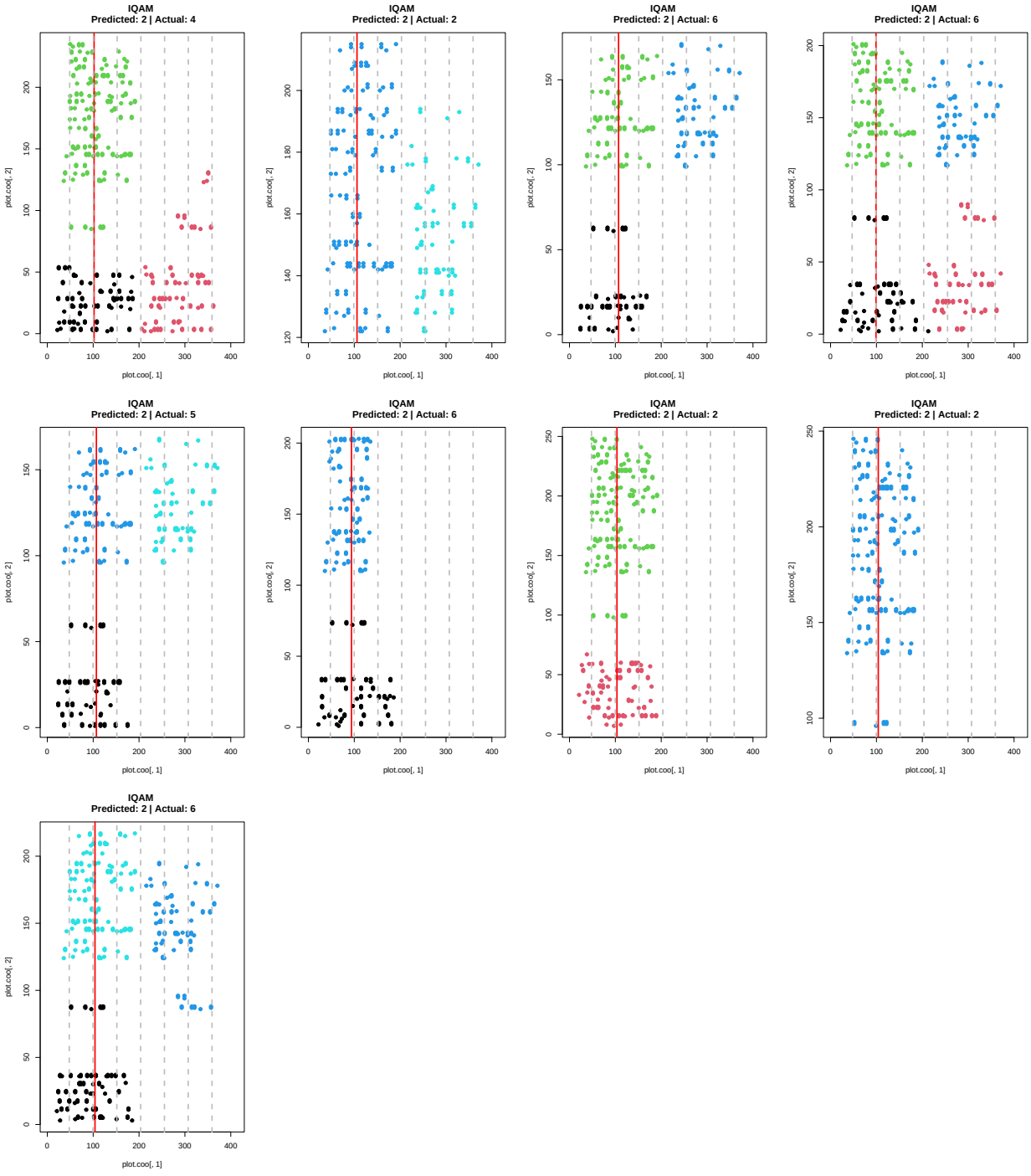
```
# Plot
Map(function(x, y){

  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
```

