KID
Shade Extraction

Fabian Blasch

08.08.2021

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, ...))
```

Import KIDs

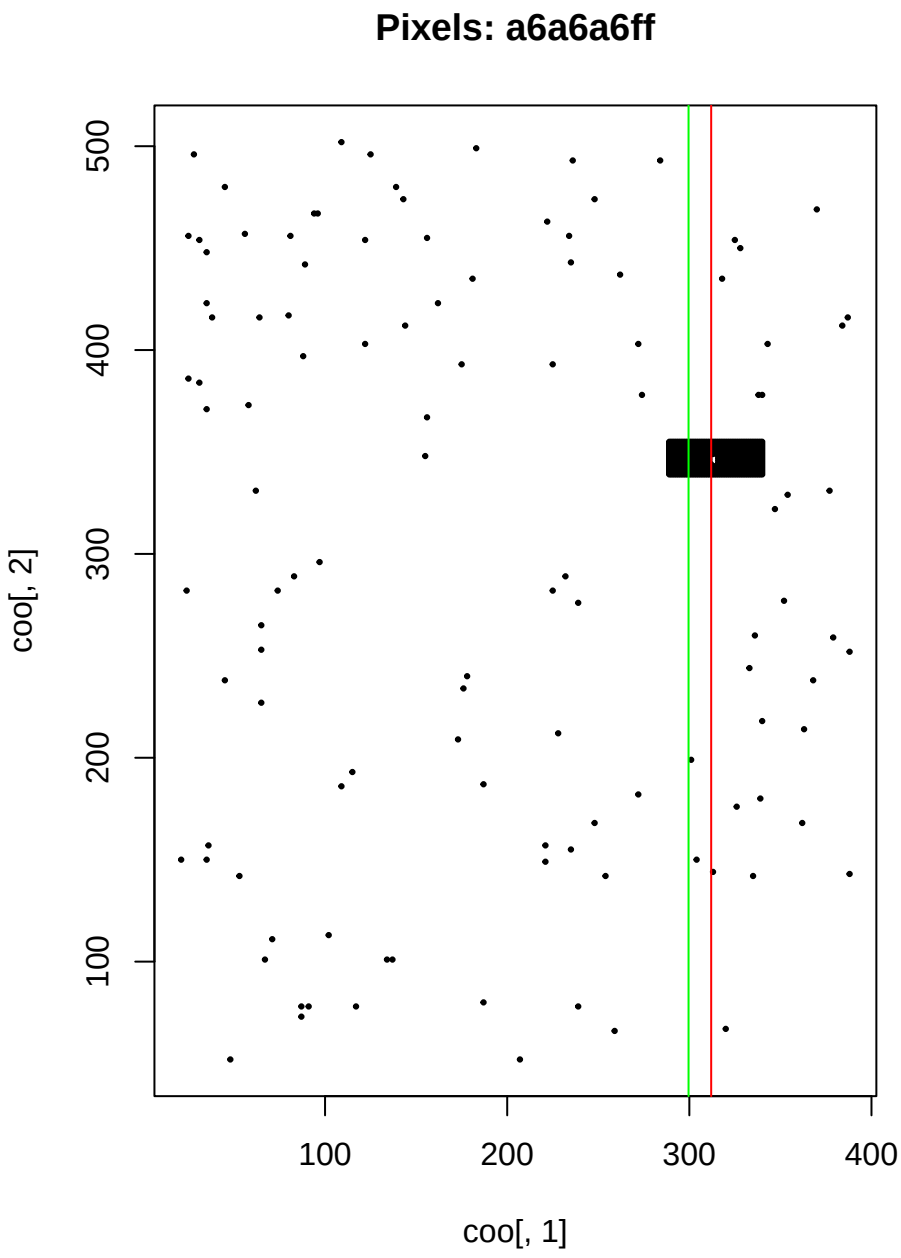
```
# set
setwd("C:/Users/blasc/OneDrive/Documents/GitHub/KID/KIDs")

