###############################################################################

# Description : CA for B8IT109 Advanced Data Analytics

# Lecturer : Dr Shahram Azizi

# Author : Barry Sheppard - Student Number 10387786

# Date : 2019/08/18

# Notes : Question 1

###############################################################################

#######################################

# Normal prep code #

#######################################

# This clears the workspace environment

rm(list = ls())

# This sets the working directory to the same as the file

setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# This installs all the packages needed if not already loaded

if (!require("pacman")) install.packages("pacman")

pacman::p\_load("psych")

# Question 1

# Use in-built dataset ‘airquality’,

# a)explore the general feature of dataset using appropriate R functions.

# (5 Marks)

# b)perform data cleansing if required.

# (5 Marks)

# c)consider ‘Temp’ attributes and compute the central and variational

# measures.

# (10 Marks)

# d)apply boxplot technique to detect outlier of ‘wind’ attribute if any.

# (10 Marks)

data("airquality")

df <- airquality

# Part a)

# First we look at the dataset structure

str(df)

# 'data.frame': 153 obs. of 6 variables:

# $ Ozone : int 41 36 12 18 NA 28 23 19 8 NA ...

# $ Solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...

# $ Wind : num 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...

# $ Temp : int 67 72 74 62 56 66 65 59 61 69 ...

# $ Month : int 5 5 5 5 5 5 5 5 5 5 ...

# $ Day : int 1 2 3 4 5 6 7 8 9 10 ...

# we can look at some of the variable values using the psych package describe

describe(df)

# vars n mean sd median trimmed mad min max range skew kurtosis

# Ozone 1 116 42.13 32.99 31.5 37.80 25.95 1.0 168.0 167 1.21 1.11

# Solar.R 2 146 185.93 90.06 205.0 190.34 98.59 7.0 334.0 327 -0.42 -1.00

# Wind 3 153 9.96 3.52 9.7 9.87 3.41 1.7 20.7 19 0.34 0.03

# Temp 4 153 77.88 9.47 79.0 78.28 8.90 56.0 97.0 41 -0.37 -0.46

# Month 5 153 6.99 1.42 7.0 6.99 1.48 5.0 9.0 4 0.00 -1.32

# Day 6 153 15.80 8.86 16.0 15.80 11.86 1.0 31.0 30 0.00 -1.22

# se

# Ozone 3.06

# Solar.R 7.45

# Wind 0.28

# Temp 0.77

# Month 0.11

# Day 0.72

# Part b

# Lets see if there are any missing values

df[!complete.cases(df),]

# So yeah, we can see there are rows with missing data

# Lets look at it by column and get some counts.

sum(is.na(df$Ozone)) # 37 missing items

sum(is.na(df$Solar.R)) # 7 missing items

sum(is.na(df$Wind)) # 0 missing items

sum(is.na(df$Temp)) # 0 missing items

sum(is.na(df$Month)) # 0 missing items

sum(is.na(df$Day)) # 0 missing items

# As we are going to be looking at the Temp and Wind attributes which have

# no missing items, we can conclude there is no need to have concern

# about the missing balues for Ozone and Solar.R

# Part c)

describe(df$Temp)

# vars n mean sd median trimmed mad min max range skew kurtosis se

# X1 1 153 77.88 9.47 79 78.28 8.9 56 97 41 -0.37 -0.46 0.77

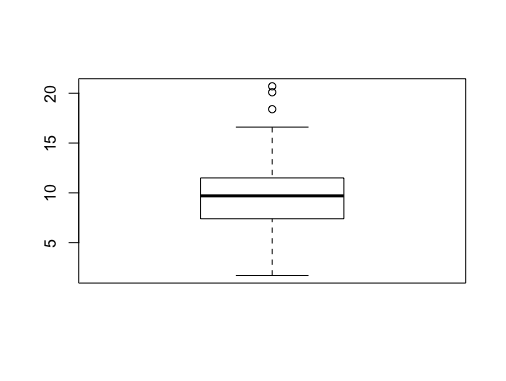
mean(df$Temp)

sd(df$Temp)

# the mean temperature is 77.88 with a standard deviation of 9.47

# Part d)

boxplot(df$Wind)



# From the pot we can there are 3 outliers at the top of the range

tail(sort(df$Wind), 3)

# 18.4 20.1 20.7

###############################################################################

# Description : CA for B8IT109 Advanced Data Analytics

# Lecturer : Dr Shahram Azizi

# Author : Barry Sheppard - Student Number 10387786

# Date : 2019/08/18

# Notes : Question 2

###############################################################################

#######################################

# Normal prep code #

#######################################

# This clears the workspace environment

rm(list = ls())

# This sets the working directory to the same as the file

setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# This installs all the packages needed if not already loaded

if (!require("pacman")) install.packages("pacman")

pacman::p\_load('psych', 'caret', 'e1071')

# Question 2

# Use dataset available on http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv , then:

# (a) Train the model using 80% of this dataset and suggest an appropriate GLM to model homekick to togo, ydline and kicker variables.