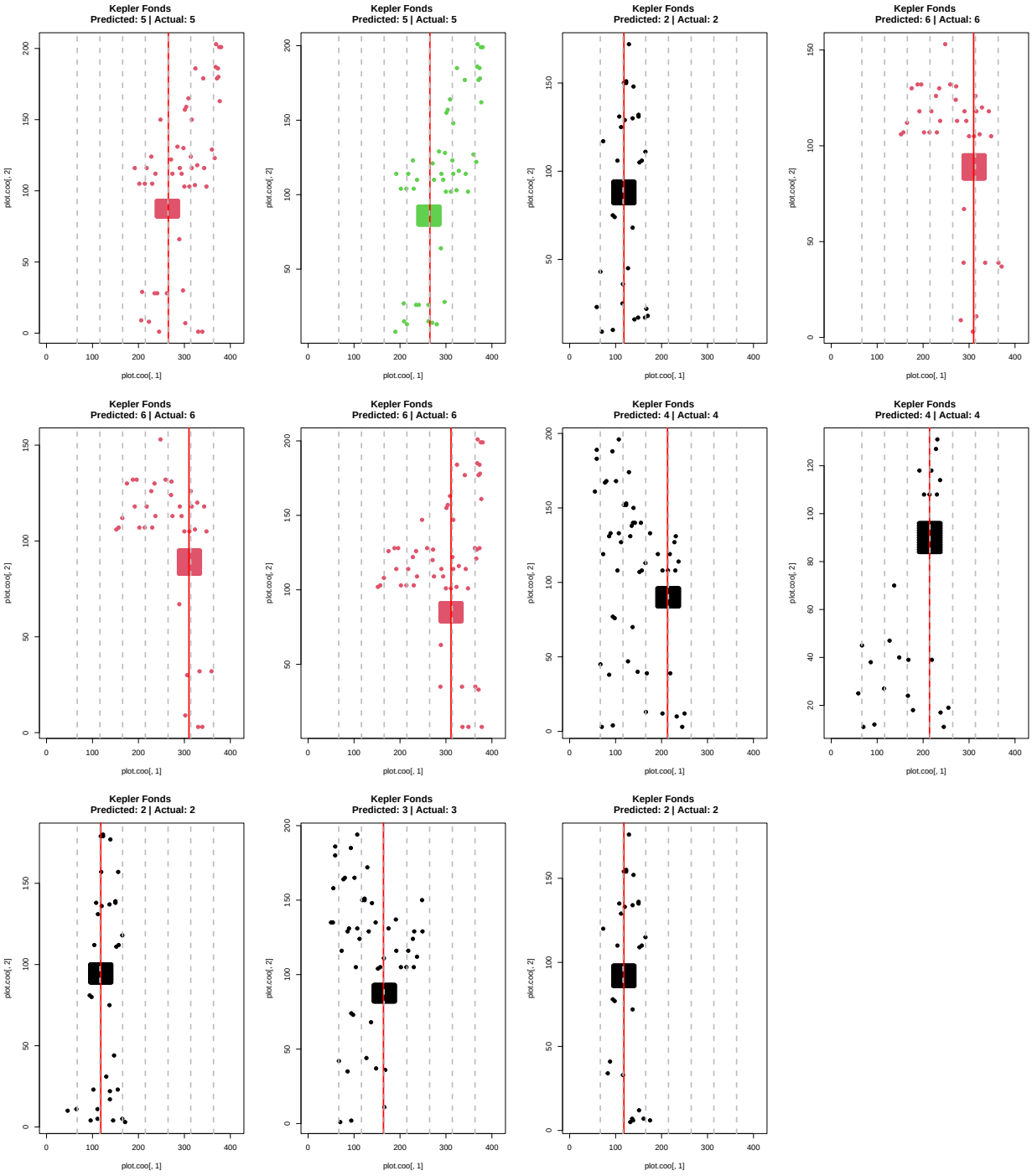
```r
# over KAGs
invis.Map(function(m, n){

  # 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, 413),
           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)
```

