Lab08

STAT 131A Spring 2023

In the lab we will cover

1. Pairs plots
2. PCA

Pairs Plots

We will start by reading in the data of college information

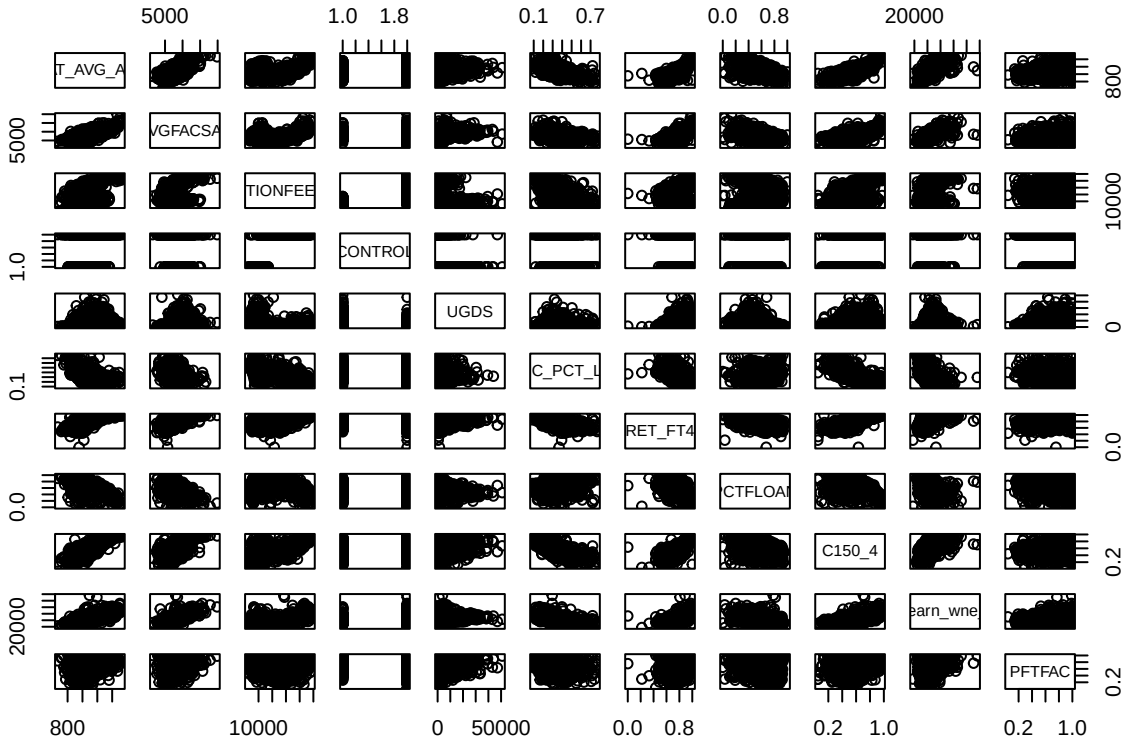
```
scorecard <- read.csv(("college.csv"), stringsAsFactors = FALSE,na.strings = c("NA","PrivacySuppressed"))
```

We will exclude those that are for-profit institutes:

```
scorecard<-scorecard[-which(scorecard$CONTROL==3),]
```

Now we will do a pairs plot. The default pairs plot requires a *matrix* of *numeric* values, and some of our values are factors. So we will leave those off (the first 3 columns), as well as a number of other variables. Notice how I can remove a range of values with the `-` (and how I have to put `c()` around it).

```
smallScores<-scorecard[,-c(1:3,4,5,6,9,11,14:17,18:22,24:27,31)]
pairs(smallScores)
```