# all PDF files in the current directory
file_names <- list.files(pattern = ".pdf")

# extract text split by linebreak
lapply(file_names, function(x){

  # split by line break and extract text
  strsplit(pdftools::pdf_text(x), "\n")

}) -> pdf.text.list
```

```r
# import PDF and convert to Png
lapply(file_names, function(x){

  # convert first page of pdf to bitmap
  pdftools::pdf_render_page(x, page = 1, dpi = 50)

}) -> bitmap.list

# to JPG
jpeg::writeJPEG(bitmap.list[[1]], "test.jpeg")

# JPEG
imt <- jpeg::readJPEG("test.jpeg")
```

Extract SSRI

Bitmap

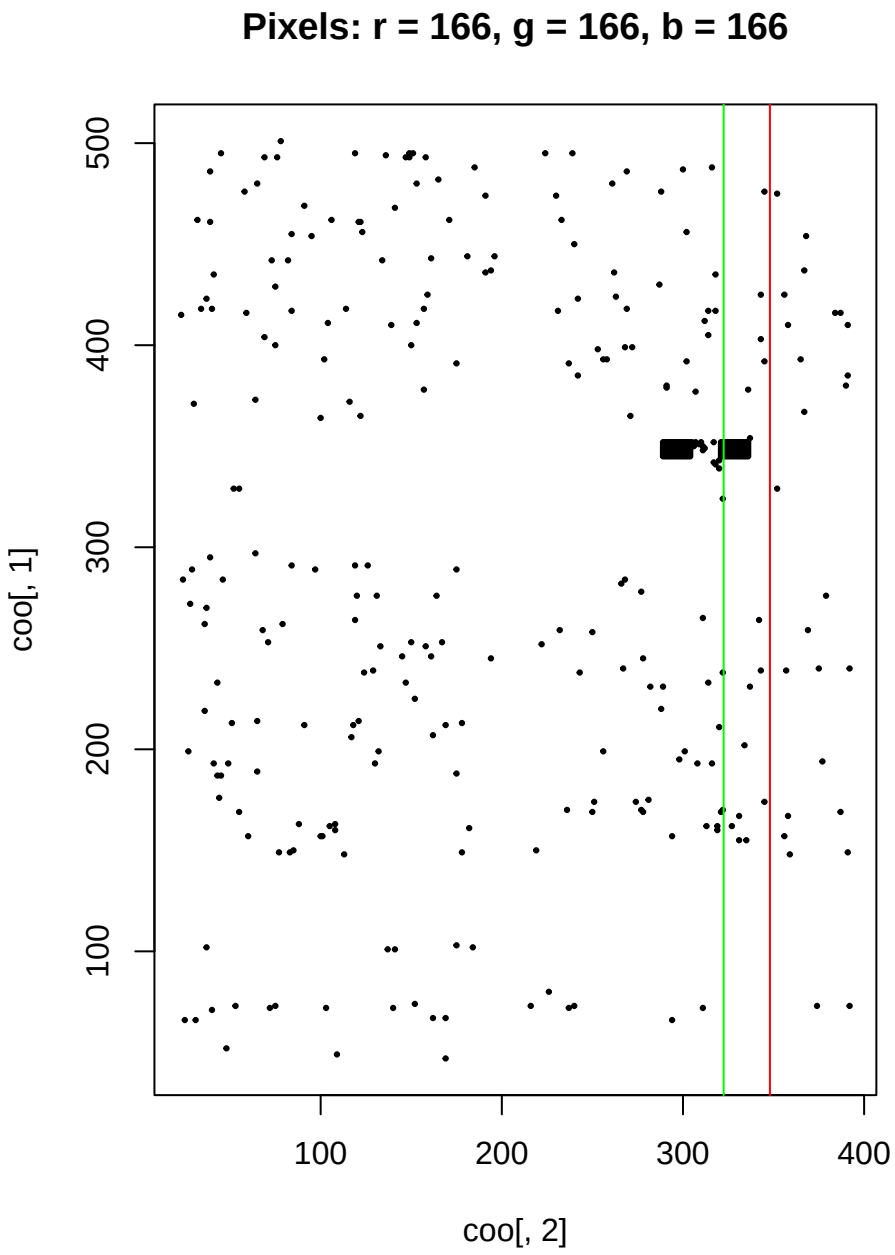
```r
# bitmap of first fund
bitmp1 <- bitmap.list[[1]]

# col
# bitmp1[,587, 690]

# coordinates
coo <- which(bitmp1[1,,] == "a6" & bitmp1[2,,] == "a6" & bitmp1[3,,] == "a6" &
             bitmp1[4,,] == "ff", arr.ind = T)

# plot
{
plot(coo[,1], coo[, 2], main = "Pixels: a6a6a6ff", pch = 19, cex = 0.3)
abline(v = median(coo[, 1]), col = "red")
abline(v = mean(coo[, 1]), col = "green")
}
```



JPEG

```
# convert by alpha value in this case 255
imt <- imt * 255

# coordinates
    # coo <- which((0 < imt[,, 1]  & imt[,, 1] < 10) &
    #              (75 < imt[,, 2] & imt[,, 2] < 85) &
    #              (130 < imt[,, 3] & imt[,, 3] < 150), arr.ind = T)
```

```r
coo <- which(imt[,,1] == 166 & imt[,,2] == 166 & imt[,,3] == 166, arr.ind = T)

