WESTERN SYDNEY
UNIVERSITY

Computing, Engineering & Mathematics

ASSIGNMENT / REPORT COVER SHEET

This sheet must be attached to all material being submitted for marking.

Student name: **Student number:**	Quang Dong Nguyen 20744696	**Student name:** **Student number:**	
Sections completed individually	All questions	**Sections completed individually**	
Unit name & number:	Unit Name: Introduction to Data Science Number: COMP2025		
Tutorial day and time:	Tutorial: Monday Time: 11am - 1pm		
Title of Assignment:	Assignment-2		
Student Submitting the Assignment:	Quang Dong Nguyen		
Date submitted:	14/10/2022		

Student Declaration (must be signed)

Declaration:

☑ I hold a copy of this assignment if the original is lost or damaged.

☑ I hereby certify that no part of this assignment or product has been copied from any other student's work or from any other source except where due acknowledgement is made in the assignment.

☑ No part of the assignment/product has been written / produced for me by any other person except where collaboration has been authorised by the subject lecturer/tutor concerned

☑ I am aware that this work may be reproduced and submitted to plagiarism detection software programs for the purpose of detecting possible plagiarism ***(which may retain a copy on its database for future plagiarism checking)***

☑ I hereby certify that no part of this assignment or product has been submitted by me in another (previous or current) assessment, except where appropriately referenced, and with prior permission from the Lecturer/Tutor/ Unit Co-ordinator for this unit.

☑

Student signature and date:	 14/10/2022
Student signature and date	

Note: An examiner or lecturer/tutor has the right to not mark this assignment if the above declaration has not been signed.

Introduction to Data Science (Assignment 2)

Quang Dong Nguyen

2022-09-27

Question 1 - Data Visualisation

```
kc <- read.csv("kc_house2.csv")
kc_second <- kc
View(kc_second)
head(kc_second)
```

```
##           id bedrooms bathrooms sqft_living sqft_lot floors waterfront
## 1 7922800400        5      3.25        3.25   14.342      2          0
## 2 1516000055        3      2.25        2.15   21.235      1          0
## 3 2123039032        1      0.75        0.76   10.079      1          1
## 4 9297300045        3      2.00        1.97    4.166      2          0
## 5 1860600135        5      2.50        3.65    9.050      2          0
## 6 1560930070        4      3.50        2.84   40.139      1          0
##   sqft_living15 sqft_lot15 price_cat
## 1          2.96     11.044       low
## 2          2.57     18.900       low
## 3          1.23     14.267       low
## 4          2.39      4.166       low
## 5          2.88      5.400      high
## 6          3.18     36.852       low
```

```
#Check the number of dimension of the dataset:
dim(kc_second)
```

```
## [1] 340  10
```

```
# Varriable names check:
names(kc_second)
```

```
##  [1] "id"            "bedrooms"      "bathrooms"     "sqft_living"
##  [5] "sqft_lot"      "floors"        "waterfront"    "sqft_living15"
##  [9] "sqft_lot15"    "price_cat"
```

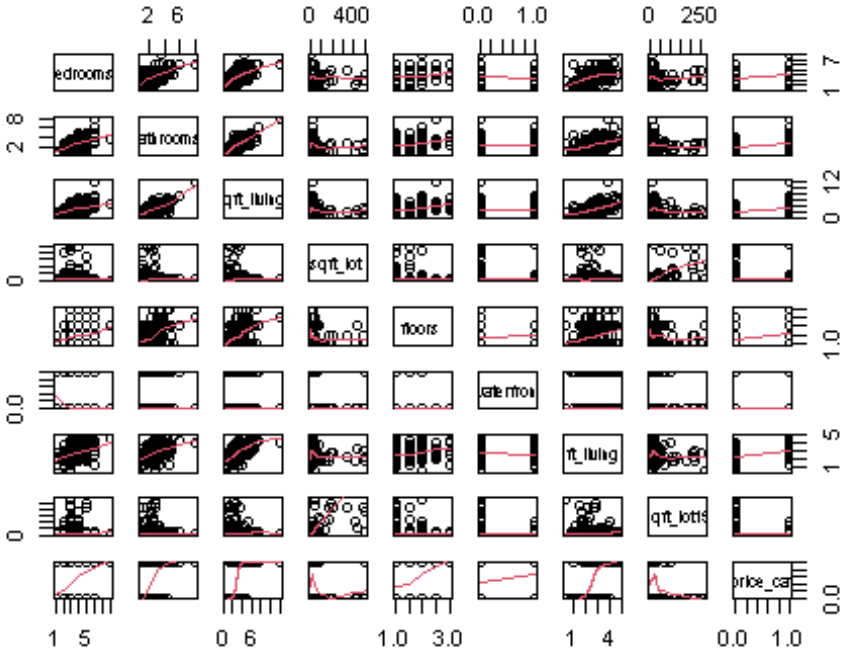
```
#Class check:
str(kc_second)
```

```
## 'data.frame':    340 obs. of  10 variables:
##  $ id           : num  7.92e+09 1.52e+09 2.12e+09 9.30e+09 1.86e+09 ...
##  $ bedrooms     : int  5 3 1 3 5 4 3 3 3 2 ...
##  $ bathrooms    : num  3.25 2.25 0.75 2 2.5 3.5 3 2.5 3 2.25 ...
##  $ sqft_living  : num  3.25 2.15 0.76 1.97 3.65 ...
##  $ sqft_lot     : num  14.34 21.23 10.08 4.17 9.05 ...
##  $ floors       : num  2 1 1 2 2 1 2 2 2 1.5 ...
##  $ waterfront   : int  0 0 1 0 0 0 1 1 0 0 ...
##  $ sqft_living15: num  2.96 2.57 1.23 2.39 2.88 3.18 2.28 3.98 3.08 2.13 .
```

```
..
##  $ sqft_lot15    : num  11.04 18.9 14.27 4.17 5.4 ...
##  $ price_cat     : chr  "low" "low" "low" "low" ...
```

Use appropriate data visualisation techniques and comment on the association of the price of King County houses with other characteristics of the house.

```
#Overall visualisation of the dataset kc_second
kc_second$price_cat <-ifelse(kc_second$price_cat == "high", 1 , 0)
pairs(kc_second[2:10], panel = panel.smooth)
```




```
#Correlation of price_cat with other characteristics of the house.
cor(kc_second[2:10])
```