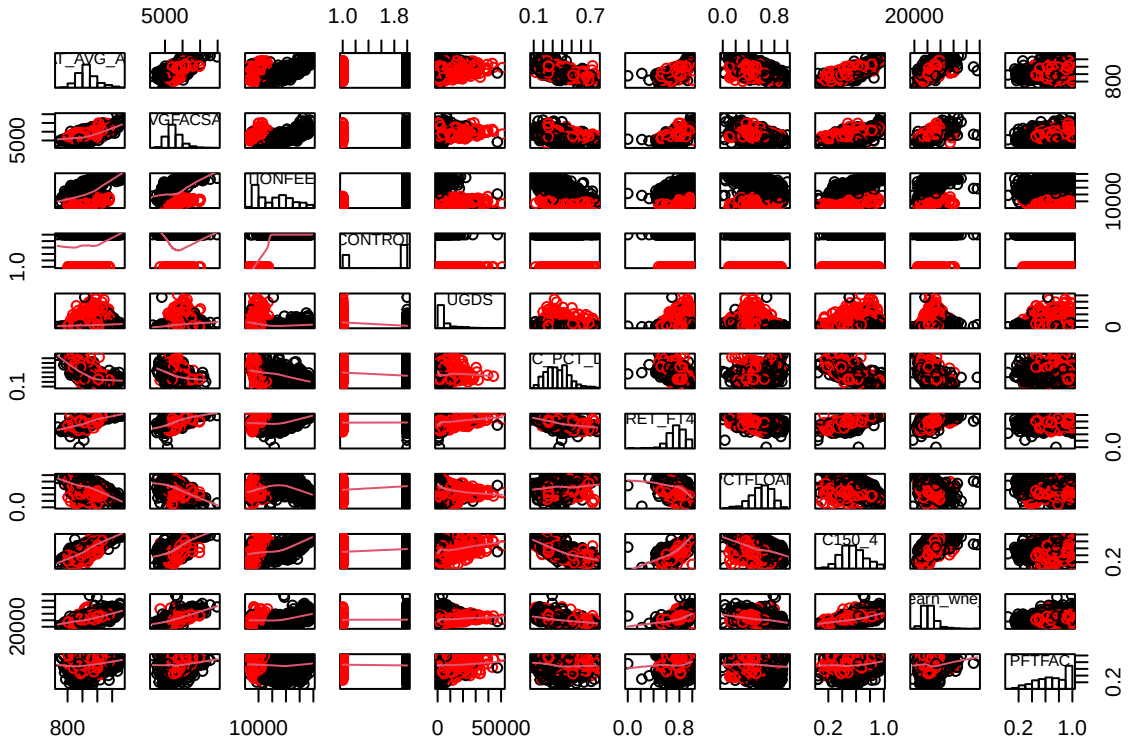


We can improve the `pairs` function by providing alternative functions to draw on the lower/upper/or diagonal plots. Here we use a function provided in the help of `pairs` to draw a histogram of the variable on the diagonal, instead of just printing the name. A function we will see later (`gpairs`) does this automatically, so don't stress about following this.

```
panel.hist <- function(x, ...)
{
    #from help of pairs
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y)
}
```

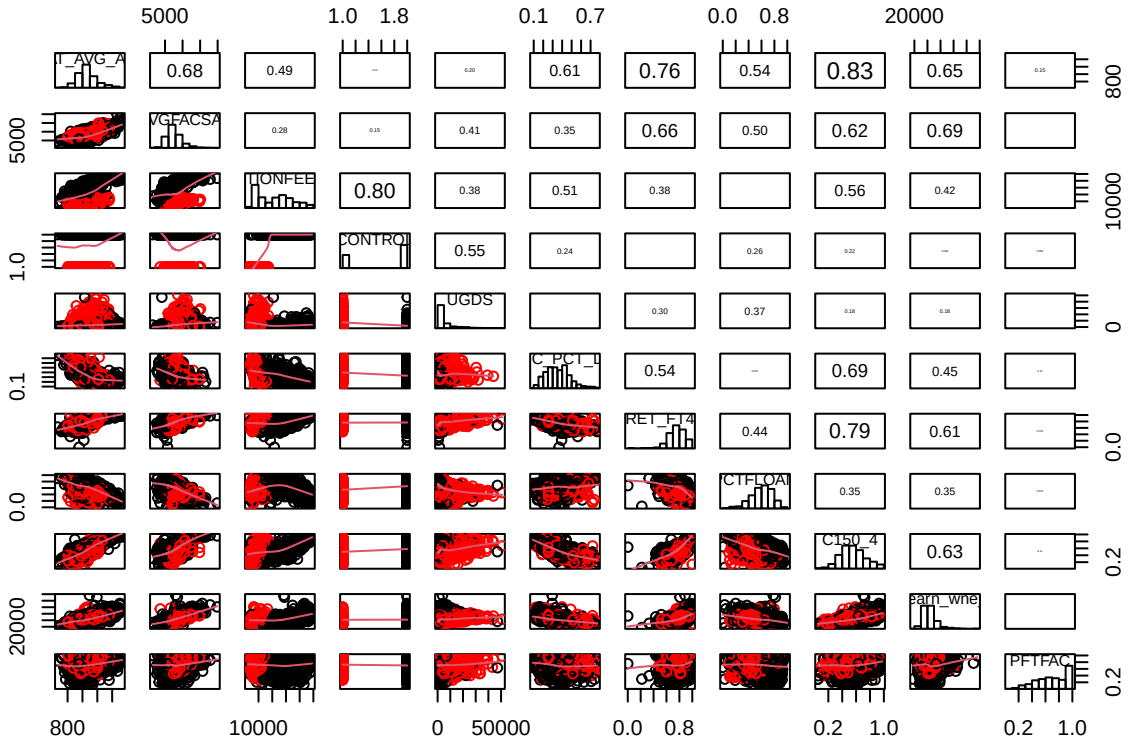
Now we call pairs, and we set the argument `diag.panel` to be the function we just wrote. We also can make the scatterplots have a loess line through them by using `panel.smooth` as our function (this is a built in function like `smooth.scatter`). We also color the points as to whether they are public or private using the `col` argument.

```
pairs(smallScores, lower.panel = panel.smooth, col=c("red","black")[smallScores$CONTROL],diag.panel = pa
```