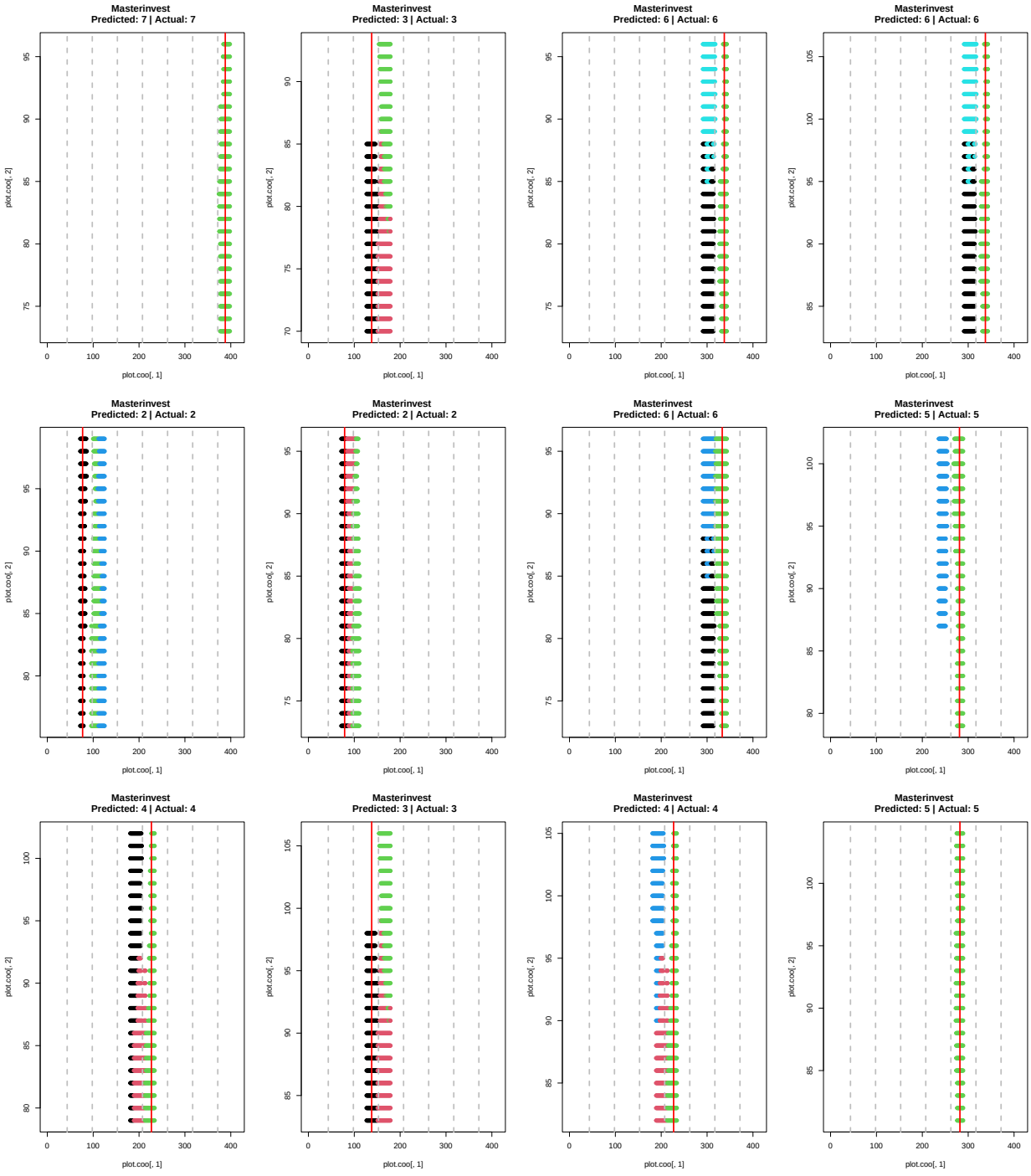

Allianz
Predicted: 6 | Actual: 6
Allianz
Predicted: 6 | Actual: 6
Allianz
Predicted: 2 | Actual: 2
Allianz
Predicted: 6 | Actual: 6
Allianz
Predicted: 2 | Actual: 2
Allianz
Predicted: 3 | Actual: 3
Allianz
Predicted: 6 | Actual: 6
Allianz
Predicted: 1 | Actual: 1
Allianz
Predicted: 3 | Actual: 3
Allianz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

