DSA1101 Topic 4: K-Nearest Neighbours

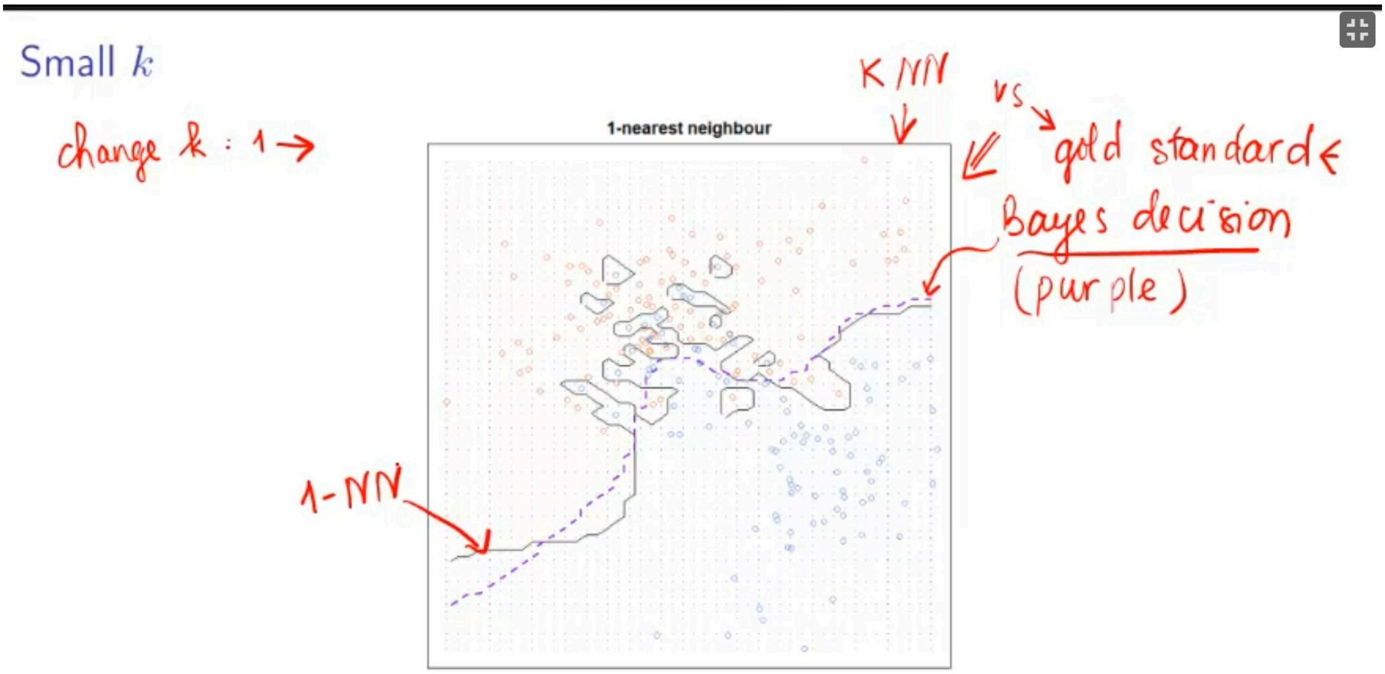
Dawn Cheung

2024-03-04

Definition of Terms

- has n number of training points
- features denoted as x
- categorical response is y
- with info x, the predicted y is Ĝ(x) [G hat (x)] or really y hat is fine too
- since we only consider binary responses (ie 0 or 1) only, the prediction Ĝ(x) is either 0 or 1 too

KNN



Blue $= 0$, orange $= 1$. The k-nearest neighbors classification using $k = 1$.