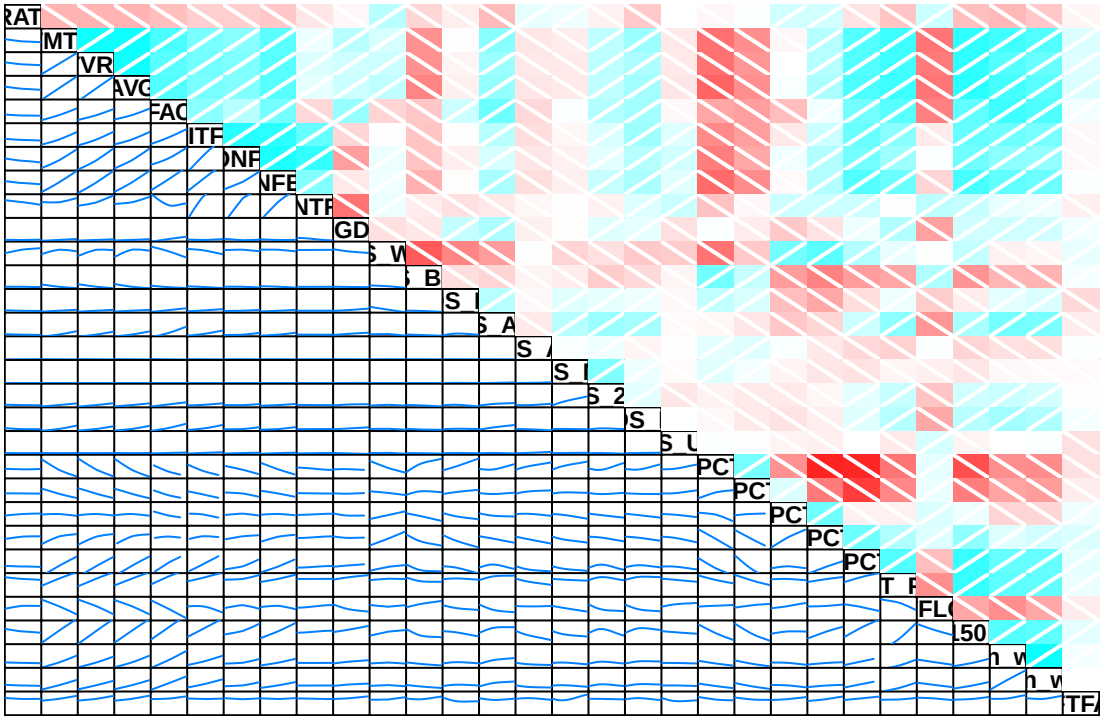
Now we make the upper and lower panels different. The upper panel is now going to print the correlation of the variables, where the size of the correlation will be relative to the size of the correlation (again, this is a function from the help of `pairs`). A function we will see later (`gpairs`) does this automatically, so don't stress about following this.

```
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    #from help of pairs
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y,use="pairwise.complete.obs"))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}
pairs(smallScores, lower.panel = panel.smooth, upper.panel = panel.cor,col=c("red","black")[smallScores
```



Using the function `corrgram` we can visualize the correlations for a large number of variables. This take a bit of time to render. It also outputs an enormous number of warnings, so we are going to suppress those warnings so as to not clutter up this document.

```
#install.packages("gpairs)
library(gpairs)
suppressWarnings(corrgram(scorecard[,-c(1:3)]))
```



Excercise 1

Read in data.

```
BikeData <- read.csv("DailyBikeSharingDataset.csv")
```

(a) Generate a matrix of scatter plots for all variable pairs in `BikeData` that are of class `numeric` using the `pairs()` function.

```
# Insert your code for plotting here
```

(b) Generate a matrix of scatter plots for all variable pairs in `BikeData` using the `gpairs()` function in the `gpairs` package.

```
library(gpairs)
# Insert your code for plotting here
```