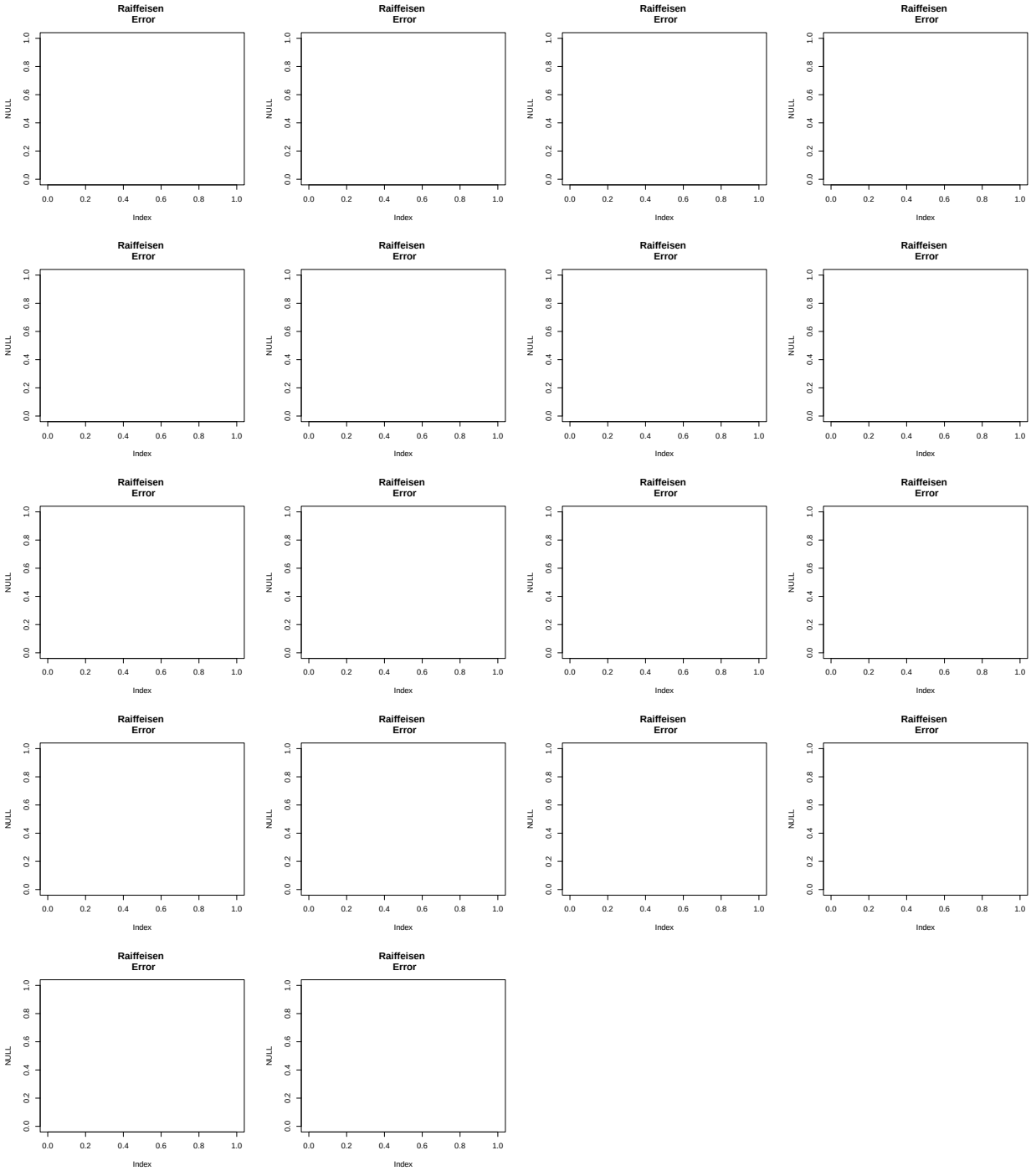

Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 4 | Actual: 4
Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 4 | Actual: 4
Amundi
Predicted: 4 | Actual: 4
Amundi
Predicted: 3 | Actual: 4
Amundi
Predicted: 6 | Actual: 6
Amundi
Predicted: 4 | Actual: 4
Amundi
Predicted: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