Scaling x

```
market <- read.csv("~/GitHub/DSA1101 Slayers/datasets/Smarket.csv")
attach(market)
Lag1 = scale(Lag1) # to standardise all the predictors (the x es)
```

To be done BEFORE APPLYING KNN

Will not be needed for stock market dataset cus lap1 to lap 5 are all *similar in magnitude* => will not change the outcome a lot

Stock Market Dataset

Predicting the direction of the stock market: whether it goes up or down => 1 is up, 0 is down

knn() will require the following arguments:

- Matrix of predictors/features x for training
- Matrix of predictors/features x to be predicted
- Vector containing class labels for the training data
- Value for k (number of nearest neighbours for the classifier)

```
# Enter code here
library(class)
```

```
## Warning: package 'class' was built under R version 4.3.2
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
market <- read.csv("~/GitHub/DSA1101 Slayers/datasets/Smarket.csv")
head(market)
```

```
##   X Year   Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today Direction
## 1 1 2001  0.381 -0.192 -2.624 -1.055  5.010 1.1913  0.959        Up
## 2 2 2001  0.959  0.381 -0.192 -2.624 -1.055 1.2965  1.032        Up
## 3 3 2001  1.032  0.959  0.381 -0.192 -2.624 1.4112 -0.623      Down
## 4 4 2001 -0.623  1.032  0.959  0.381 -0.192 1.2760  0.614        Up
## 5 5 2001  0.614 -0.623  1.032  0.959  0.381 1.2057  0.213        Up
## 6 6 2001  0.213  0.614 -0.623  1.032  0.959 1.3491  1.392        Up
```

```
# we see that X is the row number,
# Lag 1 to 5, the percentage returns for the 5 prev days => these are our predictors
dim(market) #get no. rows and columns
```

```
## [1] 1250   10
```

```
attach(market)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     Lag1
```

```
## The following objects are masked from market (pos = 5):
##
##     Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, X, Year
```

Gameplan:

- all years before 2005 (ie 2001 - 2004) will be training set
- year 2005 will be testing set

```
#indexes of all the rows where Year < 2005
index.train = which(Year < 2005) #returns a vector

#create data frame that has all the rows before 2005
train.data = market[index.train, ] #REMEMBER THE ,] BC MARKET IS NOT A LIST, ITS A DATAFRAME
so u need specify the rows and columns
# leave blank ie [index.train,(empty)] = returns all columns

#the rest of the rows go to test data
test.data = market[-index.train, ]

dim(train.data); dim(test.data) #dim is dimentions hehe
```

```
## [1] 998  10
```

```
## [1] 252  10
```

Getting arguments 1 and 2

```
#now we're filtering by all the columns we're gonna use, for both test and train datasets. id
k why we're doing this after spiltting them, OK NOW I GETS cus we're gonna split them further
into predictors and responses hehe
#REMEMBER THESE ARE DATAFRAMES SO SPECIFY ROW AND COLUMN
train.x = train.data[ ,c("Lag1", "Lag2", "Lag3", "Lag4", "Lag5")]
test.x = test.data[ ,c("Lag1", "Lag2", "Lag3", "Lag4", "Lag5")]
```

Getting argument 3

```
#now get the responses to train & test the algorithm
train.y = train.data[ ,c("Direction")]
test.y = test.data[ ,c("Direction")] #these are the 'real' responses, NOTE it is NOT needed f
or knn()
```