A simple PCA example

This is a simple example for PCA from R Bloggers. The iris dataset is perhaps the best known dataset for classification. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.

```
# Load data
data(iris)
head(iris, 3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

There are five variables in the dataset `iris`, where `Sepal.Length`, `Sepal.Width`, `Petal.Length` and `Petal.Width` are continuous and `Species` is categorical.

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

Apply PCA to the four continuous variables

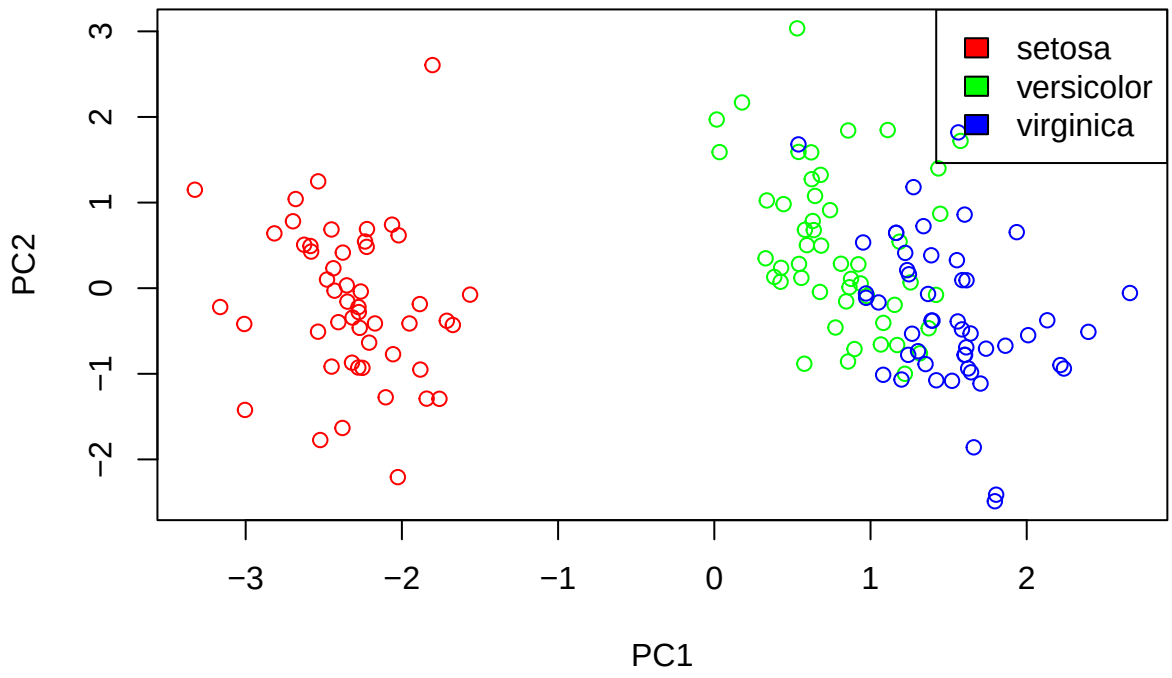
```
# log transform on the four continuous variable
log.ir <- log(iris[, 1:4])
# the Species variable
ir.species <- iris[, 5]

# apply PCA - scale. = TRUE is highly advisable, but default is FALSE.
# center and scale is used to standardize the variables prior to the application of PCA.
ir.pca <- prcomp(log.ir,
                 center = TRUE,
                 scale. = TRUE)
```

Plot the PC1 and PC2.

```
colorvec <- c("red", "green", "blue")
names(colorvec) = unique(ir.species)

plot(ir.pca$x[, 1], ir.pca$x[, 2], col = unname(colorvec[ir.species]), xlab = "PC1", ylab = "PC2")
legend("topright", legend = names(colorvec), fill = unname(colorvec))
```



Lets add a few NA values to the data.

```
iris[2,3]= NA
iris[5,2]= NA
iris[10,1] = NA
```

Apply PCA as before, but make change to code to address the missing values.

```
log.ir <- log(iris[, 1:4])
ir.species <- iris[, 5]

#ir.pca <- prcomp(log.ir,
#                 center = TRUE,
#                 scale. = TRUE)
```

Excercise 2

Read Data.

```
bodyFatData <- read.csv("bodyfat_short.csv")
```

(a) Run PCA on the data using the `prcomp()` function. Save the result to `res.pca`.

```
# insert your code here to run pca,

# res.pca <-
```

(b) Create a scatter plot of the first principal component versus the second principal component.

```
# insert your code here for the scatter plot
```

(c) Use `summary` function on your `res`.pca‘ object. How many PCs will you choose in order to explain at least 80% of the data variance?

```
# insert your code here to summary your PCs
```

(d) Use `screenplot` function to plot variances against the number of components

```
# insert your code here to plot
```