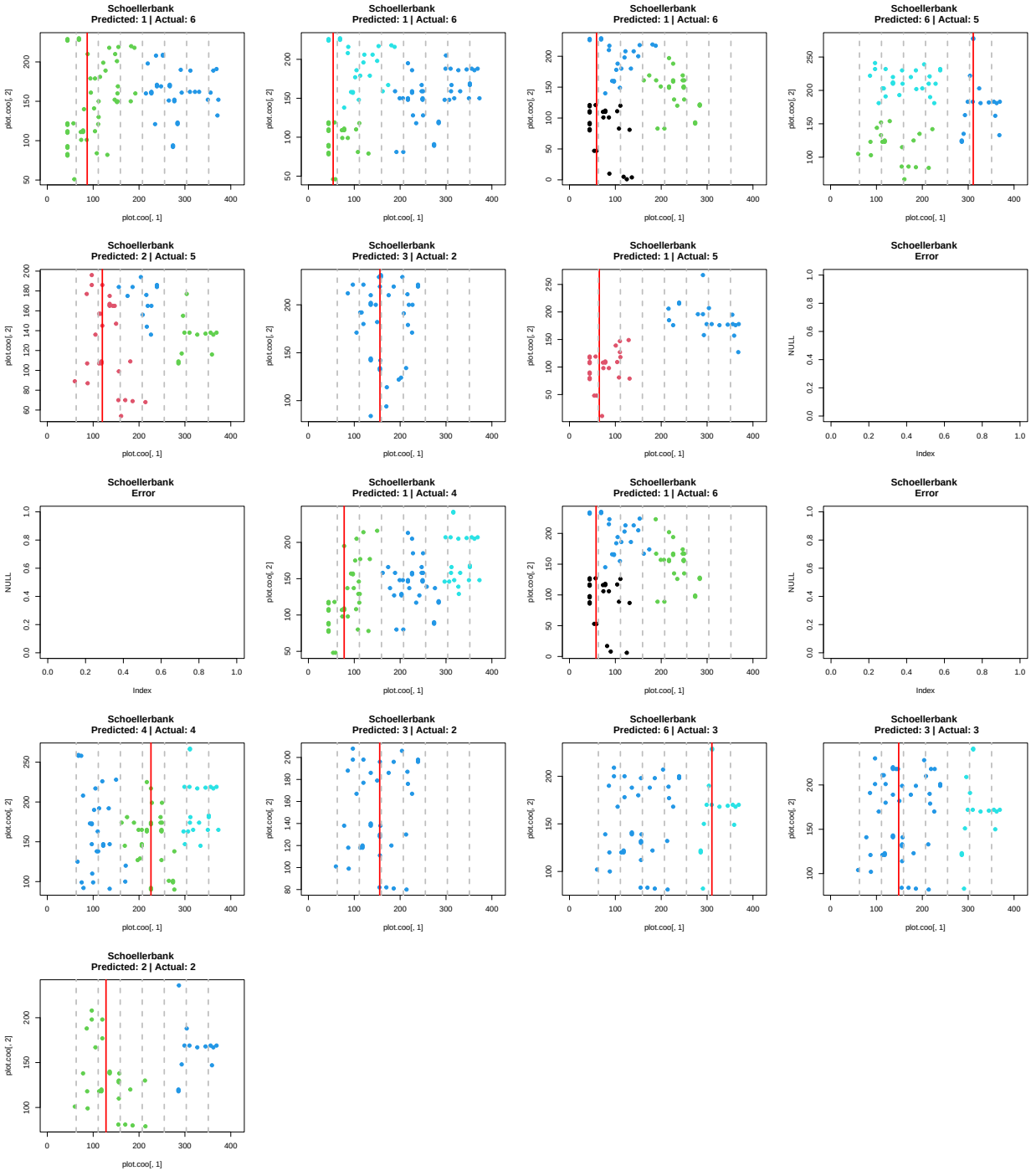

Erste
Predicted: 4 | Actual: 4
Erste
Predicted: 6 | Actual: 6
Erste
Predicted: 1 | Actual: 1
Erste
Predicted: 6 | Actual: 6
Erste
Predicted: 4 | Actual: 4
Erste
Predicted: 7 | Actual: 7
Erste
Predicted: 7 | Actual: 7
Erste
Predicted: 5 | Actual: 5
Erste
Predicted: 3 | Actual: 3
Erste
Predicted: 4 | Actual: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