# (b) Specify the significant variables on homekick at the level of

# 𝛼=0.05, and estimate the parameters of your model.

# (c) Predict the test dataset using the trained model.

# (d) Provide the confusion matrix and obtain the probability of correctness of predictions.

# First we load and review the dataset

df <- read.csv("http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv")

head(df)

names(df)

str(df)

describe(df)

# (a) Train the model using 80% of this dataset and suggest an

# appropriate GLM to model homekick to togo, ydline and kicker

# variables.

df <- na.omit(df)

# Split the dataset into 80 train and 20 test.

n <- nrow(df)

indexes <- sample(n,n\*(80/100))

trainset <- df[indexes,]

testset <- df[-indexes,]

# To determine what #type of model to use we need to look

# at the ourcome variable homekick

str(df$homekick)

max(df$homekick) #1

min(df$homekick) #0

# As the outcome is binary, we create a glm model using binomial

# family

model <- glm(homekick ~ togo + ydline + kicker, data = trainset, family = 'binomial')

# (b) Specify the significant variables on homekick at the level

# of 𝛼=0.05, and estimate the parameters of your model

summary(model)

# Call:

# glm(formula = homekick ~ togo + ydline + kicker, family = "binomial",

# data = trainset)

#

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -1.3384 -1.1532 -0.9887 1.1876 1.4596

#

# Coefficients:

# Estimate Std. Error z value Pr(>|z|)

# (Intercept) 0.13064 0.20628 0.633 0.5265

# togo -0.04219 0.01795 -2.351 0.0187 \*

# ydline 0.01156 0.00753 1.535 0.1247

# kicker -0.00550 0.00620 -0.887 0.3750

# ---

# Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

#

# (Dispersion parameter for binomial family taken to be 1)

#

# Null deviance: 1150.1 on 829 degrees of freedom

# Residual deviance: 1143.2 on 826 degrees of freedom

# (1 observation deleted due to missingness)

# AIC: 1151.2

#

# Number of Fisher Scoring iterations:

# at alpha of 0.05 only togo is significant

# Using this we can estimate the parameters as

# Intercept = 0 (as non-sig), beta for togo is - 0.044219

# all other non-sigificant parameters are set to 0

# homekick = 0 - 0.044219 \* togo

# Let's rerun the model removing the other non-signifcant items

model2 <- glm(homekick ~ togo, data=trainset, family='binomial')

summary(model2)

#

# Call:

# glm(formula = homekick ~ togo, family = "binomial", data = trainset)

#

# Deviance Residuals:

# Min 1Q Median 3Q Max

# -1.237 -1.166 -1.025 1.188 1.419

#

# Coefficients:

# Estimate Std. Error z value Pr(>|z|)

# (Intercept) 0.17100 0.13233 1.292 0.1963

# togo -0.03283 0.01691 -1.942 0.0522 .

# ---

# Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

#

# (Dispersion parameter for binomial family taken to be 1)

#

# Null deviance: 1150.1 on 829 degrees of freedom

# Residual deviance: 1146.3 on 828 degrees of freedom

# (1 observation deleted due to missingness)

# AIC: 1150.3

#

# Number of Fisher Scoring iterations: 3

# This model ends up as

# homekick = 0.17100 - 0.03283 \* togo

# although, interestingly in this summary togo is no longer

# significant at the 0.05 level!

# (c) Predict the test dataset using the trained model.

predicted\_data <- predict(model2,testset, type='response')

# convert that to 1s and 0s

p\_data <- as.integer(predicted\_data > 0.5)

# (d) Provide the confusion matrix and obtain the probability

# of correctness of predictions.

# We will us use caret to compute a confusion matrix

confusionMatrix(data = as.factor(p\_data),

reference = as.factor(testset$homekick))