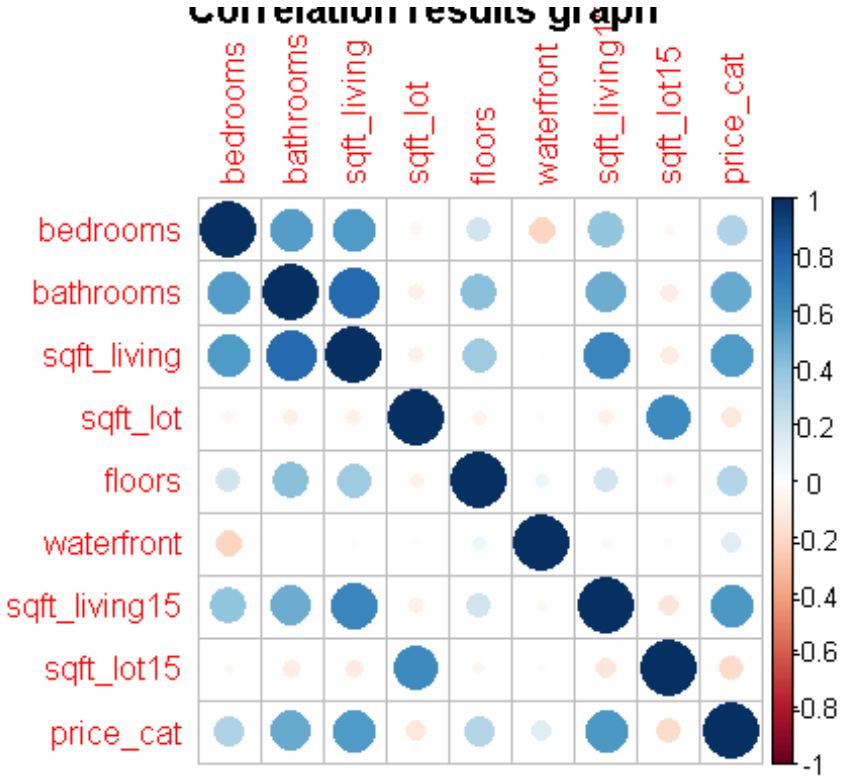
```
##                 bedrooms    bathrooms sqft_living    sqft_lot      floors
## bedrooms       1.00000000  0.558673511  0.56099484 -0.04046844  0.19324904
## bathrooms      0.55867351  1.000000000  0.77960328 -0.08296358  0.41683931
## sqft_living    0.56099484  0.779603277  1.00000000 -0.07841159  0.35602723
## sqft_lot      -0.04046844 -0.082963579 -0.07841159  1.00000000 -0.06659993
## floors         0.19324904  0.416839308  0.35602723 -0.06659993  1.00000000
## waterfront    -0.21868436 -0.005555848 -0.01519396  0.01571174  0.06713732
## sqft_living15  0.39249484  0.495298132  0.65036248 -0.07429273  0.19492162
## sqft_lot15    -0.03751371 -0.094897553 -0.10693008  0.63121861 -0.04956008
## price_cat      0.30709831  0.503675274  0.56072629 -0.12304671  0.29819632
```

```
##                 waterfront sqft_living15  sqft_lot15  price_cat
## bedrooms      -0.218684362    0.39249484 -0.03751371  0.3070983
## bathrooms     -0.005555848    0.49529813 -0.09489755  0.5036753
## sqft_living   -0.015193963    0.65036248 -0.10693008  0.5607263
## sqft_lot       0.015711737   -0.07429273  0.63121861 -0.1230467
## floors         0.067137319    0.19492162 -0.04956008  0.2981963
## waterfront     1.000000000   -0.03019461  0.01829592  0.1354341
## sqft_living15 -0.030194611    1.00000000 -0.13595666  0.5706052
## sqft_lot15     0.018295922   -0.13595666  1.00000000 -0.1842033
## price_cat      0.135434085    0.57060523 -0.18420334  1.0000000
```

In correlation with the variable price_cat, it can be seen that between:

- price_cat and bedrooms: there is an moderately-weak, positive correlation
- price_cat and bathrooms: there is a moderate, positive correlation
- price_cat and sqft_living: there is a moderate, positive correlation
- price_cat and sqft_lot: there is a weak, negative correlation
- price_cat and floors: there is a moderately-weak, positive correlation
- price_cat and waterfront: there is a weak, positive correlation
- price_cat and sqft_living15 : there is a moderately-strong, positive correlation
- price_cat and sqft_lot15: there is a weak, negative correlation

```
#Correlation results with graph:
library(corrplot)

## Warning: package 'corrplot' was built under R version 4.1.3

## corrplot 0.92 loaded

corrplot(cor(kc_second[,2:10])
         ,main = "Correlation results graph")
```


Correlation results graph
bedrooms
bathrooms
sqft_living
sqft_lot
floors
waterfront
sqft_living15
sqft_lot15
price_cat
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

Question 2 - Principal Component Analysis

a/ Calculate the mean and the variance for each appropriate variable and discuss if scaling is necessary and justify your findings.

```
head(kc_second)
```

```
##           id bedrooms bathrooms sqft_living sqft_lot floors waterfront
## 1 7922800400        5      3.25        3.25   14.342      2          0
## 2 1516000055        3      2.25        2.15   21.235      1          0
## 3 2123039032        1      0.75        0.76   10.079      1          1
## 4 9297300045        3      2.00        1.97    4.166      2          0
## 5 1860600135        5      2.50        3.65    9.050      2          0
## 6 1560930070        4      3.50        2.84   40.139      1          0
##   sqft_living15 sqft_lot15 price_cat
## 1          2.96     11.044         0
## 2          2.57     18.900         0
## 3          1.23     14.267         0
## 4          2.39      4.166         0
## 5          2.88      5.400         1
## 6          3.18     36.852         0
```

```
sapply(kc_second[,c(-1,-10)], mean)
```

```
##      bedrooms     bathrooms   sqft_living      sqft_lot        floors
##     3.6147059     2.5419118     2.9760853    24.7375941     1.4441176
##    waterfront sqft_living15    sqft_lot15
##     0.1882353     2.6945765    19.8577235
```

Mean calculation for appropriate variables, which include:

- Bedrooms, Bathrooms, Sqft_living, Sqft_lot, Floors, Waterfront, Sqft_living15, Sqft_lot.
- Variable such as: price_cat and id, are not appropriate to calculate mean.

Among the mean of each variable, there are 2 distinct variables with higher mean than other variables, they are sqft_lot (24.7375941) and sqft_lot15(19.8577235); they are roughly ten times higher than other variables' mean value.

```
#Calculating variance
sapply(kc_second[,c(-1,-10)], var)
```

```
##      bedrooms     bathrooms   sqft_living      sqft_lot        floors
##     1.0929984     0.8564609     1.5279319  3779.5763196     0.2638296
##    waterfront sqft_living15    sqft_lot15
##     0.1532535     0.5280590  1299.3700862
```

Variance calculation for appropriate variables, which include:

- Bedrooms, Bathrooms, Sqft_living, Sqft_lot, Floors, Waterfront, Sqft_living15, Sqft_lot.
- Variable such as: price_cat and id, are not appropriate to calculate variance.