# Pixel / Coordinateplot
{
plot(coo[,2], coo[, 1], main = "Pixels: r = 166, g = 166, b = 166", pch = 19, cex = 0.3)
abline(v = median(coo[, 1]), col = "red")
abline(v = mean(coo[, 1]), col = "green")
}
```

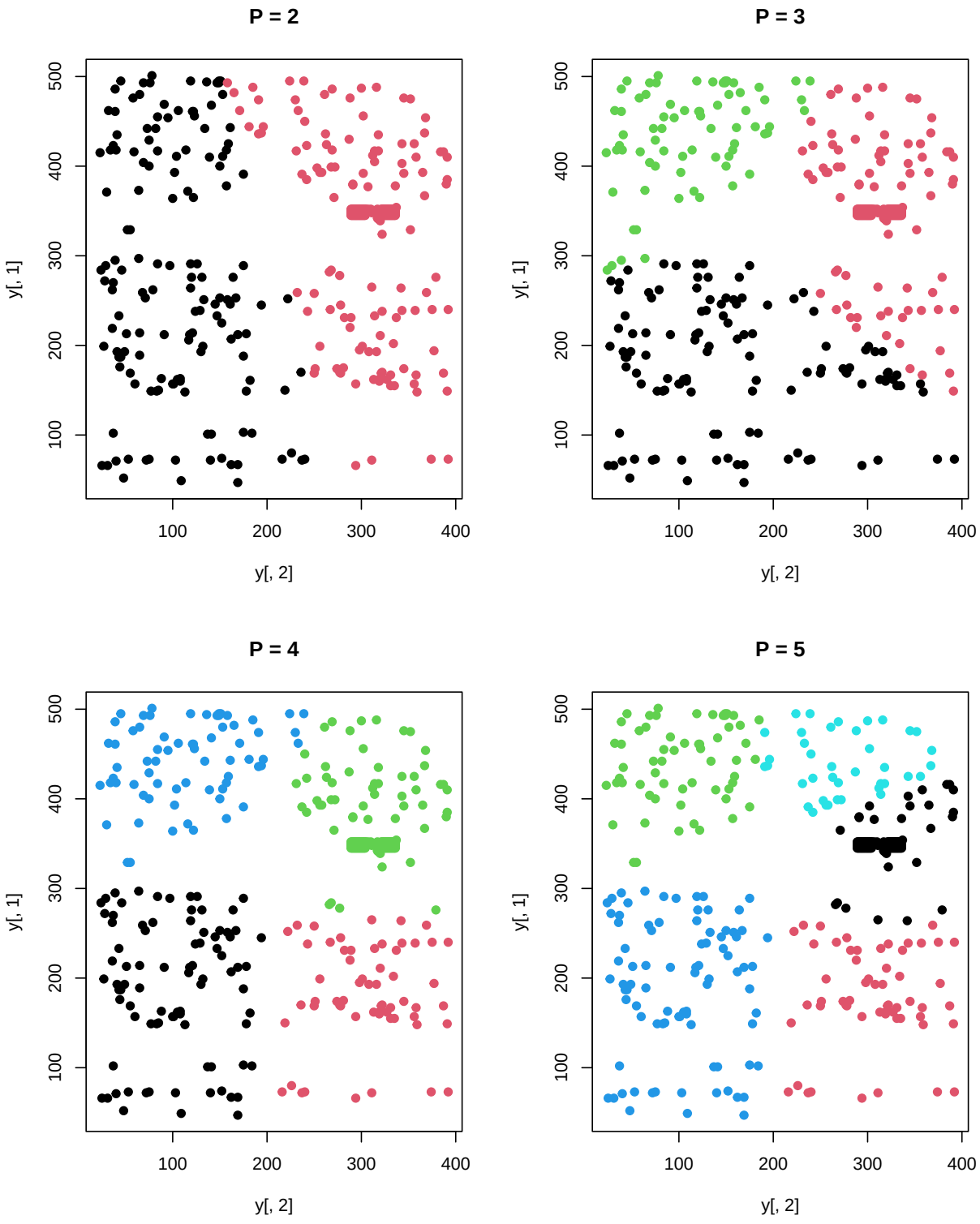


Classify utilizing k-means.

```
# Classify with different amount of groups then check for sil coef

# amt of groups
p <- 2:5

# estimate
lapply(p, function(x){

  # merge cluster into df
  dat <- cbind(coo, kmeans(coo, x)$cluster)

  # silhouette
  tmp1 <- cluster::silhouette(dat[, ncol(dat)], dist(coo))

  # return SC and Data
  list(
  "SC" = max(tapply(tmp1[, "sil_width"], tmp1[, "cluster"], mean)),
  "dat" = dat)

}) -> dat.kmeans

# sil coef
sapply(dat.kmeans, "[[", 1)
```

```
## [1] 0.6955551 0.7567326 0.7792204 0.7554876
```

```
# data.frames with the classification of different amt. of groups
# arrange
par(mfrow = c(2, 2))