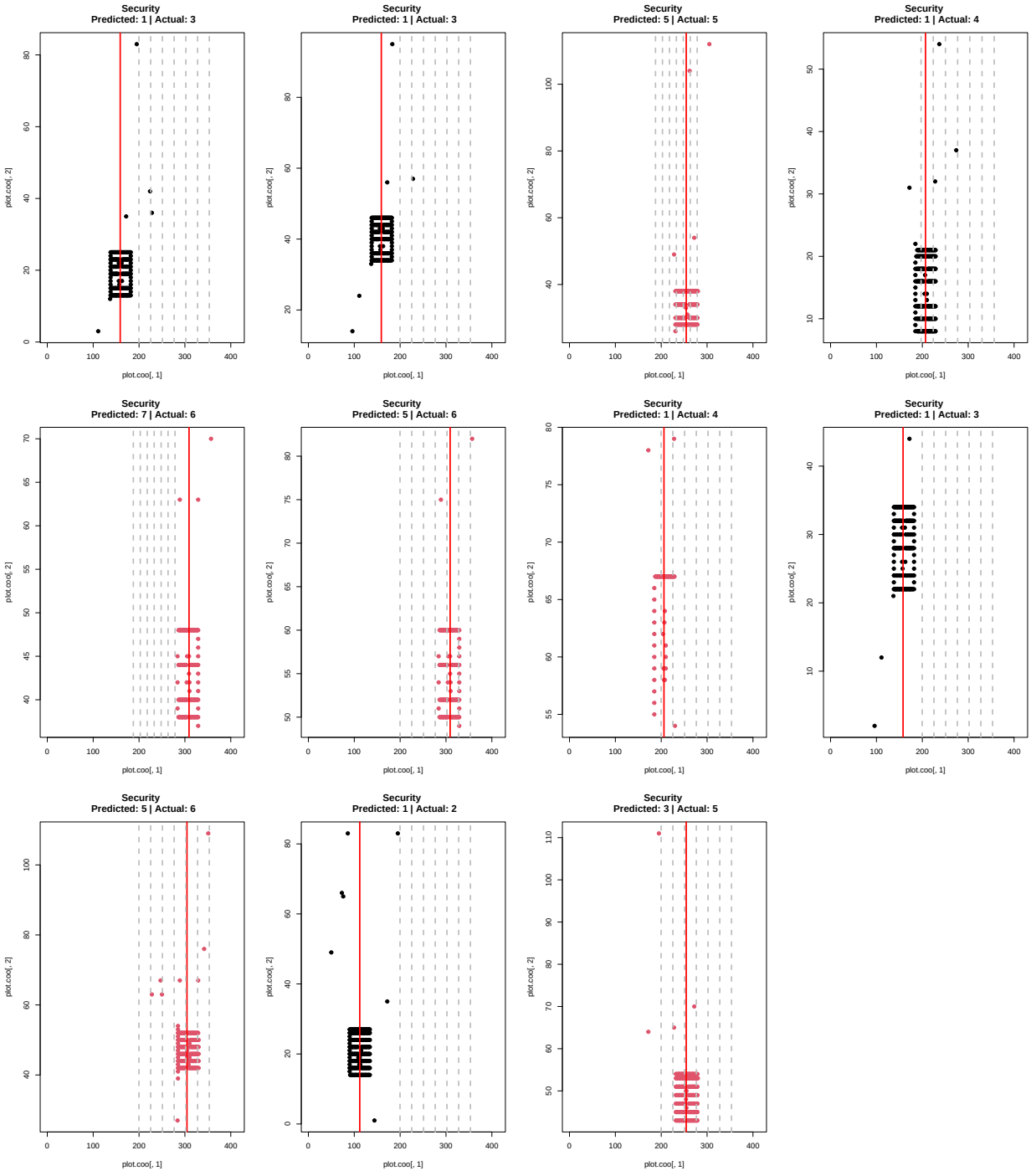

IQAM
Predicted: 2 | Actual: 4
IQAM
Predicted: 2 | Actual: 2
IQAM
Predicted: 2 | Actual: 6
IQAM
Predicted: 2 | Actual: 6
IQAM
Predicted: 2 | Actual: 5
IQAM
Predicted: 2 | Actual: 6
IQAM
Predicted: 2 | Actual: 2
IQAM
Predicted: 2 | Actual: 2
IQAM
Predicted: 2 | Actual: 6
plot.coo[, 1]
plot.coo[, 2]