Among the variance of each variable, there are also two special variable which have unusual higher values than other variables, which are sqft_lot (3779.573196) and sqft_lot15 (12.993700862). Meanwhile, the other variables' variance remains low.

From the dataset, we observe sqft_lot and sqft_lot15 are distinctive to other variables as their mean is higher and their variance is much more bigger than other variables' mean and variance.

-> Therefore, scaling is proper, for generalising and narrowing the data into a more observable range. The next command *prcomp* in part b) will have the dataset scaled and will provide the model with a calculated standard deviation.

b/ Perform a Principal Component Analysis and give the principal component loadings.

```
#b/
PCA_scaled <- prcomp(kc_second[,2:10], scale. = TRUE)

#component loadings of each variables
PCA_scaled$rotation
```

```
##                      PC1         PC2          PC3         PC4          PC5
## bedrooms      -0.3609277 -0.14452606 -0.391594523 -0.03679290  0.539347002
## bathrooms     -0.4597545 -0.10765695 -0.005347966 -0.14504717  0.274260864
## sqft_living   -0.4822084 -0.10561844 -0.016254405  0.06202895  0.110479156
## sqft_lot       0.1122122 -0.69106266  0.059816119  0.09235565 -0.140173592
## floors        -0.2661610 -0.04579729  0.266709057 -0.82906002 -0.301913123
## waterfront     0.0139074  0.02199295  0.836258713  0.15736301  0.480029512
## sqft_living15 -0.4083281 -0.04163799  0.000199204  0.42523117 -0.377314057
## sqft_lot15     0.1329041 -0.68803886  0.047162457 -0.02454745  0.008095502
## price_cat     -0.3972531  0.03297068  0.264756986  0.26777356 -0.371510711
##                      PC6         PC7         PC8           PC9
## bedrooms      -0.44805610  0.44058107 -0.09143695 -0.0046473562
## bathrooms      0.24144093 -0.52150954  0.01808664 -0.5917401297
## sqft_living    0.34496601 -0.21276632  0.01253046  0.7586661288
## sqft_lot      -0.06418836 -0.10866685 -0.67982010 -0.0008025312
## floors        -0.02266235  0.26810477 -0.06503876  0.0284845909
## waterfront     0.02599156  0.20155285 -0.05874334 -0.0072743038
## sqft_living15  0.38508118  0.53586864 -0.01118449 -0.2697761215
## sqft_lot15     0.01410204  0.07419814  0.70731654 -0.0067593521
## price_cat     -0.68421837 -0.26771968  0.14455857  0.0236209625
```

There are 8 principal components contributed by each following variables:

- Bedrooms, bathrooms, sqft_living, sqft_lot, floors, waterfront, sqft_living15, sqft_lot15
- Major contribution to each principal component is provided by:
  - PC1: bedrooms, bathrooms, sqft_living, sqft_living15, price_cat
  - PC2: sqft_lot, sqft_lot15
  - PC3: bedrooms and waterfront
  - PC4: floors and sqft_living15
  - PC5: bedrooms, floors, sqft_living15 and price_cat
  - PC6: bedrooms, sqft_living, sqft_living15 and price_cat
  - PC7: bedrooms, bathrooms and sqft_living15
  - PC8: sqft_lot and sqft_lot15
  - PC9: bathrooms and sqft_living

c/ Explain the proportion of variance explained by each principal component using a graph.

```
#c/
summary_PCA <- summary(PCA_scaled)
summary_PCA

## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6    PC7
## Standard deviation     1.8495 1.2616 1.0813 0.92292 0.78742 0.64538 0.6163
## Proportion of Variance 0.3801 0.1768 0.1299 0.09464 0.06889 0.04628 0.0422
## Cumulative Proportion  0.3801 0.5569 0.6868 0.78145 0.85034 0.89662 0.9388
##                           PC8     PC9
## Standard deviation     0.60116 0.43501
## Proportion of Variance 0.04015 0.02103
## Cumulative Proportion  0.97897 1.00000
```

From the summary above, the first principal component explains most of the variation in the dataset (PC1: 38.01%), while the last principal component explains the least variation (2.103%). Each principal component has explained about:

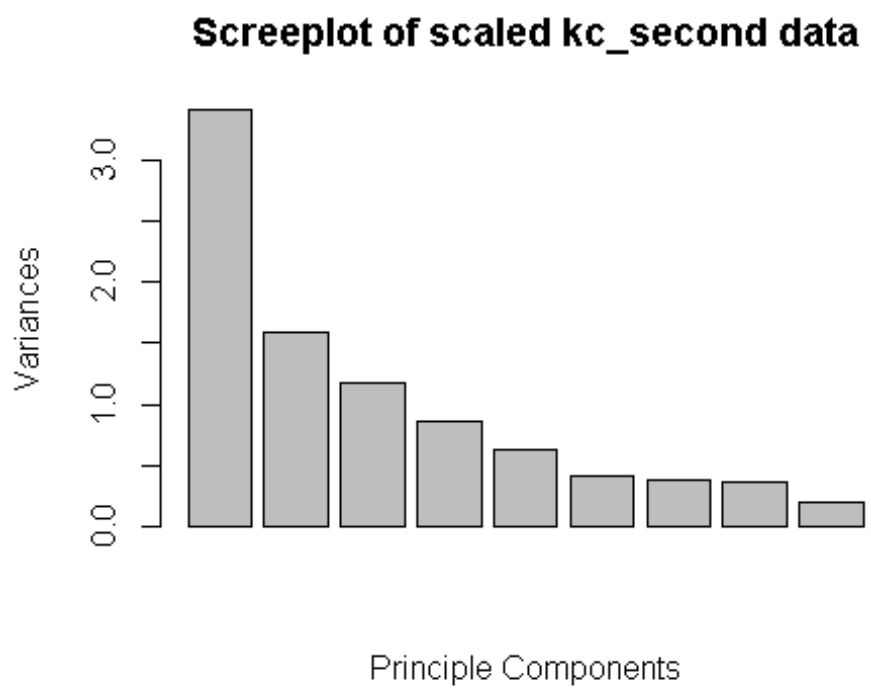
- PC1: 38.01% of the total variation
- PC2: 17.68% of the total variation
- PC3: 12.99% of the total variation

- PC4: 9.464% of the total variation
- PC5: 6.889% of the total variation
- PC6: 4.628% of the total variation
- PC7: 4.22% of the total variation
- PC8: 4.015% of the total variation
- PC9: 2.103% of the total variation

Cumulatively, the **first** and **second principal components** explain a total of 55.69% of the variation in the data. and so on, **until PC8,** where most of the variation (97.897%) in the data is explained.

