# 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.

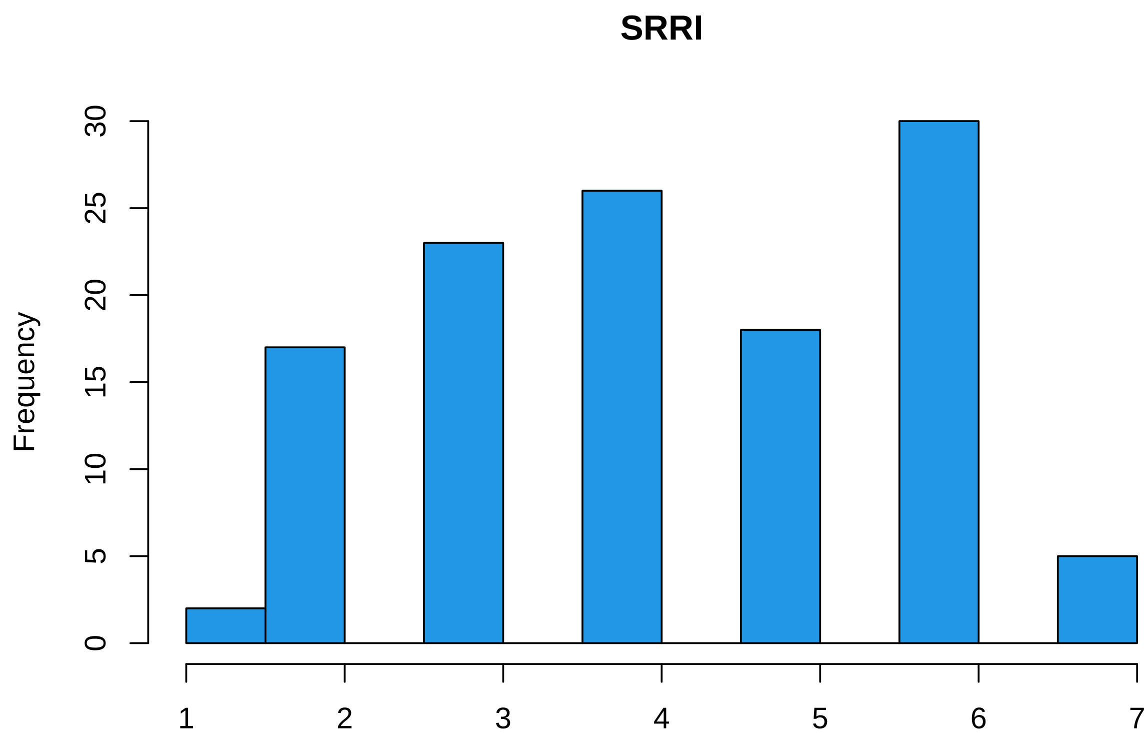
```
# set
setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs")

# 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/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
```

## 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/blasc/OneDrive/Documents/GitHub/KID/Code/Functions/SRRI_ext.R")
```

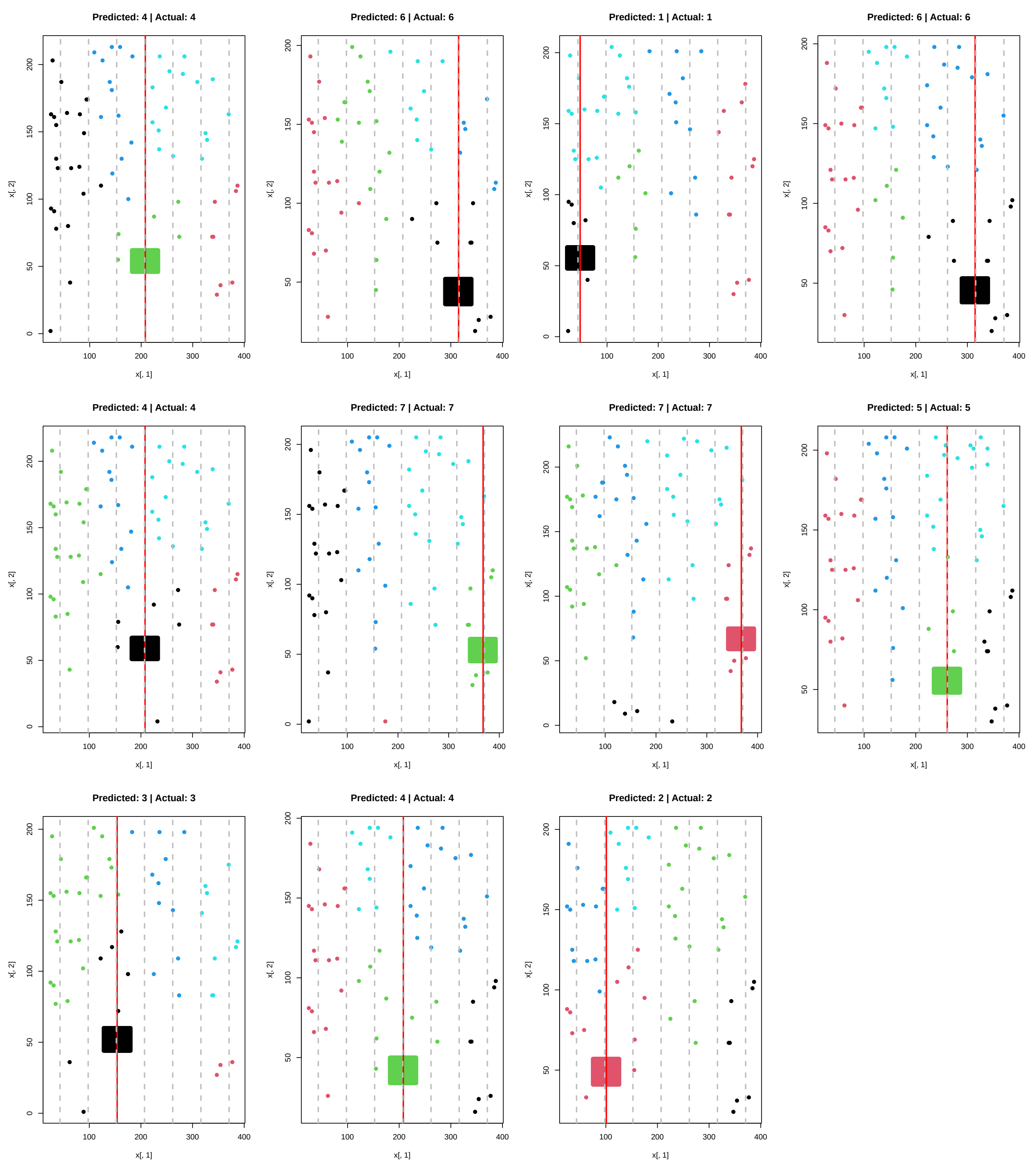
## Tests

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, y){

  # 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(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: 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 uniquely identify position of 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 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: 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: 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: 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: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
## Error in SRRI_ext(doc = z, col = y) : Error: Could not detect SRRI.
```