OK WE FORMING THE MODEL NOW

```
library(class)

knn.pred = knn(train.x, test.x, train.y, k = 1)

knn.pred #returns the prediction for the response of the test points i.e. predictions for tex
t.x
```

```
##   [1] Down Up   Up   Down Down Down Up   Down Up   Down Up   Down Up   Down Down
##  [16] Up   Down Down Up   Down Up   Down Down Down Down Up   Up   Up   Up   Down
##  [31] Up   Down Down Up   Up   Down Up   Up   Up   Down Up   Up   Up   Up   Up
##  [46] Down Up   Down Up   Up   Up   Up   Down Down Down Up   Down Up   Down Up
##  [61] Down Up   Down Up   Down Up   Down Down Down Down Down Up   Down Up   Up
##  [76] Down Down Down Up   Down Up   Down Up   Down Up   Down Up   Up   Down Up
##  [91] Down Down Up   Up   Down Up   Down Down Down Up   Down Down Up   Up   Up
## [106] Up   Up   Down Down Down Up   Up   Up   Up   Up   Down Up   Down Up   Down
## [121] Up   Down Down Up   Up   Down Up   Down Up   Up   Up   Down Down Down Up
## [136] Up   Up   Up   Down Up   Down Up   Up   Up   Up   Down Down Down Up   Down
## [151] Up   Up   Down Down Up   Down Up   Down Up   Up   Up   Down Up   Up   Up
## [166] Up   Up   Up   Up   Down Up   Down Down Up   Up   Up   Up   Down Up   Down
## [181] Up   Down Down Down Down Down Down Up   Up   Up   Down Up   Down Down Down
## [196] Up   Up   Up   Down Down Up   Down Down Down Up   Down Down Up   Up   Down
## [211] Down Down Down Up   Down Down Up   Up   Down Down Down Up   Down Up   Up
## [226] Down Down Down Down Down Up   Up   Down Up   Down Down Up   Up   Up   Up
## [241] Down Up   Up   Up   Down Up   Up   Down Down Up   Up   Up
## Levels: Down Up
```

So how did the model do?

```
data.frame(test.y, knn.pred) %>%
  slice(1:25) #lol to shorten the doc
```

```
##    test.y knn.pred
## 1    Down     Down
## 2    Down       Up
## 3    Down       Up
## 4      Up     Down
## 5    Down     Down
## 6      Up     Down
## 7    Down       Up
## 8      Up     Down
## 9    Down       Up
## 10     Up     Down
## 11     Up       Up
## 12   Down     Down
## 13   Down       Up
## 14   Down     Down
## 15   Down     Down
## 16     Up       Up
## 17     Up     Down
## 18     Up     Down
## 19   Down       Up
## 20     Up     Down
## 21     Up       Up
## 22     Up     Down
## 23   Down     Down
## 24     Up     Down
## 25   Down     Down
```

```
table(test.y, knn.pred)
```

```
##        knn.pred
## test.y Down Up
##   Down   55 56
##   Up     66 75
```

Diagnostics

Evaluation of the classifier's performance

More notations:

- For 2 class labels, C is positive and C' (C prime) is negative
  - True Positive: predict C, when actually C
  - True Negative: predict C', when actually C'
  - False Positive: predict C, when actually C'
  - False Negative: predict C', when actually C

Confusion Matrix

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

Example: Classifying Spam Emails

		Predicted Class		
		Spam	**Non-Spam**	Total
Actual Class	**Spam**	TP 3	8 FN	11
	Non-Spam	2 FP	87 TN	89
Total		5	95	100

Spam = +

non- = −

- A testing set has 100 emails (with their spam or non-spam label known).

11 89

Criteria to Evaluate

- Accuracy
- True Positive Rate (TPR)
- False Positive Rate (FPR)– Type 1 Error
- False Negative Rate (FPR)– Type 2 Error
- Precision
- ROC curve & AUC value (will learn later)

Accuracy

- It is defined as the sum of TP and TN divided by the total number of instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

- The *overall success rate*
- Basically correct / total x 100%

True Positive Rate (TPR)

- The true positive rate (TPR) shows the proportion of positive instances the classifier correctly identified:

$$\text{TPR} = \frac{TP}{TP + FN}$$

(high)

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

False Positive Rate (FPR)– Type 1 Error

- The FPR is also called the false alarm rate or Type I error rate

$$\text{FPR} = \frac{FP}{FP + TN}$$

False Negative Rate (FNR)– Type 2 Error

- It is also known as the miss rate or Type II error rate.

$$\text{FNR} = \frac{FN}{TP + FN}$$

Precision

- **Precision** is the percentage of instances that are actually positive among the marked positives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

Odds ratio for 2 by 2 matrix:

[a b]

[c d]

