# Exam 2 Review

Peter Kabai

# Review Topics

- Data Exploration
- Data Cleaning and Preprocessing
- KNN
- Linear Regression
- Logistic Regression
- R Code

# Introduction

This review session was originally presented as a Jeopardy-ish style game, with questions worth a different number of points and teams of 3 answering them. Correct answers were followed by a more in depth explanation, and when there were incorrect answers other teams were allowed to correct it. Each of the six review topics on the previous slide has 7 questions associated with it, for a total of 42 game questions.

# Data Exploration

# Data Exploration - Q1:

Write R code to import a CSV file and store it as the variable `dat`.

# Data Exploration - A1:

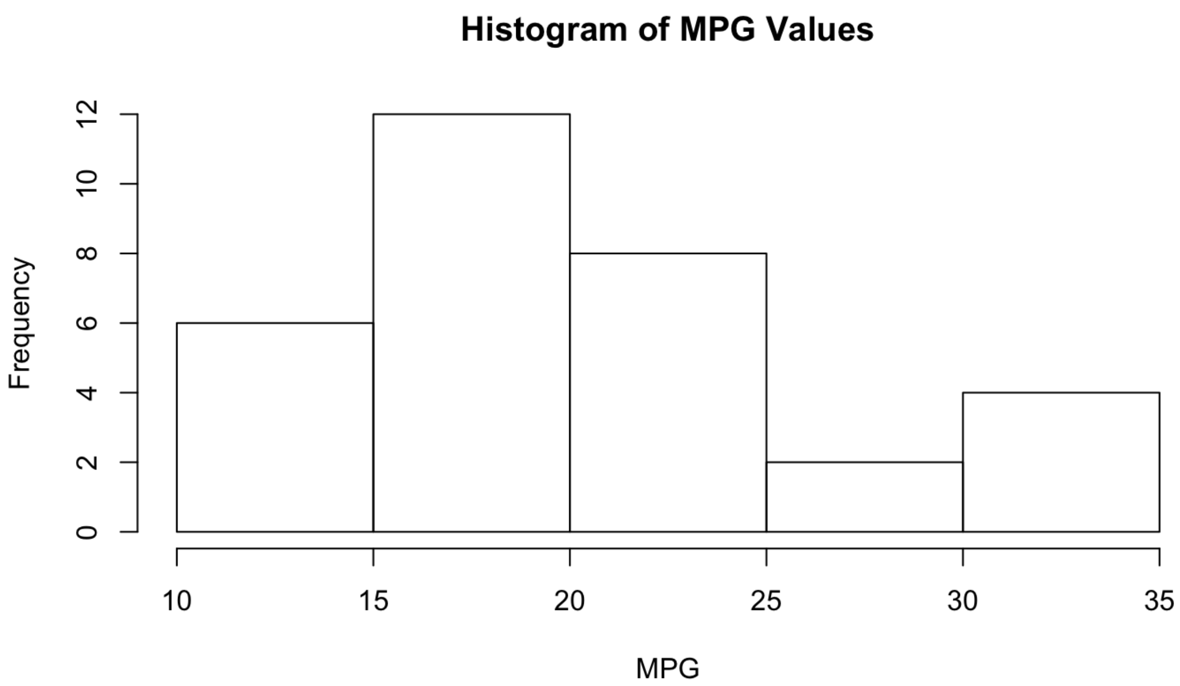
```
dat=read.csv("../../data/titanic.csv")
```

# Data Exploration - Q2:

Create a histgram of `mpg` values using the `mtcars` data. Add a title, and a clean x axis label.