# # Confusion Matrix and Statistics

#

# Reference

# Prediction 0 1

# 0 52 60

# 1 48 47

#

# Accuracy : 0.4783

# 95% CI : (0.4085, 0.5486)

# No Information Rate : 0.5169

# P-Value [Acc > NIR] : 0.8814

#

# Kappa : -0.0406

#

# Mcnemar's Test P-Value : 0.2898

#

# Sensitivity : 0.5200

# Specificity : 0.4393

# Pos Pred Value : 0.4643

# Neg Pred Value : 0.4947

# Prevalence : 0.4831

# Detection Rate : 0.2512

# Detection Prevalence : 0.5411

# Balanced Accuracy : 0.4796

#

# 'Positive' Class : 0

# So this model has a 0.4783 prediction rate.

# The below is another way to do accuracy

tab <- table(p\_data, testset$homekick) # Confusion matrix

tab

# p\_data 0 1

# 0 52 60

# 1 48 47

model\_accuracy <- sum( tab[row(tab) == col(tab)] ) / sum(tab)

model\_accuracy # 0.4783

###############################################################################

# Description : CA for B8IT109 Advanced Data Analytics

# Lecturer : Dr Shahram Azizi

# Author : Barry Sheppard - Student Number 10387786

# Date : 2019/08/18

# Notes : Question 3

###############################################################################

#######################################

# Normal prep code #

#######################################

# This clears the workspace environment

rm(list = ls())

# This sets the working directory to the same as the file

setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# This installs all the packages needed if not already loaded

if (!require("pacman")) install.packages("pacman")

pacman::p\_load('quantmod', 'xts', 'ggplot2', 'tseries', 'forecast')

# Question 3

# Using Yahoo Finance API, select a specific stock market price,

# apply time series analysis, consider ‘close price as your time

# series variable:

# (a) Validate the assumptions using graphical visualization.

# (b) Fit the optimized model for ‘close price’ and provide

# the coefficient estimates for the fitted model.

# (c) What is the estimated order for AR and MA?

# (d) Forecast h=10 step ahead prediction of wage on the plot

# of the original time series.

# Code to read data from Yahoo Finance from

# https://lamfo-unb.github.io/2017/07/22/intro-stock-analysis-1/

# Other time series code from this guide

# https://rpubs.com/JSHAH/481706

# First we need to fetch the data from Yahoo Finance

data <- getSymbols("GAW.L", src = "yahoo",

from = "2012-01-01", to = "2019-07-11",

auto.assign = FALSE)

head(data)

dim(data)

str(data)

# Remove all but the closing price

stock\_prices <- data[,4]

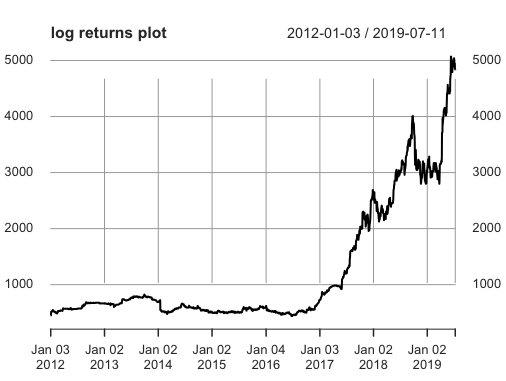
# check for any NAs

stock\_prices[is.na(stock\_prices)]

# (a) Validate the assumptions using graphical visualization.

# plot the data

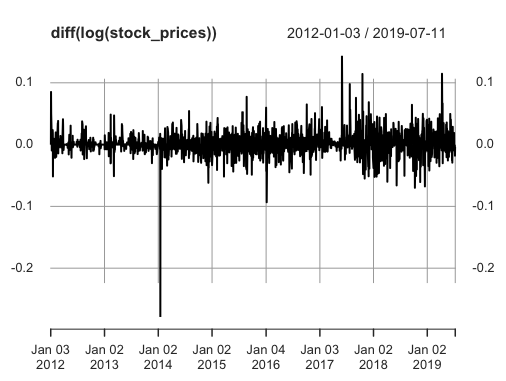
plot(stock\_prices, type = 'l', main = 'log returns plot')



# From the plot we can see the mean and variance is not stationary

# This is especially true when we compare to a normalised version

plot(diff(log(stock\_prices)))