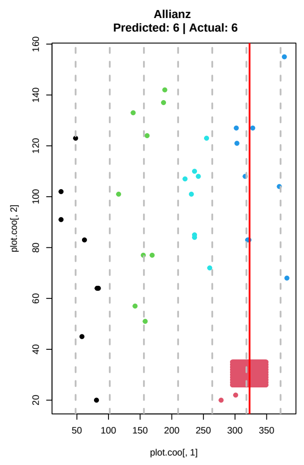
```
## [1] "Zusätzlich: Warnmeldungen:"
## [2] "1: In in_dir(input_dir(), evaluate(code, envir = env, new_device = FALSE,  :"
## [3] "  You changed the working directory to C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs (proba
## [4] "2: In in_dir(input_dir(), evaluate(code, envir = env, new_device = 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/KIDs/Erste
```

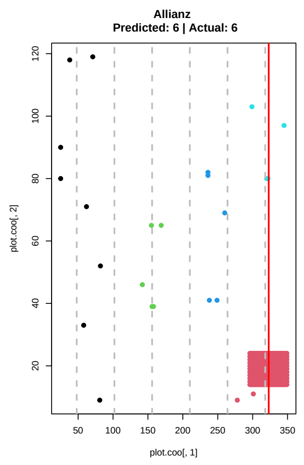
```r
# error index
lapply(test, function(x){

  # 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
```

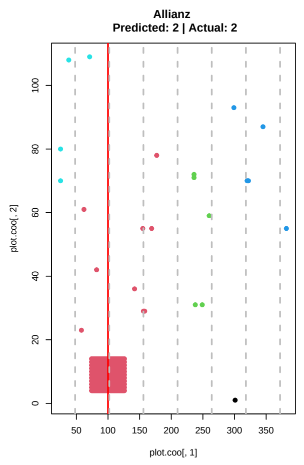
Now that we have identified all KIDs for which the extraction failed, we can proceed to see if the classification was correct for the remaining kids.

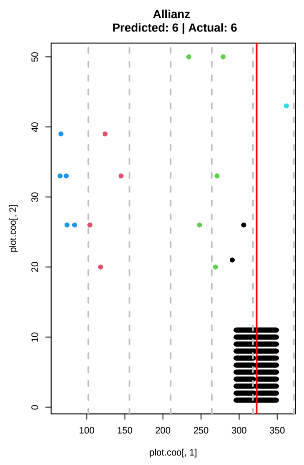
```r
# Plot
Map(function(x, y){

  sapply(y, function(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
```

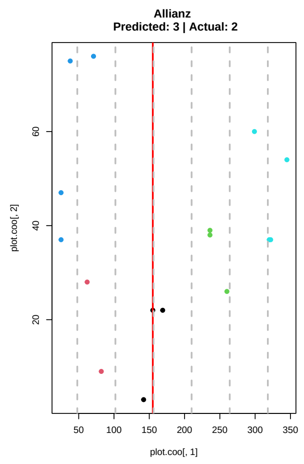
```r
par(mfrow = c(3, 4))

# 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 = "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)
```

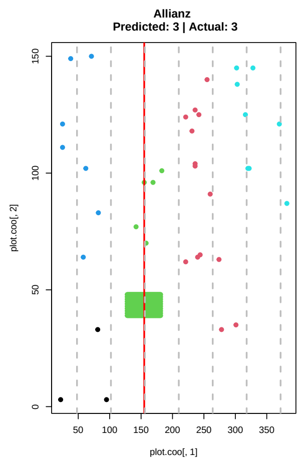

Allianz
Predicted: 6 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

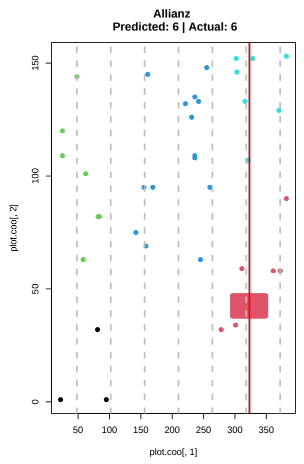

Allianz
Predicted: 6 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

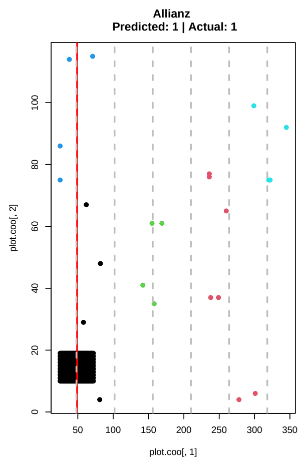

Allianz
Predicted: 2 | Actual: 2
plot.coo[, 2]
plot.coo[, 1]

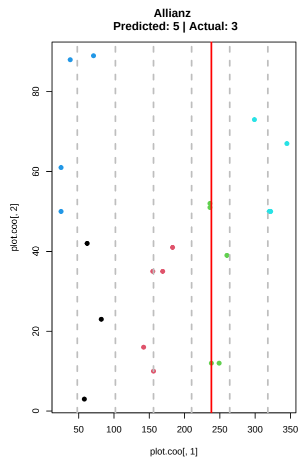

Allianz
Predicted: 6 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

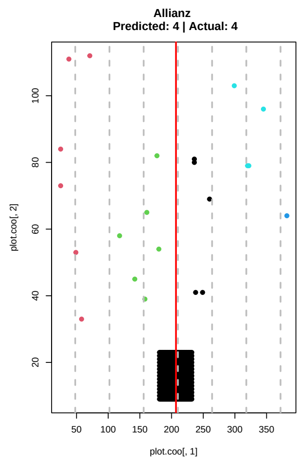

Allianz
Predicted: 3 | Actual: 2
plot.coo[, 2]
plot.coo[, 1]

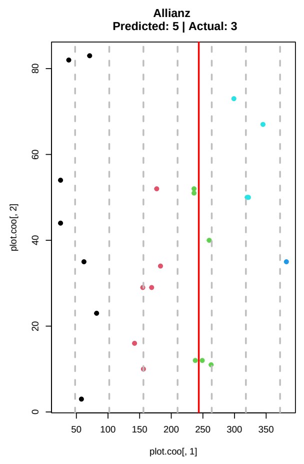

Allianz
Predicted: 3 | Actual: 3
plot.coo[, 2]
plot.coo[, 1]