Visualisation for the proportion of variance explained by each component.

```
screeplot(PCA_scaled,
          xlab = "Principle Components",
          main = "Screeplot of scaled kc_second data")
```



d/ Write the first two principal components in terms of the original variables in the given dataset.

First principal component in term of original variables:

PC1 = -0.3609277 x *bedrooms* -0.4597545 x *bathrooms*

-0.4822084 x *sqft_living* +0.1122122 x *sqft_lot*

-0.2661610 x *floors* +0.0139074 x *waterfront*

-0.4083281 x *sqft_living15* +0.1329041 x *sqft_lot15*

-0.3972531 x *price_cat*

Second principal component in term of original variables:

PC2 = -0.14452606 x *bedrooms* -0.10765695 x *bathrooms*

-0.10561844 x *sqft_living* -0.69106266 x *sqft_lot*

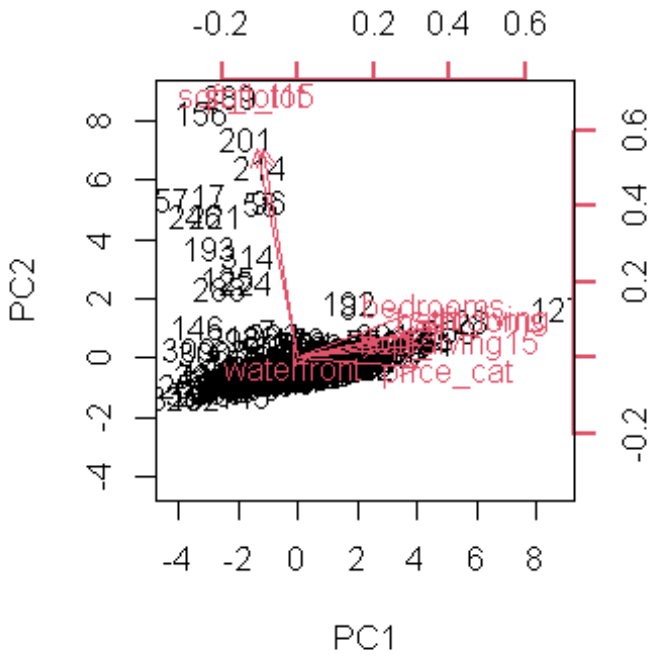
-0.04579729 x *floors* +0.02199295 x *waterfront*

- 0.04163799 x *sqft_living15* -0.68803886 x *sqft_lot15*

+0.03297068 x price_cat

e/ Construct the Biplot and interpret it.

```
#e/
par(mfrow = c(1,1))
PCA_scaled$rotation <- -PCA_scaled$rotation
PCA_scaled$x <- -PCA_scaled$x
biplot(PCA_scaled, scale = 0, col = c(1,2) + 8, lwd = 2)
```



From the biplot, it can be seen that the first component loading vector (PC1) gives an equal weight on **bedrooms, bathrooms, sqft_living, sqft_living15** and **price_cat**, which are all larger than both **waterfront, sqft_lot** and **sqft_lot15**. Therefore, this suggests a corresponding measure between the households' interior design and living space, with the overall price (**price_cat)** of a house within King County, Washington States, USA.

Meanwhile, the second loading vector (PC2) places most weight on **sqft_lot** and **sqft_lot15** than **waterfront** and **bedrooms, bathrooms, sqft_living, sqft_living15** and **price_cat**. This component corresponds to the measure of the overall exterior space of houses in King County, Washington State, USA.

Furthermore, the loading vector **waterfront** tends to separate itself from other loading vectors.Graphically, **Waterfront** vector lies at 0 on both components.

Insight:

- Houses with **greater interior design** (higher number of *bedrooms, bathrooms*) and **larger interior living space**, tend to possess **higher price**
  - Households on which lie on higher number of PC1, have **greater interior design** and **high price**. While houses on negative scale of PC1, have **less interior design** and **living space,** have **low price**.
- Houses with **great exterior living space** tends to have **lower price**
  - Households on which lie on higher number of PC2, have **greater exterior living space,** yet they have **lower price**

- While houses stay close to zero on both components have average level of exterior and interior living space

Question 3 - Clustering

a/ Cluster the 340 houses in King County in the given dataset into 2 groups using kmeans clustering.

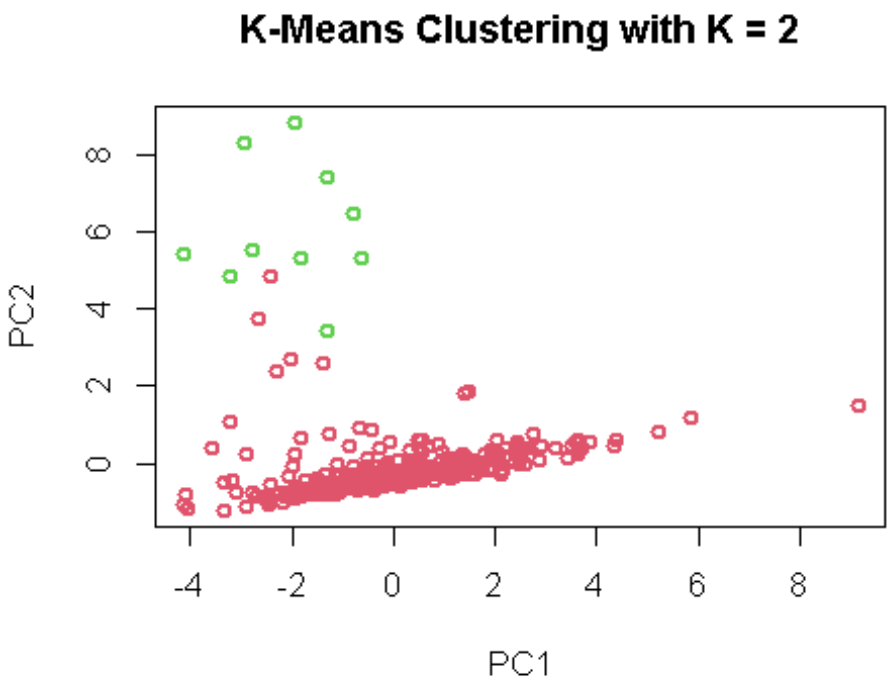
```
str(kc_second)
```

```
## 'data.frame':    340 obs. of  10 variables:
##  $ id           : num  7.92e+09 1.52e+09 2.12e+09 9.30e+09 1.86e+09 ...
##  $ bedrooms     : int  5 3 1 3 5 4 3 3 3 2 ...
##  $ bathrooms    : num  3.25 2.25 0.75 2 2.5 3.5 3 2.5 3 2.25 ...
##  $ sqft_living  : num  3.25 2.15 0.76 1.97 3.65 ...
##  $ sqft_lot     : num  14.34 21.23 10.08 4.17 9.05 ...
##  $ floors       : num  2 1 1 2 2 1 2 2 2 1.5 ...
##  $ waterfront   : int  0 0 1 0 0 0 1 1 0 0 ...
##  $ sqft_living15: num  2.96 2.57 1.23 2.39 2.88 3.18 2.28 3.98 3.08 2.13 .
..
##  $ sqft_lot15   : num  11.04 18.9 14.27 4.17 5.4 ...
##  $ price_cat    : num  0 0 0 0 1 0 0 1 1 0 ...
```

```
#Question 3 - Clustering
#a/
kc_third <- kc_second[,c(-1,-10)]


set.seed(4)
kc_third_km <- kmeans(kc_third, centers = 2)
pr_kc_third <- prcomp(kc_third, scale = TRUE)
```