# Data Exploration - A2:

```
hist(mtcars$mpg, main="Histogram of MPG Values", xlab="MPG")
```

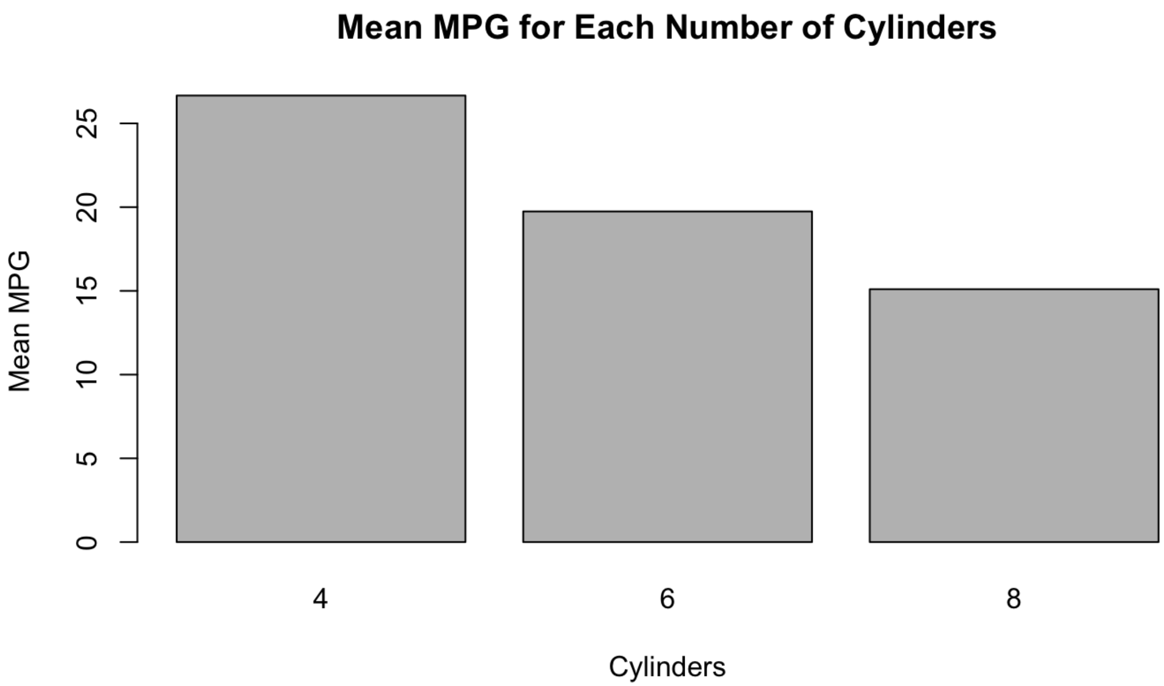


# Data Exploration - Q3:

Create a bar plot of mean `mpg` grouped `cyl` number, for the `mtcars` data.

# Data Exploration - A3:

```
data=aggregate(mpg ~ cyl, data=mtcars, mean)
barplot(data$mpg, names.arg=data$cyl, main="Mean MPG for Each Number of Cylinders",
        xlab="Cylinders", ylab="Mean MPG")
```

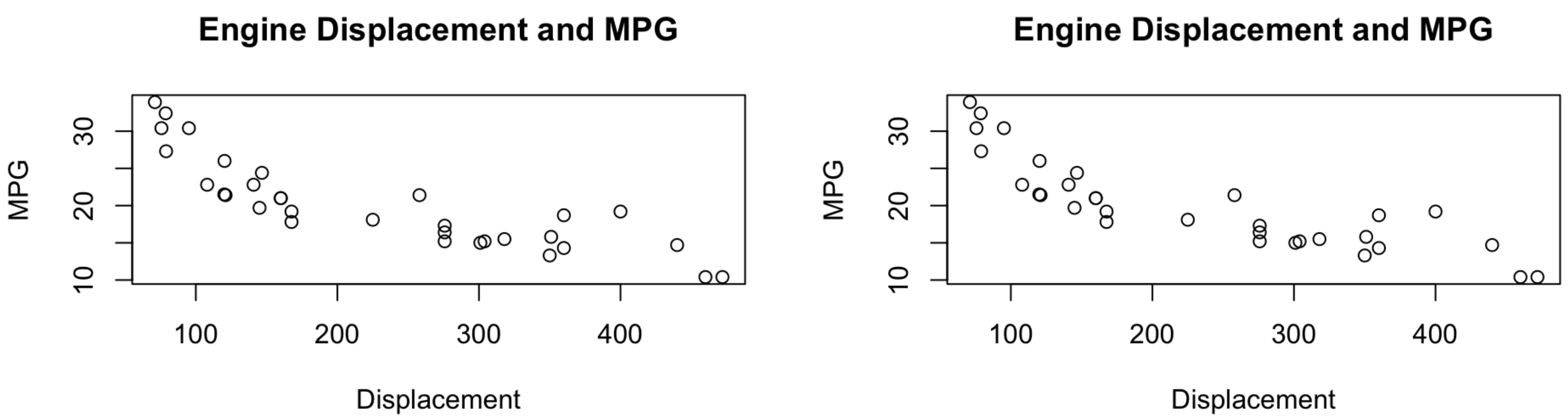


# Data Exploration - Q4:

Create a scatterplot with the `disp` on the x axis, and the `mpg` on the y axis using the `mtcars` data. There are of course, multiple ways to do this, so do it in 2 different ways. Arrange these identical plots side by side.