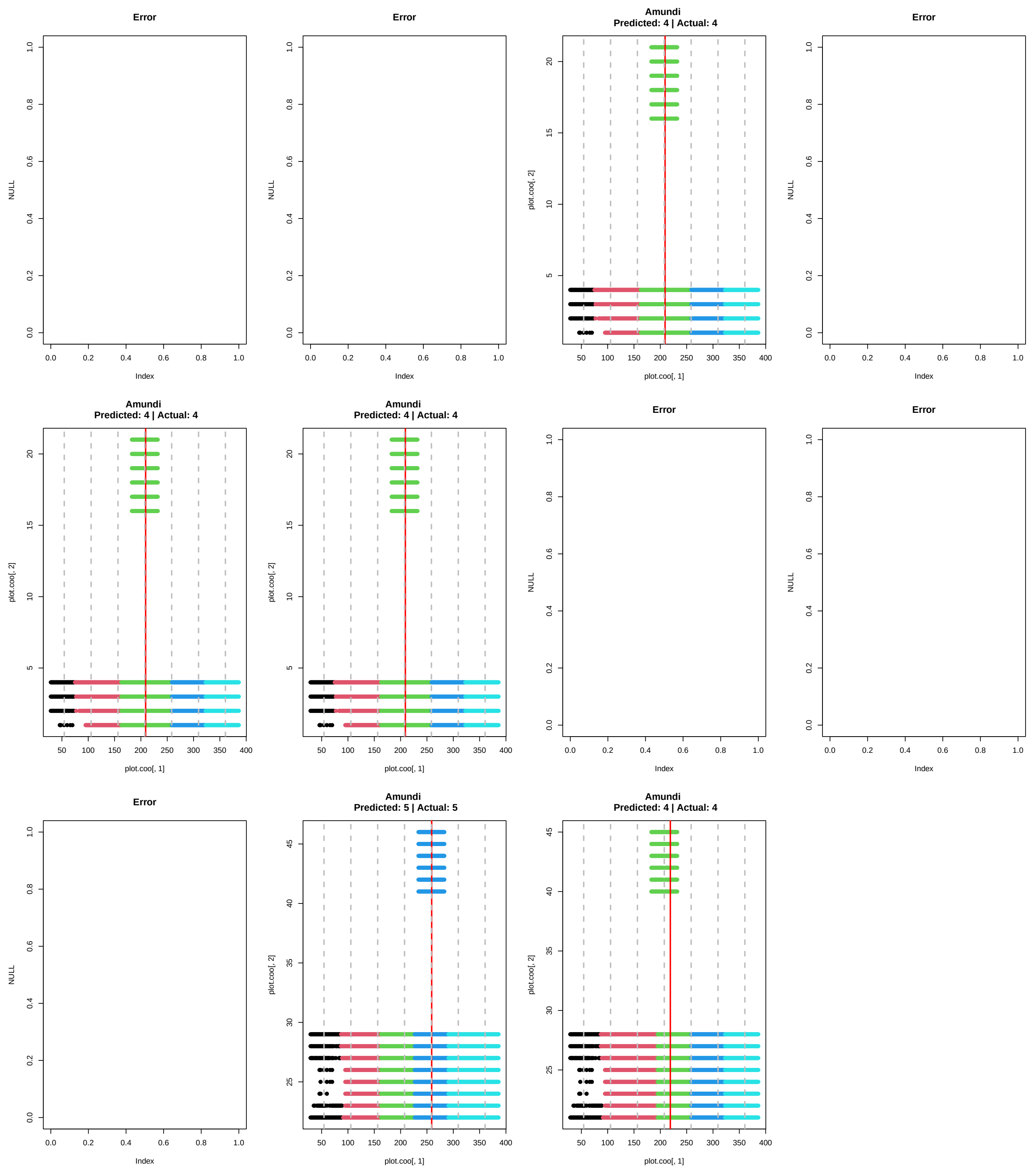

Allianz
Predicted: 6 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

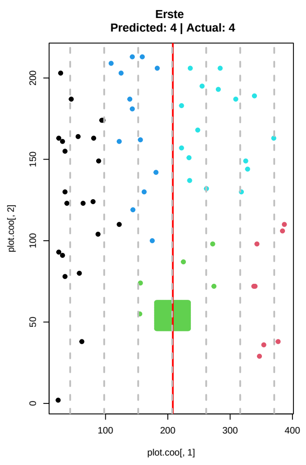

Allianz
Predicted: 1 | Actual: 1
plot.coo[, 2]
plot.coo[, 1]

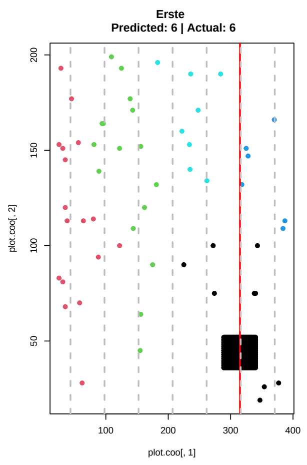

Allianz
Predicted: 5 | Actual: 3
plot.coo[, 2]
plot.coo[, 1]

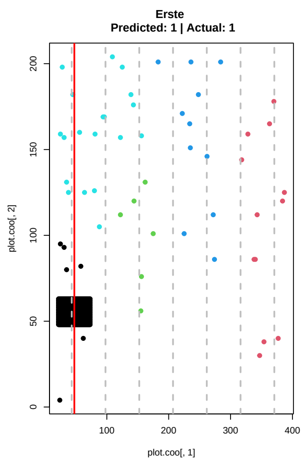

Allianz
Predicted: 4 | Actual: 4
plot.coo[, 2]
plot.coo[, 1]

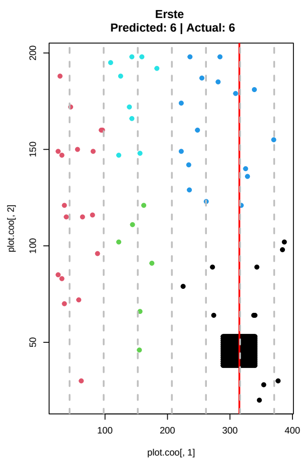

Allianz
Predicted: 5 | Actual: 3
plot.coo[, 2]
plot.coo[, 1]

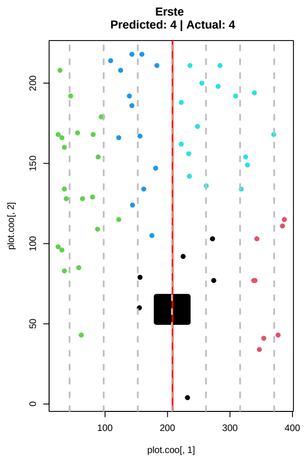

Error
Error
Amundi
Predicted: 4 | Actual: 4
Error
Amundi
Predicted: 4 | Actual: 4
Amundi
Predicted: 4 | Actual: 4
Error
Error
Error
Amundi
Predicted: 5 | Actual: 5
Amundi
Predicted: 4 | Actual: 4
NULL
Index
plot.coo[, 2]
plot.coo[, 1]

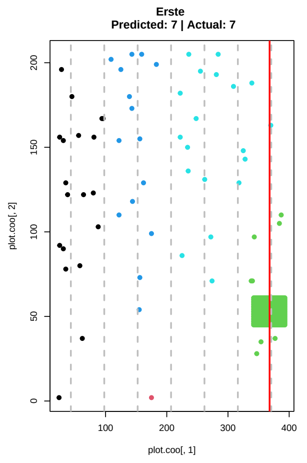

Erste
Predicted: 4 | Actual: 4
plot.coo[, 1]
plot.coo[, 2]

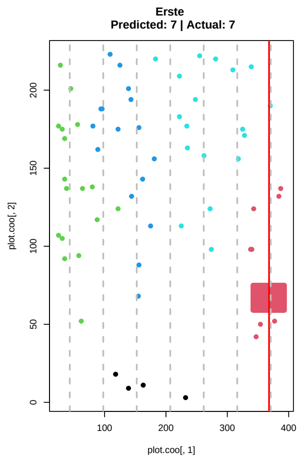

Erste
Predicted: 6 | Actual: 6
plot.coo[, 1]
plot.coo[, 2]

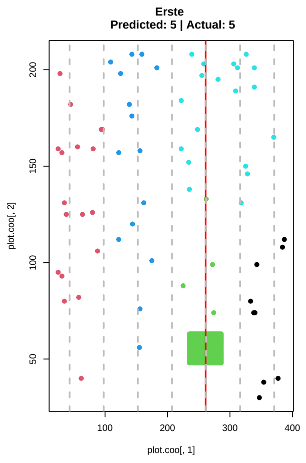

Erste
Predicted: 1 | Actual: 1
plot.coo[, 1]
plot.coo[, 2]