OR = ad/bc

```
slay <- function(x) {
  if (any(x == 0)){
    x = x + 0.5
  }
  return( (x[1,1]*x[2,2]) / (x[1,2]*x[2,1]) )
}

slay(cbind(c(1,2), c(3,4)))
```

```
## [1] 0.6666667
```

```
#remember that cbind stands for column bind
#so all inputs will be put as new columns
```

k in knn is usually odd

Accuracy

```
knn.pred = knn(train.x, test.x, train.y, k = 5)
table(test.y, knn.pred)
```

```
##        knn.pred
## test.y Down Up
##   Down   50 61
##   Up     57 84
```

```
knn.pred = knn(train.x, test.x, train.y, k = 10)
table(test.y, knn.pred)
```

```
##       knn.pred
## test.y Down Up
##   Down   52 59
##   Up     52 89
```

TUTORIAL QUESTIONS

1 Read the data from the file Colleges.txt. Consider a simple linear regression of percentage of applicants accepted (Acceptance) on the median combined math and verbal SAT score of students (SAT), called Model M1.

1a) Consider data set Colleges.txt. **Write a function in R using the matrix approach to perform a simple linear regression of percentage of applicants accepted (Acceptance) on the median combined math and verbal SAT score of students (SAT).**

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ readr     2.1.4
## ✔ ggplot2   3.4.4     ✔ stringr   1.5.0
## ✔ lubridate 1.9.2     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
come errors
```

```
library(dplyr)
collegesdb <- read.csv("~/Github/DSA1101 Slayers/datasets/Colleges.txt", sep = "\t", header =
TRUE)
attach(collegesdb)


matrix <- function(x, y){
  #rem x and y are matrixes
  #aim: get beta hat (by minimising RSS, etc)
  beta <- solve(t(x)%*%x)%*%t(x)%*%y
  return(beta)
  #solve() gets the inverse of the function
  #t() is transpose which is A^T the T thing
  #%*% multiplies the matrixes, like multiplication sign for matrixes
}
matrix(x = cbind(1, SAT), y = Acceptance)
```

```
##            [,1]
##     202.2677440
## SAT  -0.1300894
```

```
#bcos x is a 2 by n matrix, and y is a 1 by n
```

Compare the results with the answers in part (b) of Question 1.

```
#simple(SAT, Acceptance)
#simple() is NOT A BUILT IN FUNCTION is from tut 3

lm(Acceptance ~ SAT)
```

```
##
## Call:
## lm(formula = Acceptance ~ SAT)
##
## Coefficients:
## (Intercept)          SAT
##    202.2677      -0.1301
```

1b) If data set of n points has two input features, x1, x2, by matrix approach, the estimate of coefficient is still ˆβ = (XT X)−1XT y

i. Specify **matrix y, X and β.**
ii. Use your function in part (a) to **perform a multivariate linear regression of percentage of applicants accepted (Acceptance) on SAT and Top.10p - percentage of students in the top 10% of their high school graduating class**

for i,

y = response variable, Acceptance

X = matrix of columns 1, SAT and Top.10p

β = da unknowns

```
# Define matrix X
X = cbind(1, SAT, Top.10p)
matrix(x = X, y = Acceptance)
```

```
##                 [,1]
##         175.54421649
## SAT      -0.08478261
## Top.10p  -0.41029538
```

Compare the results with using lm()

```
lm(Acceptance ~ SAT + Top.10p)
```

```
##
## Call:
## lm(formula = Acceptance ~ SAT + Top.10p)
##
## Coefficients:
## (Intercept)          SAT      Top.10p
##   175.54422     -0.08478     -0.41030
```

2. A dataset on house selling price was randomly collected 1, house_selling_prices_FL.csv. It's our interest to model how y = selling price (dollar) is dependent on x = the size of the house (square feet). A simple linear regression model (y regress on x) was fitted, called Model 1. The given data has another variable, NW, which specifies if a house is in the part of the town considered less desirable (NW = 0).

2a) Derive the correlation between x and y.