b/ Visually display the clusters using the first two principal components (PCs)

```
#b/
par(mfrow=c(1,1))
plot(-pr_kc_third$x[,1:2],
     col = fitted(kc_third_km, "classes") + 1,
     lwd = 2,
     main = "K-Means Clustering with K = 2")
```



This graph represents the two clusters using the first two principal components (PC1 and PC2)

c/ Cluster the 340 houses in King County in the given dataset into 2 groups using hierarchical clustering. Consider Euclidean distance as the dissimilarity measure and the closest distance between two clusters as the maximum distance between them.

```
#c/
hh <- hclust(dist(kc_third), method = "complete")
hh

##
## Call:
## hclust(d = dist(kc_third), method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 340

cutree(hh, k = 2)# dividing dataset into 2 groups

##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1
```

```
1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [149] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1
1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1
1 1 1
## [260] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1
1 1 1
## [297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [334] 1 1 1 1 1 1 1
```

d/ Cluster the 340 houses in King County in the given dataset into 2 groups using hierarchical clustering. Consider Euclidean distance as the dissimilarity measure and the closest distance between two clusters as the average distance between them.

```
#d/
hh2 <- hclust(dist(kc_third), method = "average")
hh2

##
## Call:
## hclust(d = dist(kc_third), method = "average")
##
## Cluster method   : average
## Distance         : euclidean
## Number of objects: 340

cutree(hh2, k = 2)# dividing dataset into 2 groups

##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [149] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

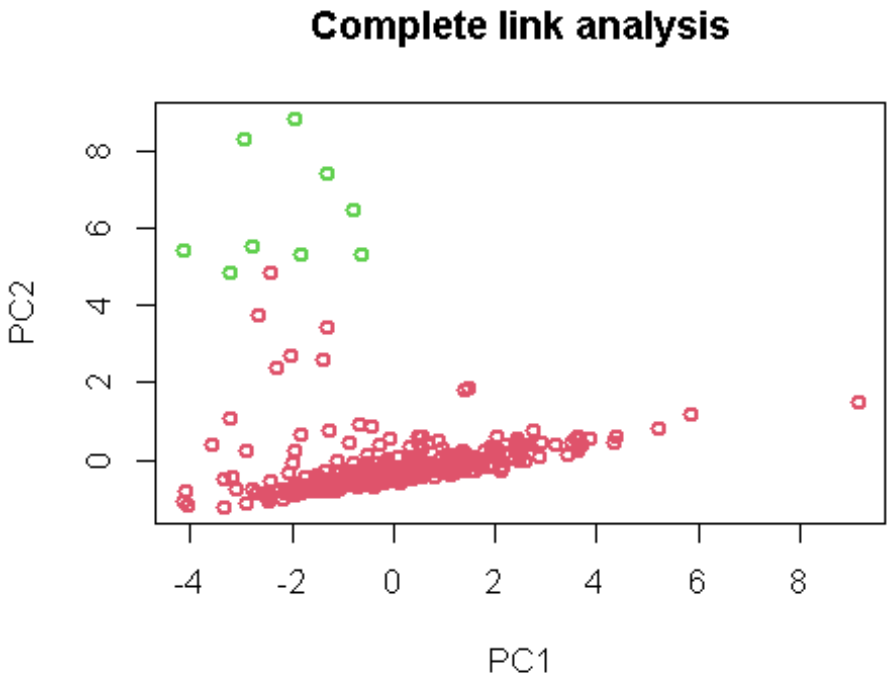
```
1 1 1
## [260] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
1 1 1
## [297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [334] 1 1 1 1 1 1 1
```

e/ Cluster the 340 houses in King County in the given dataset into 2 groups using hierarchical clustering. Consider Euclidean distance as the dissimilarity measure and the closest distance between two clusters as the minimum distance between them.

```
#e/
hh3 <- hclust(dist(kc_third), method = "single")
hh3

##
## Call:
## hclust(d = dist(kc_third), method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 340

cutree(hh3, k = 2) # dividing dataset into 2 groups

##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [149] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [260] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1
1 1 1
## [297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [334] 1 1 1 1 1 1 1
```

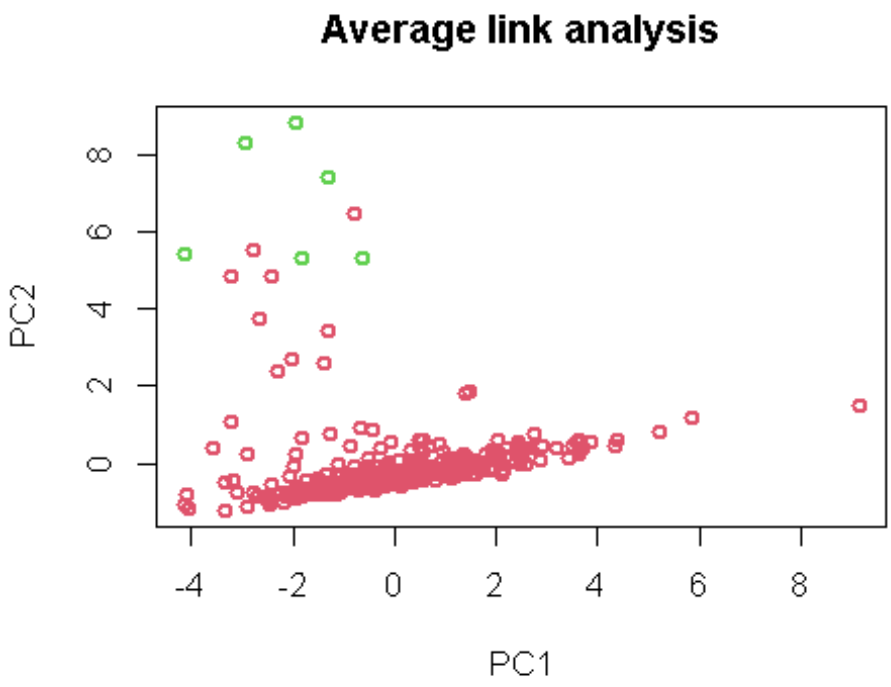
f/ Visually display the clusters obtained in part c , d and e using the first two principal components (PCs).