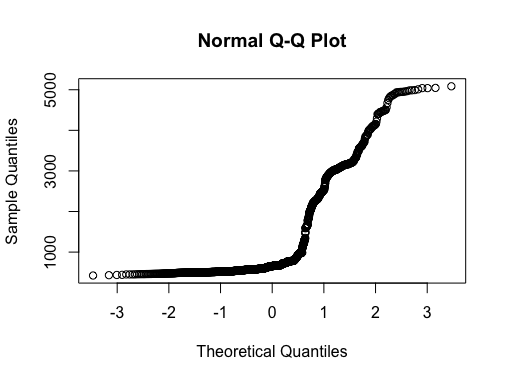


# We can also see from the qqnorm plot that it is not normal

qqnorm(stock\_prices, main = "Normal Q-Q Plot",

xlab = "Theoretical Quantiles",

ylab = "Sample Quantiles", plot.it = TRUE)



# although it's not necessarily correct on stock exchange data

# we can look to see if there is any seasonal trend

fit <- stl(stock\_prices, s.window = "periodic")

# This generates an error due to the lack of periods as its daily

periodicity(stock\_prices)

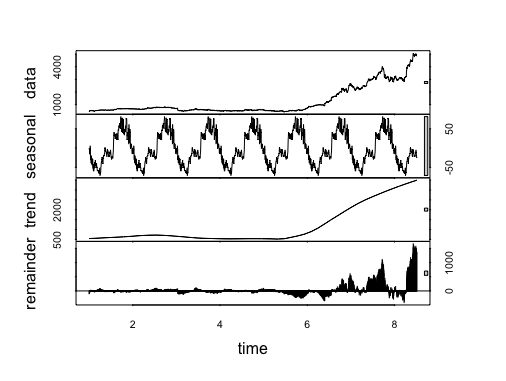
# Daily periodicity from 2012-01-03 to 2019-07-11

# Adjust it to have yearly frequency, there are ~253 trading days in the year

yearly\_stock\_prices <- ts(as.numeric(stock\_prices), frequency = 253)

fit <- stl(yearly\_stock\_prices, s.window = "periodic", robust = TRUE)

plot(fit)



# it seems there is no seasonal apparent as the data is very similar to the trend

# meaning the seasonal didn't account for much and the remainder is quite similar

# to the original data also, suggesting the forced seasonal trend accounts for very little

# (b) Fit the optimized model for ‘close price’ and provide

# the coefficient estimates for the fitted model.

auto.fit <- auto.arima(stock\_prices, seasonal = TRUE)

auto.fit

# Series: stock\_prices

# ARIMA(5,2,0)

#

# Coefficients:

# ar1 ar2 ar3 ar4 ar5

# -0.8656 -0.6400 -0.4687 -0.3844 -0.2153

# s.e. 0.0224 0.0286 0.0303 0.0286 0.0224

#

# sigma^2 estimated as 1435: log likelihood=-9594.14

# AIC=19200.27 AICc=19200.32 BIC=19233.57

# auto.fit <- auto.arima(stock\_prices, seasonal = FALSE)

# auto.fit

#

# Series: stock\_prices

# ARIMA(5,2,0)

#

# Coefficients:

# ar1 ar2 ar3 ar4 ar5

# -0.8656 -0.6400 -0.4687 -0.3844 -0.2153

# s.e. 0.0224 0.0286 0.0303 0.0286 0.0224

#

# sigma^2 estimated as 1435: log likelihood=-9594.14

# AIC=19200.27 AICc=19200.32 BIC=19233.57

# As seasonal false is the same as seasonal true, we use False

# The coeffecient estimates are

# Coefficients:

# ar1 ar2 ar3 ar4 ar5

# -0.8656 -0.6400 -0.4687 -0.3844 -0.2153

# s.e. 0.0224 0.0286 0.0303 0.0286 0.0224

# ar1 = -0.8656

# ar2 = -0.6400

# ar3 = -0.3844

# ar4 = -0.2153

# (c) What is the estimated order for AR and MA?

# The estimated order is ARIMA(5,2,0)

# p = 5

# d = 2

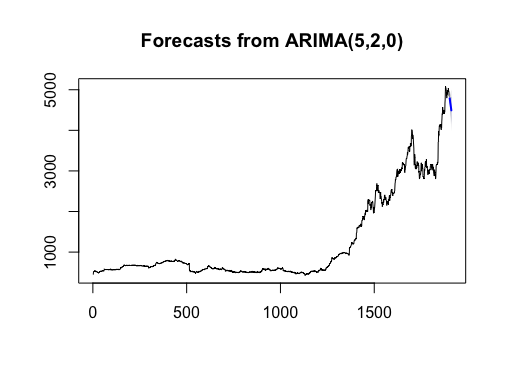
# q = 0

# (d) Forecast h=10 step ahead prediction of close price on the plot

# of the original time series.

auto.fcast <- forecast(auto.fit, h = 10)

plot(auto.fcast)