```r
#enter code here
library(dplyr)
library(class)
library(tidyverse)
housey = read.csv("~/GitHub/DSA1101 Slayers/datasets/house_selling_prices_FL.csv")
attach(housey)
NW = as.factor(NW) #declaring NW as categorical, just to clean the data before starting


cor(price, size)
```

```
## [1] 0.7612621
```

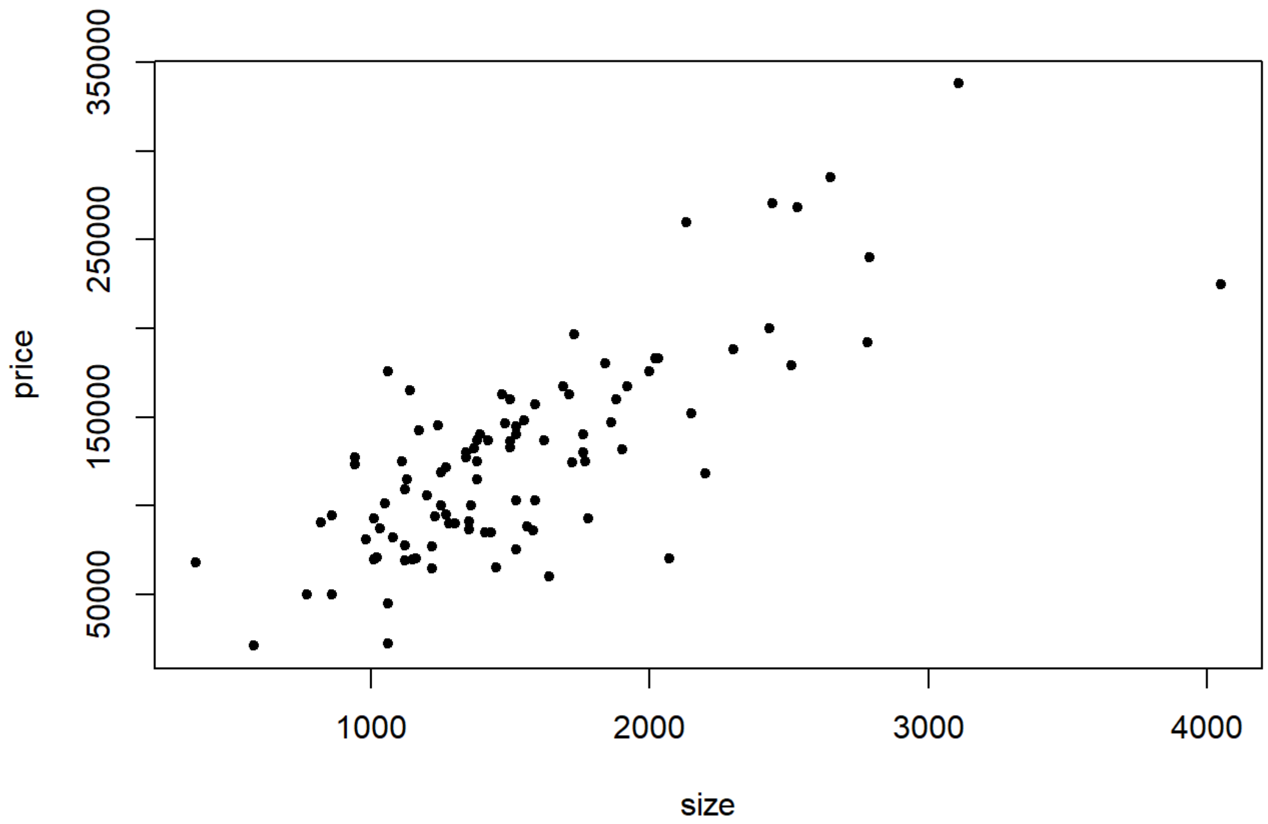
```r
cor(size, price)
```

```
## [1] 0.7612621
```

2b)Derive a scatter plot of y against x. Give your comments on the association of y and x.