```
#f/
par(mfrow = c(1,1))
plot(-pr_kc_third$x[,1:2],
     col = cutree(hh, k = 2) + 1,
#k = 2 because we want two  cluster
     lwd = 2,
     main = "Complete link analysis")
```



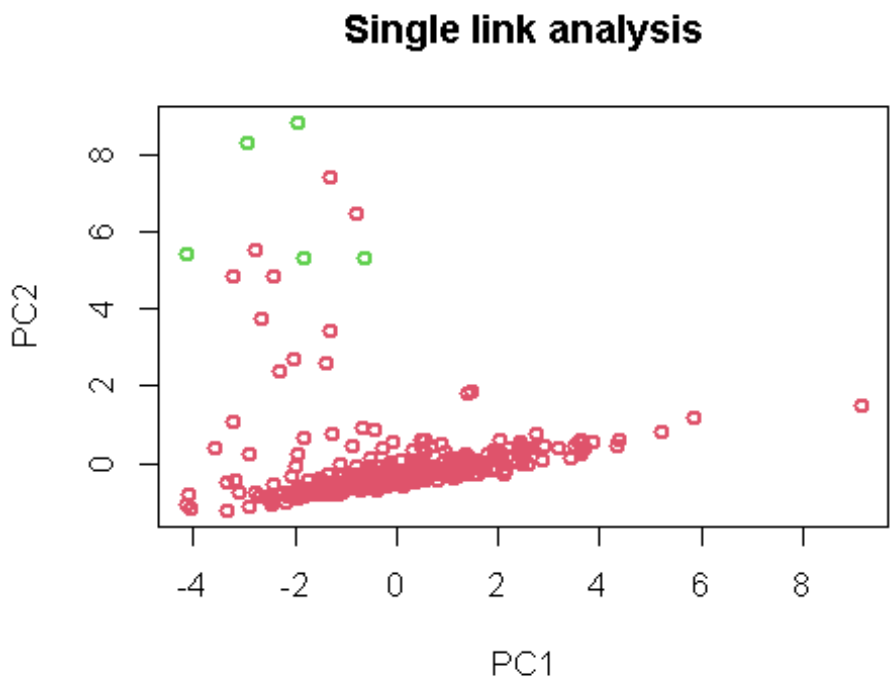
The **cluster graph** above, represents the **hierarchical cluster** data in **part c)** throughusing the first two principal components.

```
plot(-pr_kc_third$x[,1:2],
     col = cutree(hh2, k = 2) + 1,
#k = 2 because we want two  cluster
     lwd = 2,
    main = "Average link analysis")
```



The **cluster graph** above, represents the **hierarchical cluster** data in **part d)**throughusing the first two principal components.

```
plot(-pr_kc_third$x[,1:2],
     col = cutree(hh3, k = 2) + 1,
#k = 2 because we want two  cluster
     lwd = 2,
    main = "Single link analysis")
```



This is the **cluster graph** above, represents the **hierarchical cluster** data in **part e)**throughusing the first two principal components.

g/ Compare the results in part b, f and comment on your findings.

It can be observed from the graphs in part b) and part f) that:

- There are two groups of cluster (red and green).
- Each point belongs to a cluster and no observation is in more than one cluster.
  - In part f), the graph representing the hierarchical clustering data in part c) shows the two clusters (shown by red and green clusters), where the Euclidean distance between points in each cluster is maximised (or complete linkage).
  - In part f), the graph representing the hierarchical clustering data in part d) shows the two clusters (shown by red and green clusters), where the Euclidean distance between points in each cluster is average (or average linkage).
  - In part f), the graph representing the hierarchical clustering data in part e) shows the two clusters (shown by red and green clusters), where the

Euclidean distance between points in each cluster is minimised (or Single linkage).

- Meanwhile, the **k-means clustering results graph in part b)** shares a similarity in cluster grouping with **the graph representing hierarchical cluster data in part c)**. There is only one clustering point in the **hierarchical graph** that is different from from the **k-means clustering graph**, which is near 3 on the y-axis and -1 on the x-axis.
- Overall, the other 2 hierarchical graphs have different looks from the k-means clustering results graph, there are much more "red" cluster points than the "green" cluster points.
  - The red cluster in the hierarchical cluster graph with the average linkage, is dominant over the green cluster.
  - Meanwhile, the red cluster in the hierarchical cluster graph with single linkage, is over-dominant over the green cluster. There are not much green points presenting on the graph.

Further insights:

```
K1 <- cutree(hh, k = 2)
K2 <- cutree(hh2, k = 2)
K3 <- cutree(hh3, k = 2)
```

We can compare whether the cluster actually grouped the data correctly with the true price (price_cat) in the dataset kc:

```
#k- means clustering with k = 2
table(actual = kc$price_cat,cluster= kc_third_km$cluster)

##       cluster
## actual   1   2
##   high 169   1
##   low  161   9
```

- As it can be seen that using the hierarchical cluster with complete linkage, there are mixture of high and low observations in 1 and a few in cluster group 2.

```
#Hierarchical clustering with complete linkage
table(actual = kc$price_cat,cluster= K1)

##       cluster
## actual   1   2
```

```
##   high 169   1
##   low  162   8
```

- As same as for k-means clustering table, it can be seen that using the hierarchical cluster with complete linkage, there are mixture of high and low observations in 1 and a few in cluster 2. But there is one observation in cluster group 1, in variable "low", which belongs to cluter group 2.

```
#Hierarchical clustering with average linkage
table(actual = kc$price_cat,cluster= K2)

##       cluster
## actual   1   2
##   high 169   1
##   low  165   5
```

- As same as for k-means clustering table, it can be seen that using the hierarchical cluster with complete linkage, there are mixture of high and low observations in 1 and a few in cluster group 2.

```
#hierarchical cluster with single linkage
table(actual = kc$price_cat,cluster= K3)

##       cluster
## actual   1   2
##   high 169   1
##   low  166   4
```

- As same as for k-means clustering table, it can be seen that using the hierarchical cluster with complete linkage, there are mixture of high and low observations in 1 and a few in cluster group 2.

-> In assumption , clustering methods have not done well in classifying whether data has a low price or high price.(*Further insight:* This is because normally we don't have a response variable to start with in unsupervised learning. Therefore, using true **price_cat** variable to compare with the clustering data as how it is done above, is just an alternative approach to ensure whether clustering method is accurate to be used to predict the low and price of each house in kc_house dataset).

Question 4 - Support Vector Machines

a/ Divide the dataset into two sets namely training set and test set by assigning 75% of the observations to training set and the rest of the observations to the test set.