###############################################################################

# Description : CA for B8IT109 Advanced Data Analytics

# Lecturer : Dr Shahram Azizi

# Author : Barry Sheppard - Student Number 10387786

# Date : 2019/08/18

# Notes : Question 4

###############################################################################

#######################################

# Normal prep code #

#######################################

# This clears the workspace environment

rm(list = ls())

# This sets the working directory to the same as the file

setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# This installs all the packages needed if not already loaded

if (!require("pacman")) install.packages("pacman")

pacman::p\_load('psych', 'MASS', 'car', 'ggplot2', 'GGally', 'CCA', 'sem', 'cluster')

# Question 4

# Use dataset available on

# http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv

df <- read.csv('http://users.stat.ufl.edu/~winner/data/nfl2008\_fga.csv')

names(df)

head(df)

describe(df)

str(df)

#There are two rows incomplete

df[!complete.cases(data),]

# As this is only 2 out of 1039 observations, we choose to exclude these

data <- df[complete.cases(df),]

# 1. Use LDA to classify the dataset into few classes so that

# at least 90% of information of dataset is explained through

# new classification. (Hint: model the variable “qtr” to

# variables “togo”, “kicker”, and “ydline”). How many LDs do

# you choose? Explain the reason.

qtrlda <- lda(qtr ~ togo + kicker + ydline, data = data)

qtrlda

# Call:

# lda(qtr ~ togo + kicker + ydline, data = data)

#

# Prior probabilities of groups:

# 1 2 3 4 5

# 0.20636451 0.35969142 0.17550627 0.24590164 0.01253616

#

# Group means:

# togo kicker ydline

# 1 6.481308 19.64486 17.22897

# 2 6.973190 18.77212 19.30027

# 3 6.543956 19.96703 19.03297

# 4 6.792157 20.20000 18.53725

# 5 5.923077 22.61538 19.53846

#

# Coefficients of linear discriminants:

# LD1 LD2 LD3

# togo 0.06665269 0.12498308 0.20996464

# kicker -0.04134867 -0.06009657 0.05013225

# ydline 0.07726467 -0.07173243 -0.02257770

#

# Proportion of trace:

# LD1 LD2 LD3

# 0.615 0.322 0.063

0.615 + 0.322 # 0.937

# LD1 and LD2 combined explain 93.7% of the information and are selected

# 2. Apply PCA, and identify the important principle components

# involving at least 90% of dataset variation. Explain your

# decision strategy? Plot principle components versus their

# variance (Hint: to sketch the plot use the Scree plot).

# We start by removing any of the non-continuous data

# We also remove season as it has 0 variance

keep\_columns <- c('qtr', 'min', 'sec', 'down', 'togo','kicker','ydline','distance',

'homekick', 'kickdiff', 'timerem', 'offscore', 'defscore',

'GOOD', 'Missed', 'Blocked')

data <- data[,keep\_columns]

names(data)

describe(data)

fit <- princomp(data, cor = TRUE)

summary(fit)

# Importance of components:

# Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6

# Standard deviation 2.0048406 1.6721636 1.3083384 1.22081818 1.11060318 1.00083350

# Proportion of Variance 0.2512116 0.1747582 0.1069843 0.09314981 0.07708996 0.06260423

# Cumulative Proportion 0.2512116 0.4259698 0.5329542 0.62610397 0.70319393 0.76579816

# Comp.7 Comp.8 Comp.9 Comp.10 Comp.11 Comp.12

# Standard deviation 0.97841761 0.95110310 0.88207039 0.82066687 0.62552037 0.203651798

# Proportion of Variance 0.05983131 0.05653732 0.04862801 0.04209338 0.02445473 0.002592128

# Cumulative Proportion 0.82562948 0.88216680 0.93079481 0.97288819 0.99734293 0.999935053

# Comp.13 Comp.14 Comp.15 Comp.16

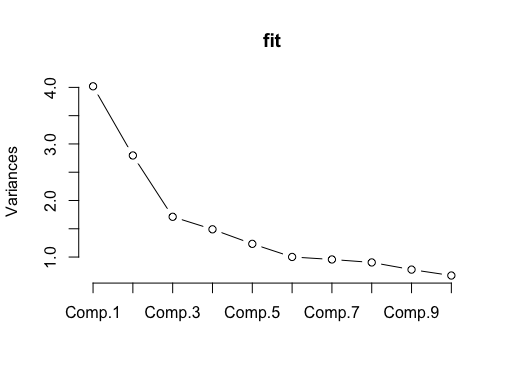
# Standard deviation 3.223577e-02 3.041048e-08 2.446684e-08 8.914025e-09