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

  plot(y[, 2], y[, 1], col = y[, ncol(y)], pch = 19, main = paste("P =", x))

}, p, lapply(dat.kmeans, "[[", 2))
```



Classify utilizing hierarchical clustering

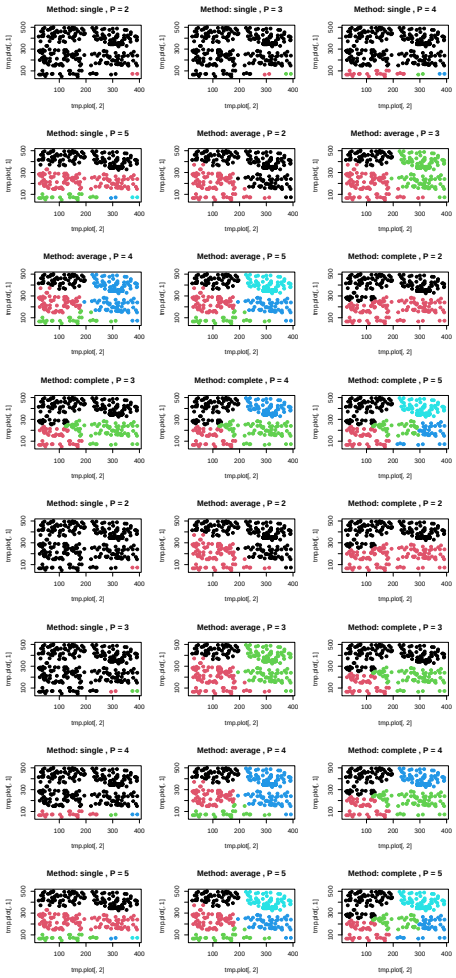
```
# methods
meth <- c("single", "average", "complete")
```

```r
# ramp up p
p <- 2:5

# estimate
# over methods
lapply(meth, function(x){

  # over p
  lapply(p, function(y){

    # get grouping
    tmp1 <- agnes(coo, method = x, diss = F)

    # restrict amnt of groups
    tmp2 <- cutree(tmp1, k = y)

    # bind
    tmp3 <- cbind(coo, tmp2)

    # calculate coefficients
    tmp4 <- silhouette(tmp3[, ncol(tmp3)], dist(tmp3[, 2:3]))

    # SC
    SC <- max(tapply(tmp4[, "sil_width"], tmp4[, "cluster"], mean))

    # return
    list("Data" = tmp3,
    "SC" = SC,
    "tmp.plot.silhouette" = tmp4)

  }) |> setNames(nm = paste("P =", p))

}) |> setNames(nm = meth) -> Group.list

# SC
lapply(Group.list, \(x){
  sapply(x, "[[", "SC")
})
```

```
## $single
##    P = 2    P = 3    P = 4    P = 5
## 0.861175 0.773586 0.773586 0.773586
##
## $average
##     P = 2     P = 3     P = 4     P = 5
## 0.5673892 0.8306566 0.7953262 0.4068845
##
## $complete
##     P = 2     P = 3     P = 4     P = 5
## 0.3356607 0.6844499 0.6189983 0.3368753
```

```r
# plot
# arrange
par(mfrow = c(14, 3))
```

```r
# plot
# over methods
invis.lapply(meth, \(x){

  # over p
  invis.lapply(paste("P =", p), \(y){

    # Data
    tmp.plot <- Group.list[[x]][[y]][["Data"]]

    # plot
    plot(tmp.plot[, 2], tmp.plot[, 1], col = tmp.plot[, ncol(tmp.plot)], pch = 19,
    main = paste("Method:", x, ",", y))

  })
})

invis.lapply(paste("P =", p), \(x){

  # over p
  invis.lapply(meth, \(y){

    # Data
    tmp.plot <- Group.list[[y]][[x]][["Data"]]

    # plot
    plot(tmp.plot[, 2], tmp.plot[, 1], col = tmp.plot[, ncol(tmp.plot)], pch = 19,
    main = paste("Method:", y, ",", x))

  })
})
```


Method: single , P = 2
Method: single , P = 3
Method: single , P = 4
Method: single , P = 5
Method: average , P = 2
Method: average , P = 3
Method: average , P = 4
Method: average , P = 5
Method: complete , P = 2
Method: complete , P = 3
Method: complete , P = 4
Method: complete , P = 5
Method: single , P = 2
Method: average , P = 2
Method: complete , P = 2
Method: single , P = 3
Method: average , P = 3
Method: complete , P = 3
Method: single , P = 4
Method: average , P = 4
Method: complete , P = 4
Method: single , P = 5
Method: average , P = 5
Method: complete , P = 5
tmp.plot[, 1]
tmp.plot[, 2]