```
#Question 4 -Support Vector Machines
str(kc_second)
```

```
## 'data.frame':    340 obs. of  10 variables:
##  $ id           : num  7.92e+09 1.52e+09 2.12e+09 9.30e+09 1.86e+09 ...
##  $ bedrooms     : int  5 3 1 3 5 4 3 3 3 2 ...
##  $ bathrooms    : num  3.25 2.25 0.75 2 2.5 3.5 3 2.5 3 2.25 ...
##  $ sqft_living  : num  3.25 2.15 0.76 1.97 3.65 ...
##  $ sqft_lot     : num  14.34 21.23 10.08 4.17 9.05 ...
##  $ floors       : num  2 1 1 2 2 1 2 2 2 1.5 ...
##  $ waterfront   : int  0 0 1 0 0 0 1 1 0 0 ...
##  $ sqft_living15: num  2.96 2.57 1.23 2.39 2.88 3.18 2.28 3.98 3.08 2.13 .
..
##  $ sqft_lot15   : num  11.04 18.9 14.27 4.17 5.4 ...
##  $ price_cat    : num  0 0 0 0 1 0 0 1 1 0 ...
```

```
#a/
set.seed(1)
kc_second$price_cat <- ifelse(kc_second$price_cat == 1, "high", "low")
s <- sample(1:nrow(kc_third), 255) #340 * 75% = 255
train_set <- kc_second[s,] #75% of data
test_set <- kc_second[-s,] #the other 25% of data
```

Dataset is requested to be aggregated into 75% of the observations for training set and the other 25% of the observations is for test set

- The data set of kc_house2, which assigned under the value of kc_second, has 340 observations, which means 75% of 340 observations is 255, and the rest is 85 observations which bears 25% of the overall observations in the data set.

b/ Fit a support vector classifier, in order to classify whether a house has high or low price. [Hint: Clearly report the cross-validation errors associated with different values of this parameter].

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.1.3
```

```
svm1 <- svm(as.factor(price_cat) ~
            bedrooms + bathrooms + sqft_living + sqft_lot + floors
            + waterfront + sqft_living15 + sqft_lot15,
            data= train_set,
            kernel = "linear",
            cost = 1,
```

```
            scale = TRUE)
summary(svm1)

##
## Call:
## svm(formula = as.factor(price_cat) ~ bedrooms + bathrooms + sqft_living +
##     sqft_lot + floors + waterfront + sqft_living15 + sqft_lot15,
##     data = train_set, kernel = "linear", cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  107
##
##  ( 54 53 )
##
##
## Number of Classes:  2
##
## Levels:
##  high low
```

- The fitted model is supported with linear SVM-Kernel function along with the cost of 1.
- There are total of 107 support vectors (54 support vectors are from one class and 53 support vectors are from the other).
- And the number of class is 2 with 2 levels: High and Low, which used to predict price_cat.

```
set.seed(1)
tune1 <- tune(svm, as.factor(price_cat) ~
                   bedrooms + bathrooms + sqft_living + sqft_lot + floors
                   + waterfront + sqft_living15 + sqft_lot15,
                   data = train_set,
                   kernel = "linear",
                   ranges = list(cost = c(0.001,0.01,0.1,1,10,100,100)),
                   scale = TRUE)
summary(tune1)

##
## Parameter tuning of 'svm':
##
```

```
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1729231
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.3883077 0.10264407
## 2 1e-02 0.1884615 0.07166013
## 3 1e-01 0.1729231 0.06879521
## 4 1e+00 0.1803077 0.07676839
## 5 1e+01 0.1843077 0.06378300
## 6 1e+02 0.1804615 0.06179881
## 7 1e+02 0.1804615 0.06179881
```

```
bestmod1 <- tune1$best.model
summary(bestmod1)
```

```
##
## Call:
## best.tune(method = svm, train.x = as.factor(price_cat) ~ bedrooms +
##     bathrooms + sqft_living + sqft_lot + floors + waterfront + sqft_living
15 +
##     sqft_lot15, data = train_set, ranges = list(cost = c(0.001, 0.01,
##     0.1, 1, 10, 100, 100)), kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##
## Number of Support Vectors:  123
##
##  ( 62 61 )
##
##
## Number of Classes:  2
##
## Levels:
##  high low
```

Set.seed() function is needed to set a fixed tuning value, due to vary changes in dataset division and selection when tuning function without set.seed.

By assessing the above summary, the optimal model is achieved when the sampling method is 10-fold cross-validation and the cost is 0.1. Because when the cost is 1, the minimum validation error is gained (0.1729231).

-> Therefore, the fittest model, using **linear kernel**, is obtained when **the cost is 0.1**. And the fittest model consists of **123 support vectors** and 62 from one class and 61 from the other class

Thus, we are then using the fittest model obtained from the previous method, to predict test set and calculation the mis-classification rate:

```
#Misclassification rate of test set for model 1
test_pred1 <- predict(tune1$best.model, newdata= test_set)
misclass_table1 <- table(test_pred1, test_set[,"price_cat"])
mis_rate1 <- (misclass_table1[1,2] + misclass_table1[2,1])/sum(misclass_table
1)
mis_rate1
```

```
## [1] 0.1764706
```

c/ Fit a support vector machine with polynomial basis kernels, in order to classify whether a house has high or low price. [Hint: Clearly report the cross-validation errors associated with different values of this parameter.].

```
svm2 <- svm(as.factor(price_cat) ~
              bedrooms + bathrooms + sqft_living + sqft_lot + floors
            + waterfront + sqft_living15 + sqft_lot15,
            data= train_set,
            kernel = "polynomial",
            cost = 1,
            scale = TRUE)
summary(svm2)
```

```
##
## Call:
## svm(formula = as.factor(price_cat) ~ bedrooms + bathrooms + sqft_living +
##     sqft_lot + floors + waterfront + sqft_living15 + sqft_lot15,
##     data = train_set, kernel = "polynomial", cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
```

```
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  174
##
##  ( 88 86 )
##
##
## Number of Classes:  2
##
## Levels:
##  high low
```