# Proportion of Variance 6.494657e-05 5.779982e-17 3.741415e-17 4.966240e-18

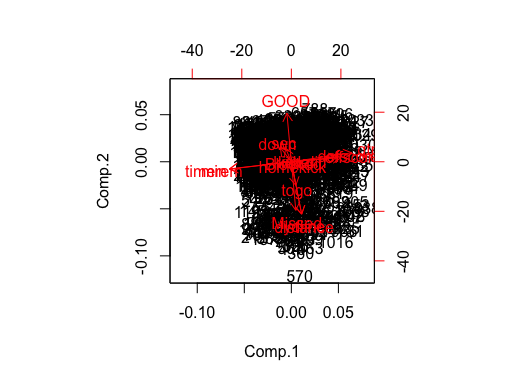
# Cumulative Proportion 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00

loadings(fit)

plot(fit, type = 'lines')



biplot(fit)



# From the screeplot we have a couple of options we could choose from

# Comp1 + Comp2 + Comp3 explains 53% of the data and the increase to 4 is only 9%

# Another possibile point of inflection could be 6 which explains 76% and the inrease

# to 7 is only 6%

# in this quesiton however, we're being asked to explain 90% of the dataset variation

# for that we need a total of 9 components which explain 93%

# 3. Split the dataset into two sets of variables so that

# X=( togo,kicker,ydline) and Y=( distance, homekick).

names(data)

x <- data[,5:7]

y <- data[,8:9]

names(x)

# [1] "togo" "kicker" "ydline"

names(y)

# [1] "distance" "homekick"

# Apply

# canonical correlation analysis to find the cross-correlation

# between X and Y. What is the correlation between ydline and

# distance?

# display the canonical correlations

cc1 <- cc(x, y)

cc1$cor

# [1] 0.99894989 0.06975549

cor(x, y) # correlation between two set of variables

# distance homekick

# togo 0.315641454 -0.04838438

# kicker -0.001951722 -0.02363159

# ydline 0.998947222 0.04295427

# The correlation betweel ydline and distance is 0.998947222

# So positively correlated and almost 1!

# Use K-means clustering analysis to identify the most

# important classes. How many classes do you select? Why?

# We use this wssplot function is to work out the lowest number of clusters

# with the highest amount of variation (information)

wssplot <- function(data, nc=10, seed=1234){

wss <- (nrow(data)-1)\*sum(apply(data,2,var))

for (i in 2:nc) {

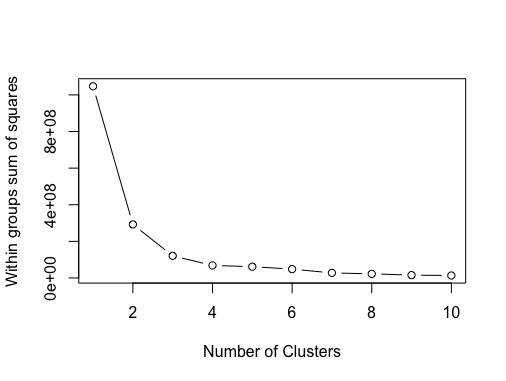
set.seed(seed)

wss[i] <- sum(kmeans(data, centers = i)$withinss)}

plot(1:nc, wss, type = "b", xlab = "Number of Clusters",

ylab = "Within groups sum of squares")}

wssplot(data, nc = 10)



# This scree plot gives us an indication of the number of clusters we should be

# looking for. In this example, after 4 there is no major change.

# We could also make a case for 3, but 4 appears to be a better point of inflection

# generate the clusters

k.means.fit <- kmeans(data, 4) # the 4 indicates the number of groups previous selected

# We can also plot this to visualise the 4 groups

library(cluster)

clusplot(data, k.means.fit$cluster, main = '2D representation of the Cluster solution',

color = TRUE, shade = TRUE,

labels = 2, lines = 0)