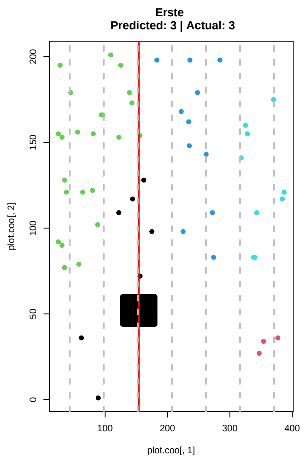

Erste
Predicted: 6 | Actual: 6
plot.coo[, 1]
plot.coo[, 2]

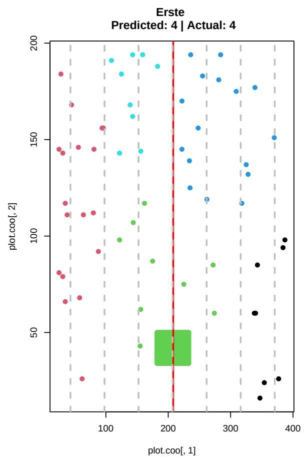

Erste
Predicted: 4 | Actual: 4
plot.coo[, 1]
plot.coo[, 2]

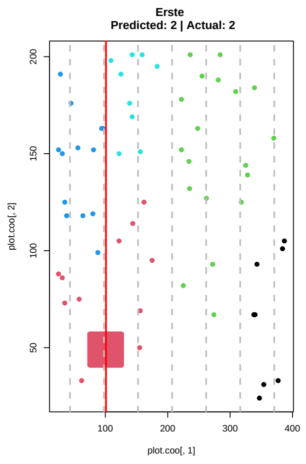

Erste
Predicted: 7 | Actual: 7
plot.coo[, 1]
plot.coo[, 2]

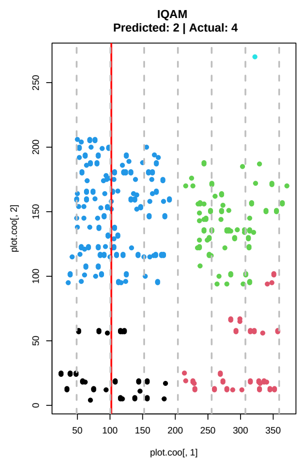

Erste
Predicted: 7 | Actual: 7
plot.coo[, 1]
plot.coo[, 2]

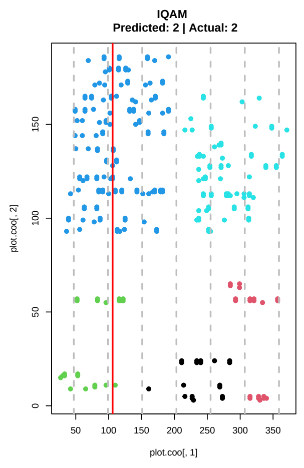

Erste
Predicted: 5 | Actual: 5
plot.coo[, 1]
plot.coo[, 2]

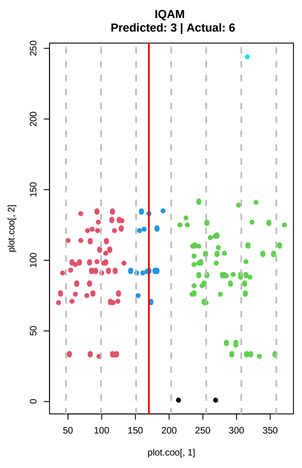

Erste
Predicted: 3 | Actual: 3
plot.coo[, 1]
plot.coo[, 2]

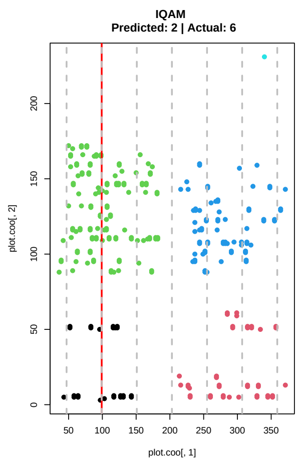

Erste
Predicted: 4 | Actual: 4
plot.coo[, 1]
plot.coo[, 2]

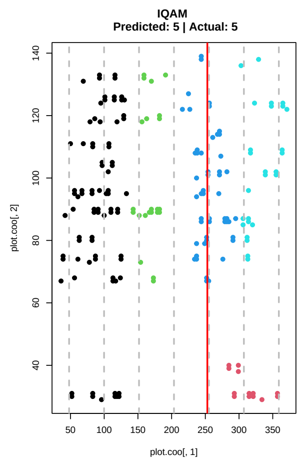

Erste
Predicted: 2 | Actual: 2
plot.coo[, 1]
plot.coo[, 2]

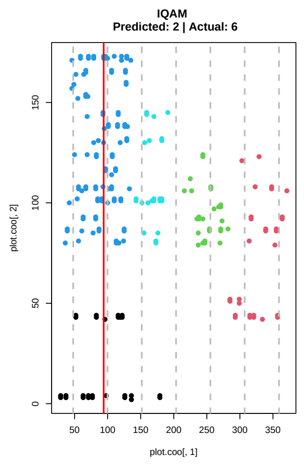

IQAM
Predicted: 2 | Actual: 4
plot.coo[, 2]
plot.coo[, 1]

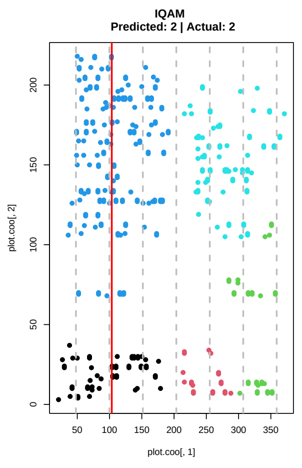

IQAM
Predicted: 2 | Actual: 2
plot.coo[, 2]
plot.coo[, 1]