- The fitted model is supported with polynomial kernel, with the cost of 1, the degree of 3 and coefficient of 0.
- There is a total of 174 support vectors (88 support vectors are from one class and 86 are from the other).
- And the number of class is 2 with 2 levels: High and Low, which used to predict price_cat.

```
set.seed(1)
tune2 <- tune(svm, as.factor(price_cat) ~
                bedrooms + bathrooms + sqft_living + sqft_lot + floors
              + waterfront + sqft_living15 + sqft_lot15,
              data = train_set,
              kernel = "polynomial",
              ranges = list(cost = c(0.001,0.01,0.1,1,10,100,100)),
              scale = TRUE)

summary(tune2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.2156923
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.5333846 0.05534210
```

```
## 2 1e-02 0.4706154 0.04825666
## 3 1e-01 0.2973846 0.13216659
## 4 1e+00 0.2236923 0.09562473
## 5 1e+01 0.2156923 0.06740342
## 6 1e+02 0.2273846 0.07255568
## 7 1e+02 0.2273846 0.07255568
```

```
bestmod2 <- tune2$best.model
summary(bestmod2)
```

```
##
## Call:
## best.tune(method = svm, train.x = as.factor(price_cat) ~ bedrooms +
##     bathrooms + sqft_living + sqft_lot + floors + waterfront + sqft_living
15 +
##     sqft_lot15, data = train_set, ranges = list(cost = c(0.001, 0.01,
##     0.1, 1, 10, 100, 100)), kernel = "polynomial", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  134
##
##  ( 66 68 )
##
##
## Number of Classes:  2
##
## Levels:
##  high low
```

Set.seed() function is needed to set a fixed tuning value for the model.

By assessing the above summary, the optimal model is achieved when the sampling method is 10-fold cross validation, the cost is 10 and the degree is 3, from which the lowest validation error is obtained (0.06740342).

-> Therefore, the best model, supported by **polynomial kernel,** is obtained when **the cost is 10** and having the **degree of 3** and **coefficient of 0**. Thus, the fittest model consists of **134 support vectors** and 66 from one class and 68 from the other class.

Then, we are using the fittest model obtained from the previous method, to predict test set and calculation the mis-classification rate:

```
#Misclassification rate of test set for model 2
test_pred2 <- predict(tune2$best.model, newdata= test_set)
misclass_table2 <- table(test_pred2, test_set[,"price_cat"])
mis_rate2 <- (misclass_table2[1,2] + misclass_table2[2,1])/sum(misclass_table
2)
mis_rate2
```

```
## [1] 0.2
```

d/ Fit a support vector machine with radial basis kernels, in order to classify whether a house has high or low price. [Hint: Clearly report the cross-validation errors associated with different values of this parameter.].

```
svm3 <- svm(as.factor(price_cat) ~
              bedrooms + bathrooms + sqft_living + sqft_lot + floors
            + waterfront + sqft_living15 + sqft_lot15,
            data= train_set,
            kernel = "radial",
            cost = 1,
            scale = TRUE)
summary(svm3)
```

```
##
## Call:
## svm(formula = as.factor(price_cat) ~ bedrooms + bathrooms + sqft_living +
##     sqft_lot + floors + waterfront + sqft_living15 + sqft_lot15,
##     data = train_set, kernel = "radial", cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  139
##
##  ( 68 71 )
##
##
## Number of Classes:  2
##
## Levels:
##  high low
```

- The fitted model is supported with radial kernel, with the cost of 1.

- There is a total of 139 support vectors (68 support vectors are from one class and 71 support vectors are from the other).
- And the number of class is 2 with 2 levels: High and Low.

```
set.seed(1)
tune3 <- tune(svm, as.factor(price_cat) ~
                bedrooms + bathrooms + sqft_living + sqft_lot + floors
              + waterfront + sqft_living15 + sqft_lot15,
              data = train_set,
              kernel = "radial",
              ranges = list(cost = c(0.001,0.01,0.1,1,10,100,100)),
              scale = TRUE)

summary(tune3)

##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1887692
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.5529231 0.06667929
## 2 1e-02 0.5529231 0.06667929
## 3 1e-01 0.2196923 0.07667241
## 4 1e+00 0.1887692 0.08216034
## 5 1e+01 0.2036923 0.07653990
## 6 1e+02 0.2696923 0.08837506
## 7 1e+02 0.2696923 0.08837506

bestmod3 <- tune3$best.model
summary(bestmod3)

##
## Call:
## best.tune(method = svm, train.x = as.factor(price_cat) ~ bedrooms +
##     bathrooms + sqft_living + sqft_lot + floors + waterfront + sqft_living
15 +
##     sqft_lot15, data = train_set, ranges = list(cost = c(0.001, 0.01,
##     0.1, 1, 10, 100, 100)), kernel = "radial", scale = TRUE)
```

```
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  139
##
##  ( 68 71 )
##
##
## Number of Classes:  2
##
## Levels:
##  high low
```

Same reason as above when creating a support vector machine model , Set.seed() function is needed to set a fixed tuning value for the model.

By assessing the above summary, the optimal model is achieved when the sampling method is 10-fold cross validation, with the cost of 1. So the corresponding minimum validation error is 0.1887692

-> Therefore, the best model when using **radial kernel** is obtained when **the cost is 1**, in which a **total number of support vectors is 139** (68 support vectors from one class and 71 support vectors from the other class).

Then, again we are using the fittest model obtained from the previous method, to predict test set and calculation the mis-classification rate:

```
#Misclassification rate of test set for model 3
test_pred3 <- predict(tune3$best.model, newdata= test_set)
misclass_table3 <- table(test_pred3, test_set[,"price_cat"])
mis_rate3 <- (misclass_table3[1,2] + misclass_table3[2,1])/sum(misclass_table
3)
mis_rate3
```

```
## [1] 0.1764706
```

e/ Comment on the prediction accuracy of the model in part b, c and d. Hence suggest the best model with clear justification.

In part b:

- The optimal model (using linear kernel) is achieved with the cost of 0.1, and with the minimum error rate of 0.1729231.
- There are 123 support vectors used: 62 support vectors are from one class and 61 support vectors are from the other.
- The misclassification rate when using on test_set is: 0.1764706.

In part c:

- The optimal model (using polynomial kernel) is achieved with the cost of 10, degree of 3, coefficient 0, and with the minimum error rate of 0.2156923.
- There are 134 support vectors used: 66 support vectors are from one class and 61 support vectors are from the other.
- The misclassification rate when using on test_set is: 0.2.

In part d:

- The optimal model (using radial kernel) is achieved with the cost of 1, and with the minimum error rate of 0.1887692.
- There are 139 support vectors used: 68 support vectors are from one class and 71 support vectors are from the other.
- The misclassification rate when using on test_set is: 0.1764706

In comparison, it can be seen that the optimal model in part b) has the same misclassification rate as the optimal model in part d) (= 0.1764706), when testing on the test_set data by using the training model created in each part.

However, During the validation of training error, using 10-fold cross validation, the model part b) has smaller validation error than the optimal model part d):

- Validation error of the optimal model in part b): 0.1729231
- Validation error of the optimal model in part d): 0.1887692

In conclusion, it is suggested that best using the **model in part b)**, where **linear kernel** is used and has the **cost of 0.1,** to predict new data in the future.

- The End -