# Data Exploration - A4:

```
par(mfrow=c(1,2))
plot(mtcars$disp, mtcars$mpg, main="Engine Displacement and MPG",
     xlab="Displacement", ylab="MPG")
plot(mpg ~ disp, data=mtcars, main="Engine Displacement and MPG",
     xlab="Displacement", ylab="MPG")
```

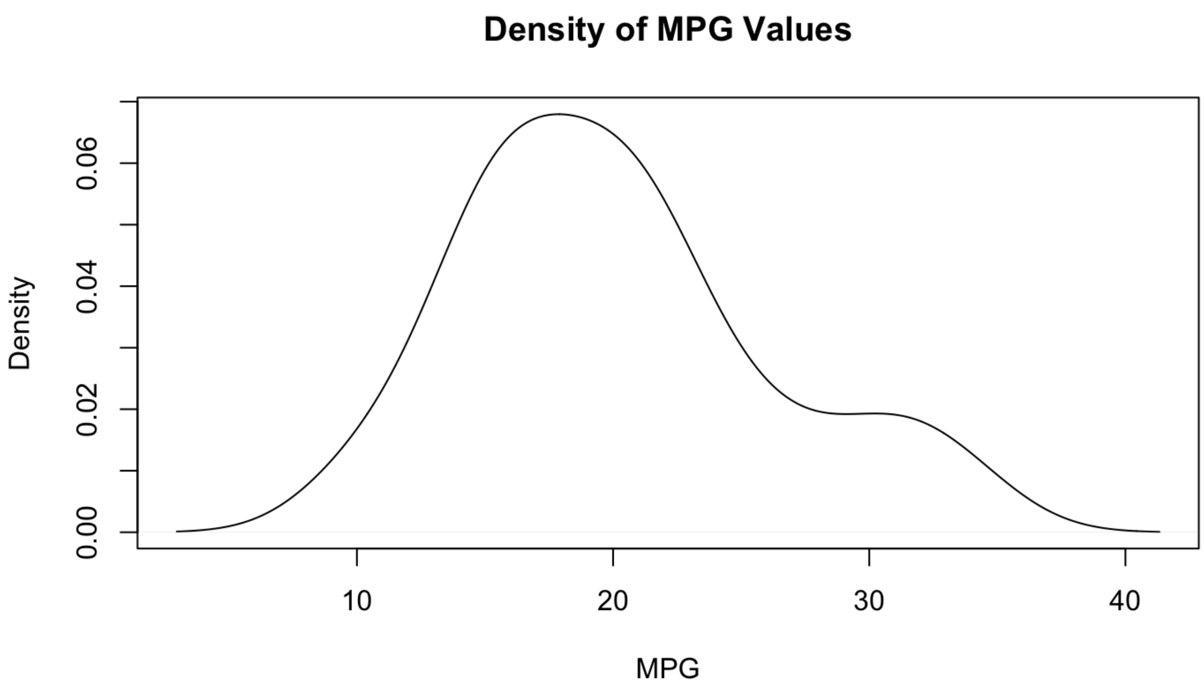


# Data Exploration - Q5:

Create a density plot of the `mpg` data from `mtcars`. Add a title and x label.