```r
# Enter code here
M1 = lm(price ~ size)

plot(size, price, pch = 20)
```



Positive relationship

- Linear correlation: no curving of points
- The variability of response is quite stable: range of y does not increase as x increases

2c) Derive R2 of Model 1. Verify that √R2 = |cor(y,x)|. In which situation we can have √R2 = cor(y, x)

```
M1 = lm(price ~ size)
summary(M1) #R^2 is hidden somewhere here
```

```
##
## Call:
## lm(formula = price ~ size)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -98567 -23582   2404  18843  89345
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9161.159  10759.786   0.851    0.397
## size           77.008      6.626  11.622   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36730 on 98 degrees of freedom
## Multiple R-squared:  0.5795, Adjusted R-squared:  0.5752
## F-statistic: 135.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
summary(M1)$r.squared #SPELL SQUARED CORRECTLY PLS
```

```
## [1] 0.57952
```

```
(cor(price, size))^2 # its cor to corr hor
```

```
## [1] 0.57952
```

```
#they will be very similar when:
#if cor(y, x) > 0, and M1 is simple y ~ x (simple model)
#why: prove cor(y, yhat) = cor(y, x)
# prove cor^2(y, yhat) = var(yhat)/var(y)
# claim var(yhat)/var(y) = R^2

#where the explainatory is quantitative
```

2d) Form a model (called Model 2) which has two regressors (x and NW). Write down the equation of Model 2.

```
# Enter code here
NW = as.factor(NW) #declaring NW as categorical

M2 = lm(price ~ size + NW)
summary(M2)
```

```
##
## Call:
## lm(formula = price ~ size + NW)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -83207 -22968    215  14135 109149
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -15257.514  11908.297  -1.281 0.203160
## size            77.985      6.209  12.560  < 2e-16 ***
## NW1          30569.087   7948.742   3.846 0.000215 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34390 on 97 degrees of freedom
## Multiple R-squared:  0.6352, Adjusted R-squared:  0.6276
## F-statistic: 84.43 on 2 and 97 DF,  p-value: < 2.2e-16
```

```
#notice that since NM is catagorical, it starts calling the variable NW1 to show that it is a
n indicator, and it is indicating that NW = 1 == True => I(NW = 1)is the indicator lol
M2$coeff
```

```
##  (Intercept)         size          NW1
## -15257.51385     77.98513  30569.08729
```

```
paste0("the equation is: price(hat) = ", M2$coeff[1], " + ", M2$coeff[2], "size + ",M2$coeff
[3], "I(NW = 0)")
```

```
## [1] "the equation is: price(hat) = -15257.5138544573 + 77.9851263761225size + 30569.087288
7755I(NW = 0)"
```

2e) Report the coefficient of variable NW in Model 2. Interpret it.

```
# Enter code here
M2$coeff[3]
```

```
##      NW1
## 30569.09
```

2 houses of the same location, then one with size increase by 1 unit will have average price larger by $77.98 [must be same location]

for houses of the same size, the one not at the NW area (or NW = 1) will be more expensive by $3059 on average

2f) Estimate the price of a house where its size is 4000 square feet and is located at the more desirable part of the town

```
# Enter code here

new_data = data.frame(size = 4000, NW = "1") #creates 2 columns
#also NW = "1" must be in quotes cus its a factor vairable (not a number)

predict(M2, newdata = new_data)
```

```
##        1
## 327252.1
```

2g) Report the R2 of Model 2. Interpret it

```
# Enter code here
summary(M2)$r.squared
```

```
## [1] 0.6351502
```

Relatively low (quite far from 1) => low amount of variability inherent in the response before the regression is performed => M2 BETTER than model 1, which has a higher R^2 (so M1 has higher variability than M2)

Extension question: is M2 significant?

Significance => look at F statistic's p-value

since p-value < 2.2e-14 compared to a model with no regressors, just an intercept (straight line), M2 does significantly better