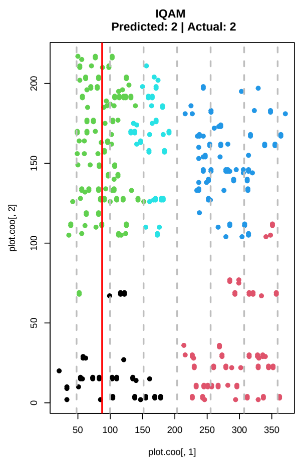

IQAM
Predicted: 3 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

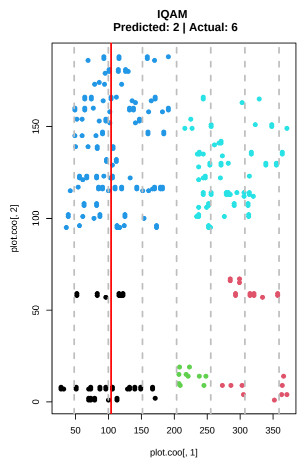

IQAM
Predicted: 2 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

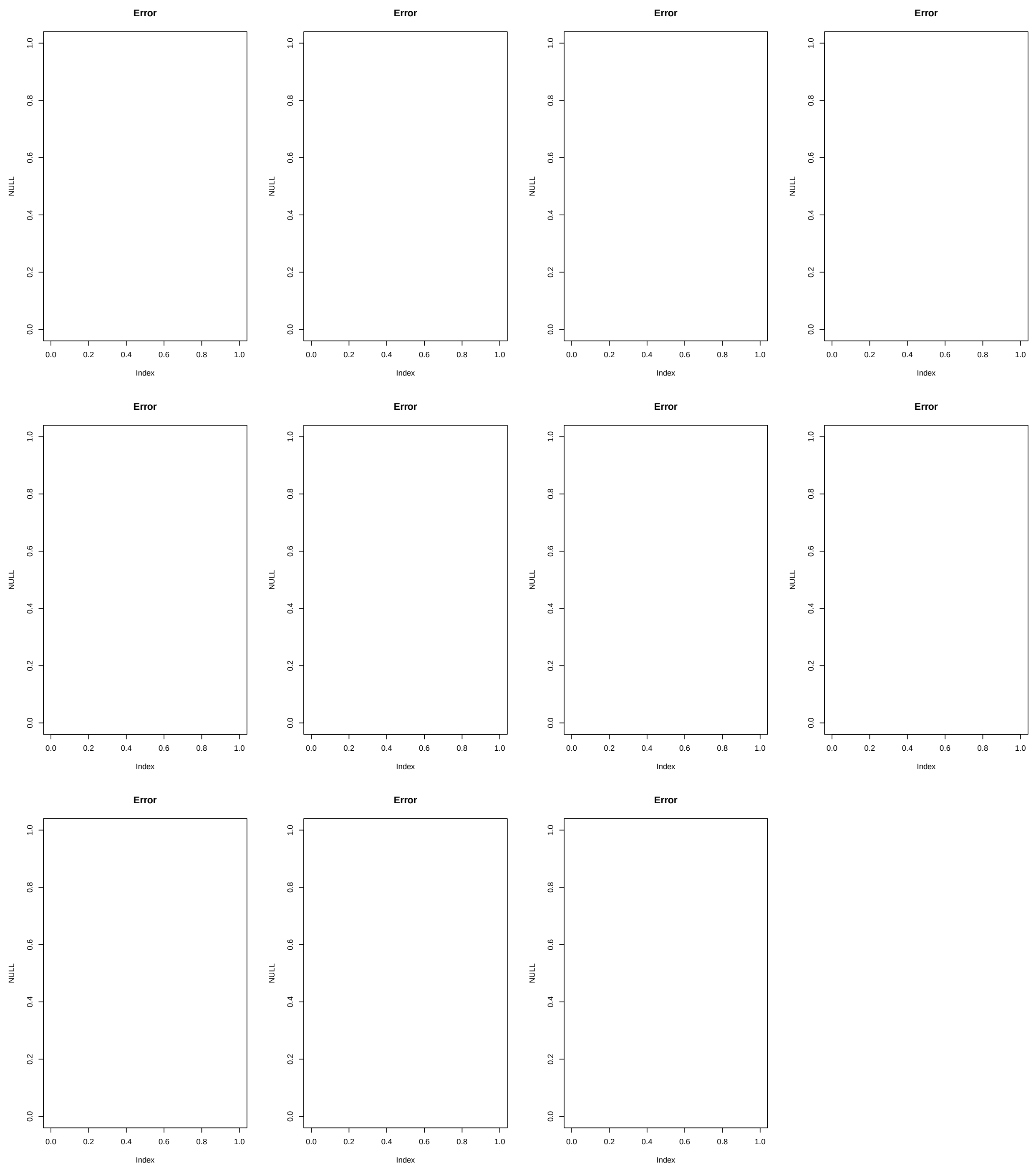

IQAM
Predicted: 5 | Actual: 5
plot.coo[, 2]
plot.coo[, 1]

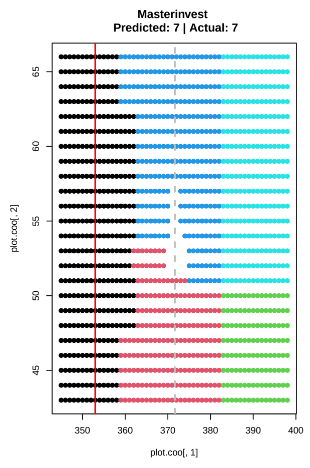

IQAM
Predicted: 2 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

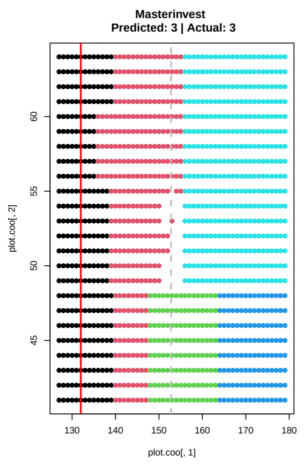

IQAM
Predicted: 2 | Actual: 2
plot.coo[, 2]
plot.coo[, 1]

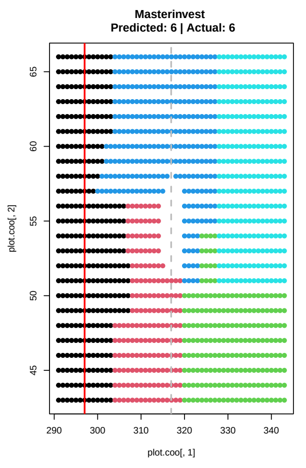

IQAM
Predicted: 2 | Actual: 2
plot.coo[, 2]
plot.coo[, 1]

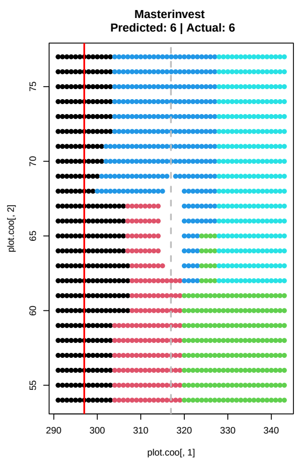

IQAM
Predicted: 2 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

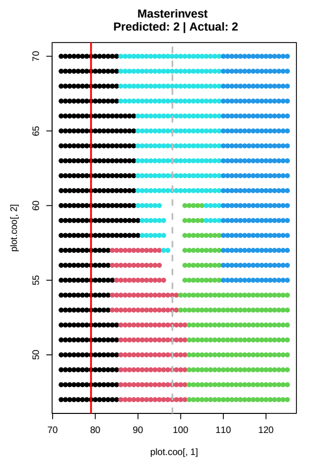

Error
NULL
Index
0.0
0.2
0.4
0.6
0.8
1.0

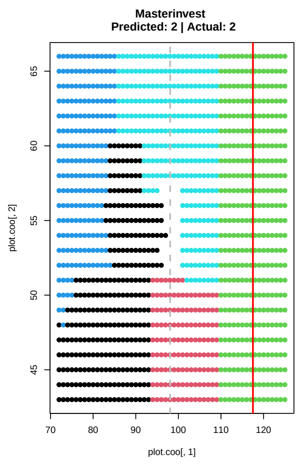

Masterinvest
Predicted: 7 | Actual: 7
plot.coo[, 1]
plot.coo[, 2]

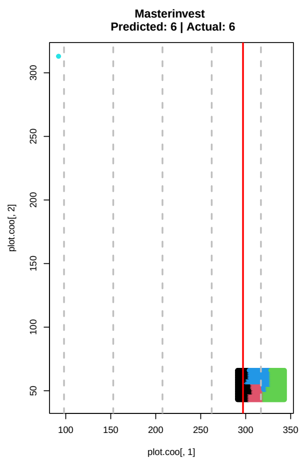

Masterinvest
Predicted: 3 | Actual: 3
plot.coo[, 1]
plot.coo[, 2]