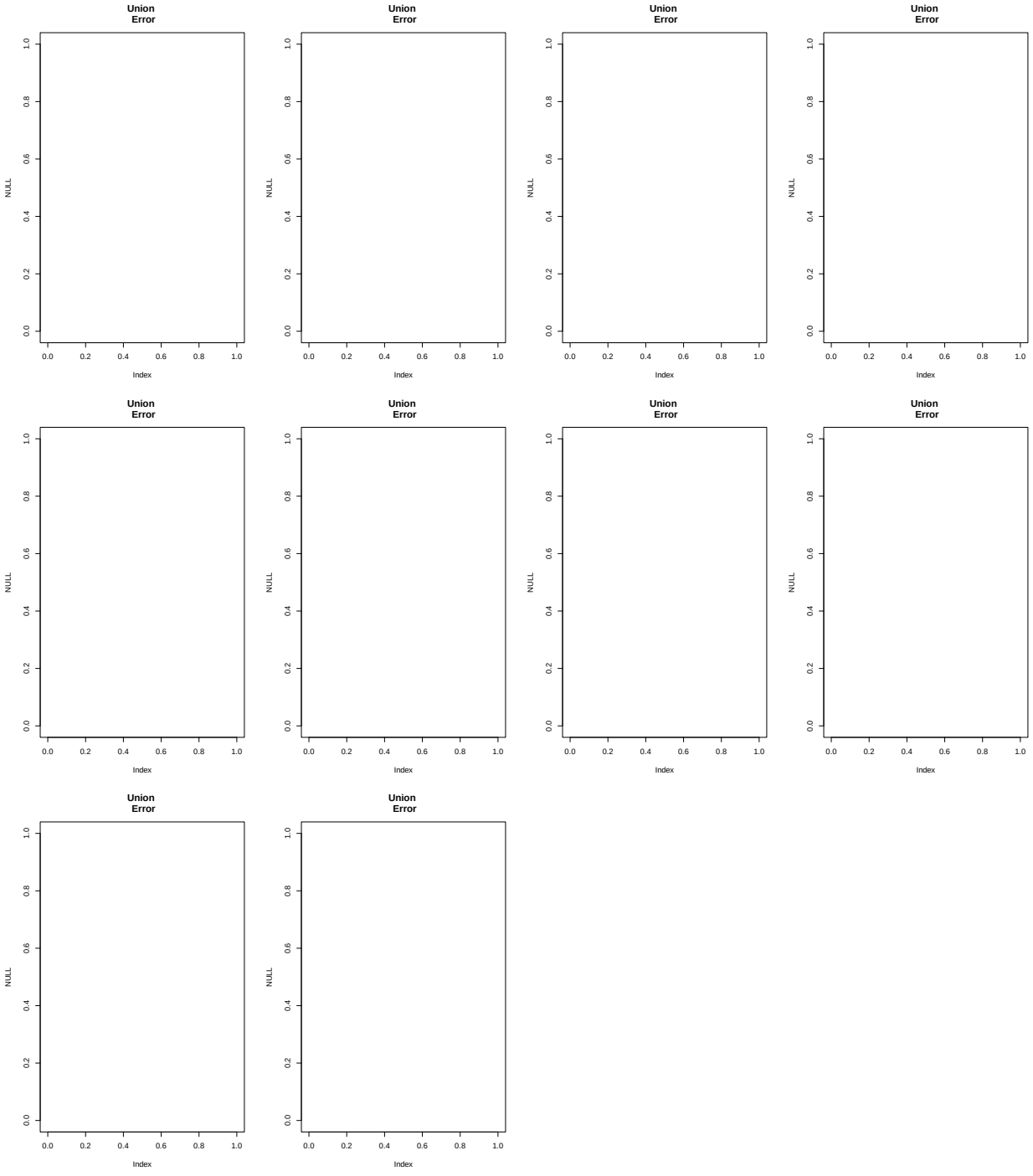

Kepler Fonds
Predicted: 5 | Actual: 5
Kepler Fonds
Predicted: 5 | Actual: 5
Kepler Fonds
Predicted: 2 | Actual: 2
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 6 | Actual: 6
Kepler Fonds
Predicted: 4 | Actual: 4
Kepler Fonds
Predicted: 4 | Actual: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


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



Problem description 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

Diagnosis

Alianz

No errors were thrown by the shade.ext(), however, in three cases the plot does not display the pixel cluster containing the SRRI. As the plots are created in order as stored in the dir we can indicidually diagnose the problem for those three.

```
# Test function
# source("C:/Users/blasc/OneDrive/Documents/GitHub/KID/Code/Functions/SRRI_ext_test.R")

# problematic .pdfs. 5, 9 and 11
# starting with the 5th

# {setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs/Allianz")
# trb.pdf.AL <- list.files(pattern = ".pdf")[c(5, 9, 11)] }
#
# debug.AL <- lapply(trb.pdf.AL, function(x){
#
#   setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs/Allianz")
#   SRRI_ext_test(x, "#A6A6A6", off = 0.1)
#
# })
```

```
## pot

# debug.Al1 <- debug.AL[[1]]
# plot(debug.Al1[, 1], debug.Al1[, 2])
#
# tmp.plot <- which(debug.Al1[1,,] == "a6" & debug.Al1[2,,] == "a6" &
#                     debug.Al1[3,,] == "a6", arr.ind = T)
# plot(tmp.plot[, 1], tmp.plot[, 2])
```