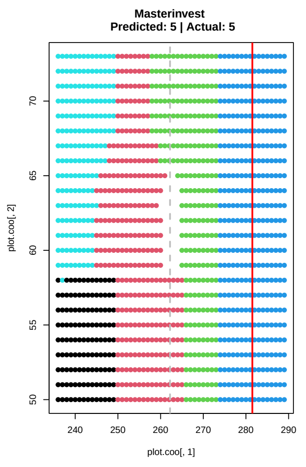

Masterinvest
Predicted: 6 | Actual: 6
plot.coo[, 1]
plot.coo[, 2]

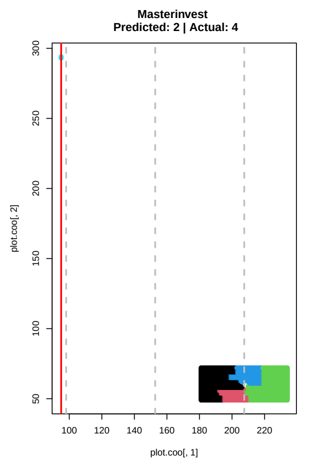

Masterinvest
Predicted: 6 | Actual: 6
plot.coo[, 1]
plot.coo[, 2]

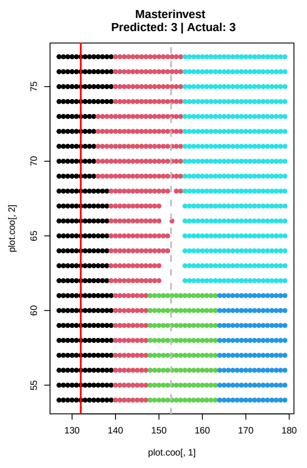

Masterinvest
Predicted: 2 | Actual: 2
plot.coo[, 1]
plot.coo[, 2]

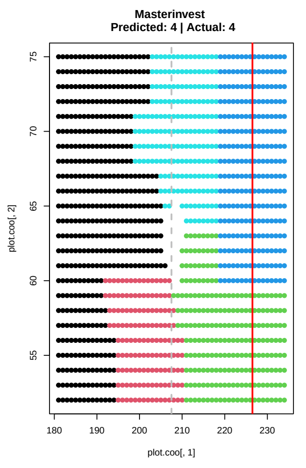

Masterinvest
Predicted: 2 | Actual: 2
plot.coo[, 1]
plot.coo[, 2]

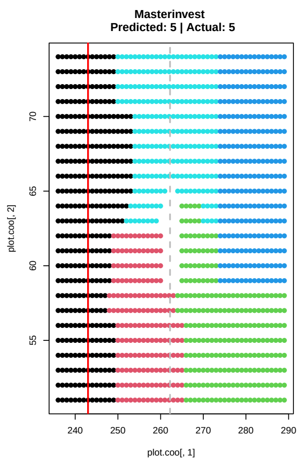

Masterinvest
Predicted: 6 | Actual: 6
plot.coo[, 1]
plot.coo[, 2]

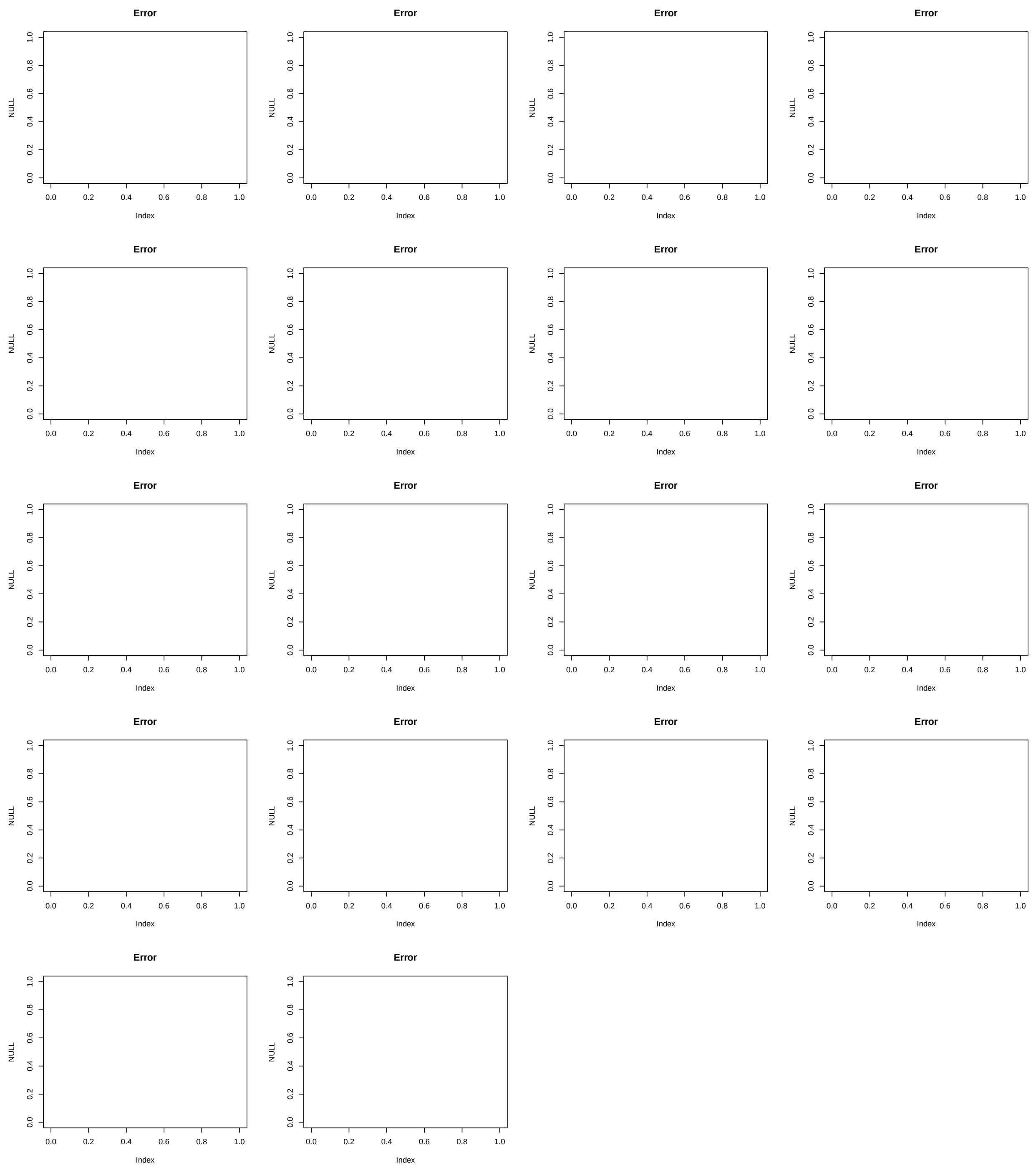

Masterinvest
Predicted: 5 | Actual: 5
plot.coo[, 1]
plot.coo[, 2]

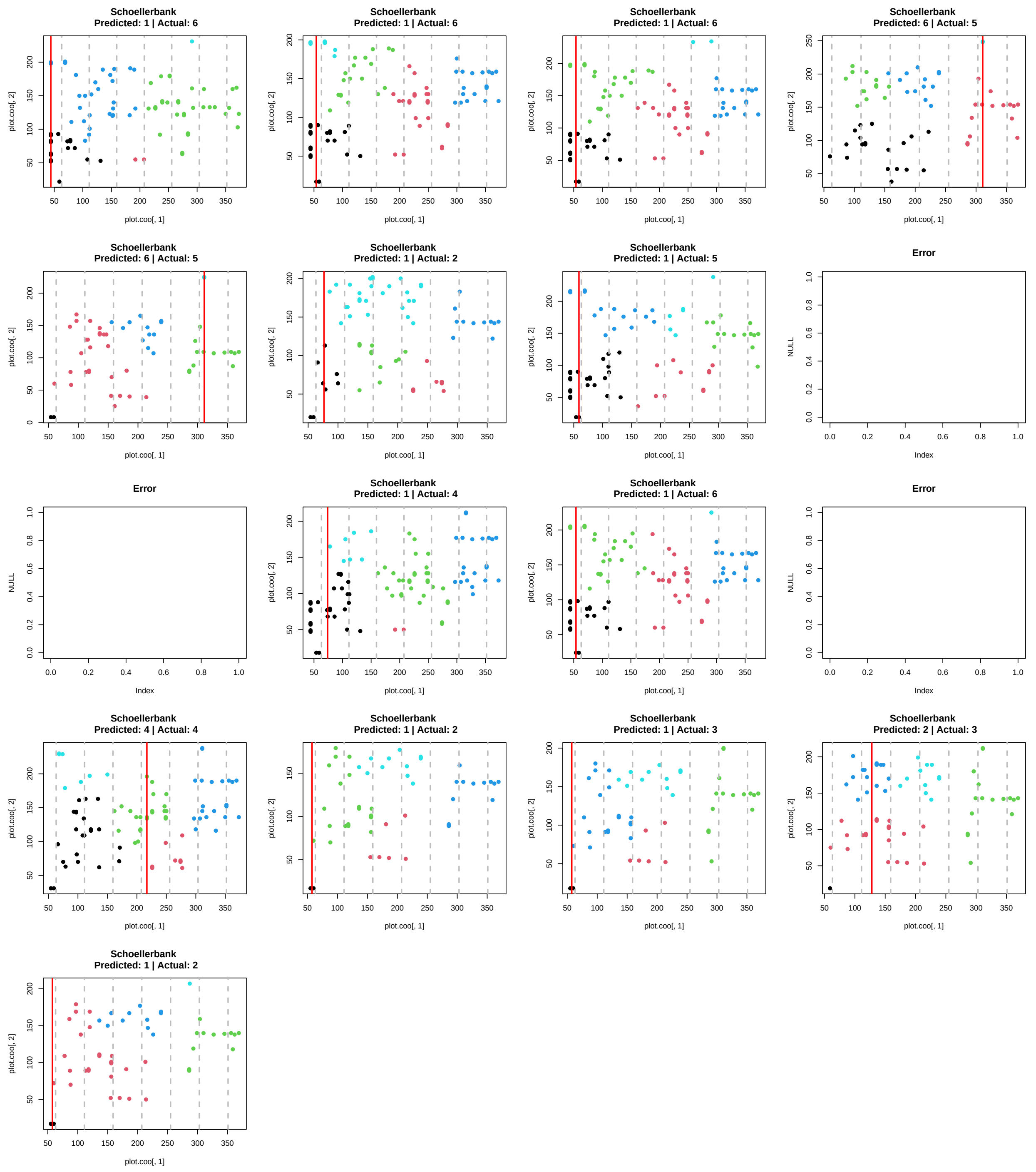

Masterinvest
Predicted: 2 | Actual: 4
plot.coo[, 1]
plot.coo[, 2]

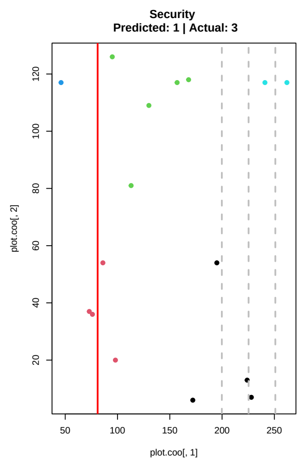

Masterinvest
Predicted: 3 | Actual: 3
plot.coo[, 1]
plot.coo[, 2]

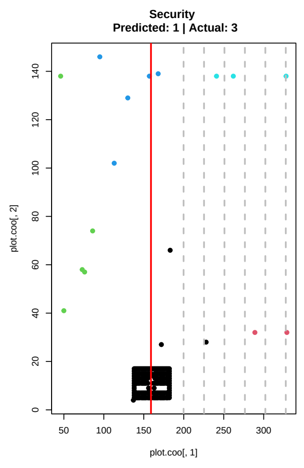

Masterinvest
Predicted: 4 | Actual: 4
plot.coo[, 1]
plot.coo[, 2]

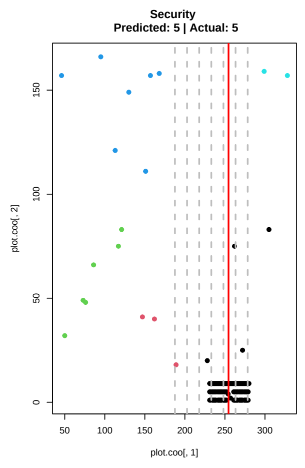

Masterinvest
Predicted: 5 | Actual: 5
plot.coo[, 1]
plot.coo[, 2]

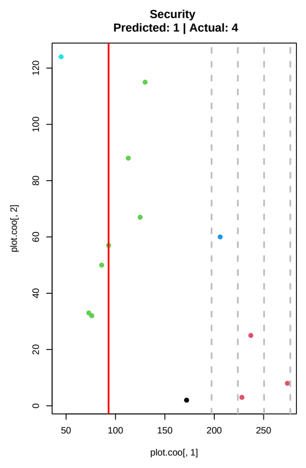

Error
NULL
Index
0.0
0.2
0.4
0.6
0.8
1.0

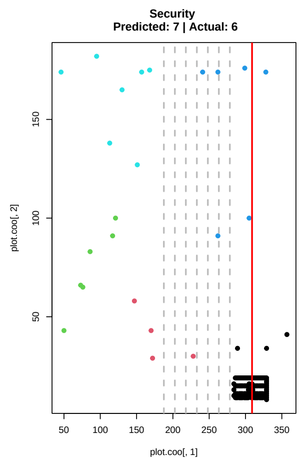

Schoellerbank
Predicted: 1 | Actual: 6
Schoellerbank
Predicted: 1 | Actual: 6
Schoellerbank
Predicted: 1 | Actual: 6
Schoellerbank
Predicted: 6 | Actual: 5
Schoellerbank
Predicted: 6 | Actual: 5
Schoellerbank
Predicted: 1 | Actual: 2
Schoellerbank
Predicted: 1 | Actual: 5
Error
Error
Schoellerbank
Predicted: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

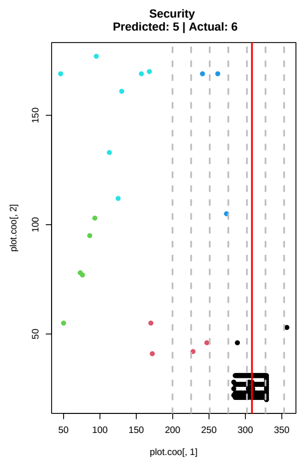

Security
Predicted: 1 | Actual: 3
plot.coo[, 2]
plot.coo[, 1]

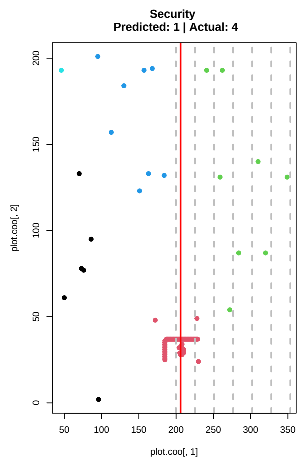

Security
Predicted: 1 | Actual: 3
plot.coo[, 2]
plot.coo[, 1]

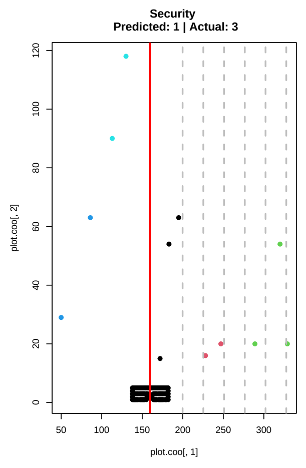

Security
Predicted: 5 | Actual: 5
plot.coo[, 2]
plot.coo[, 1]

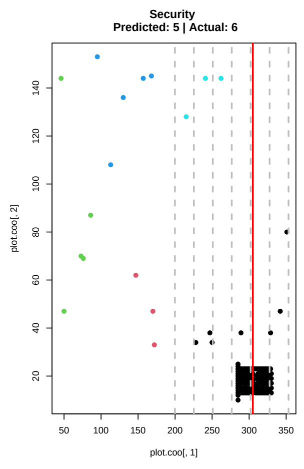

Security
Predicted: 1 | Actual: 4
plot.coo[, 2]
plot.coo[, 1]

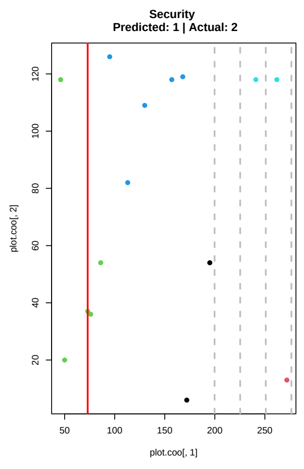

Security
Predicted: 7 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

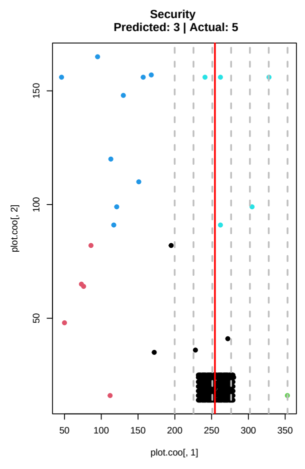

Security
Predicted: 5 | Actual: 6
plot.coo[, 2]
plot.coo[, 1]

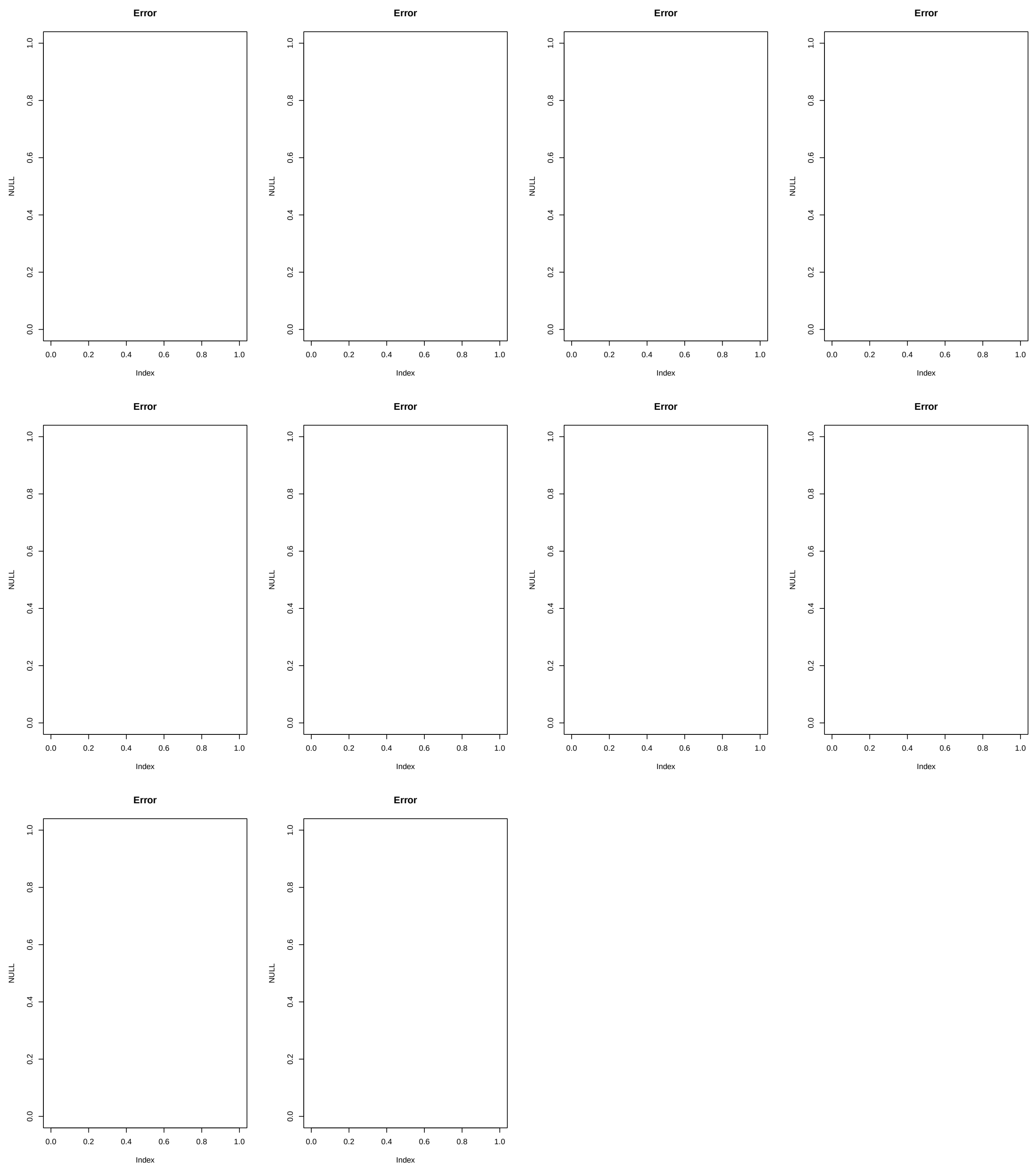

Security
Predicted: 1 | Actual: 4
plot.coo[, 2]
plot.coo[, 1]


Security
Predicted: 1 | Actual: 3
plot.coo[, 2]
plot.coo[, 1]


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


Security
Predicted: 1 | Actual: 2
plot.coo[, 2]
plot.coo[, 1]


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


Error
NULL
Index
0.0
0.2
0.4
0.6
0.8
1.0