# Data Exploration - A5:

```
plot(density(mtcars$mpg), main="Density of MPG Values", xlab="MPG")
```

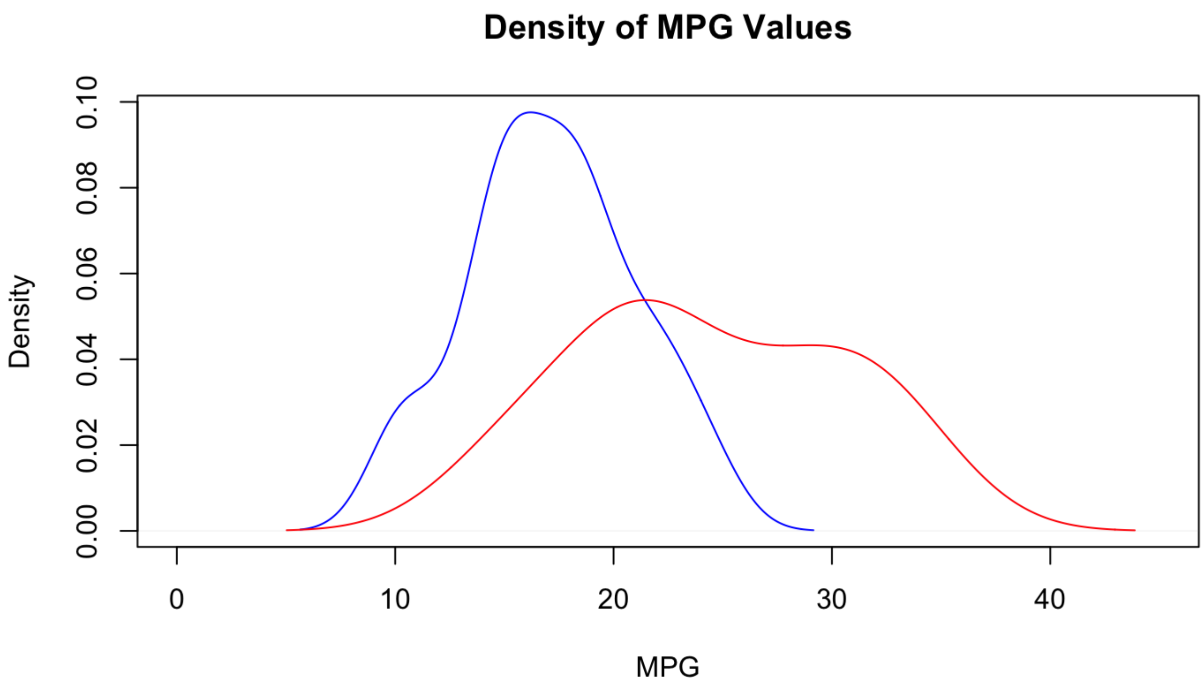


# Data Exploration - Q6:

Create a double density plot, a blue line for the `mpg` of automatic cars, where `am=0`, and a red line for manual cars, where `am=1`.

# Data Exploration - A6:

```
plot(density(mtcars[mtcars$am == 0,]$mpg), main="Density of MPG Values",
     xlab="MPG", col="blue", xlim=c(0, 45))
lines(density(mtcars[mtcars$am == 1,]$mpg), col="red")
```

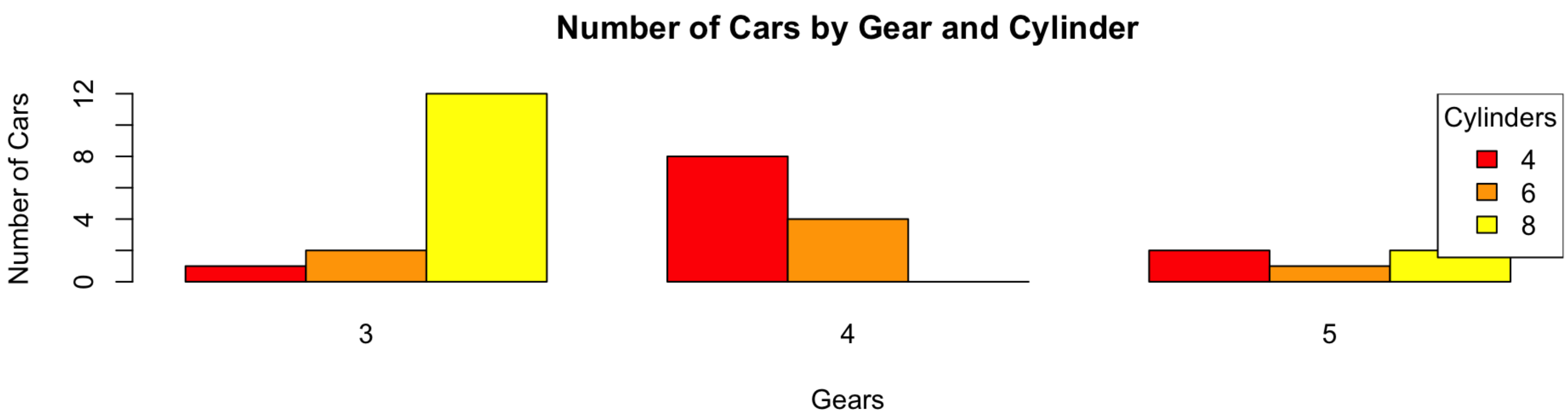


# Data Exploration - Q7:

Create a side by side bar plot using any of the columns in `mtcars`.

# Data Exploration - A7:

```
colors=c("red","orange","yellow")
barplot(table(mtcars$cyl, mtcars$gear), beside=TRUE, ylab="Number of Cars",
        xlab="Gears", col=colors, main="Number of Cars by Gear and Cylinder")
legend("topright", title="Cylinders", legend=sort(unique(mtcars$cyl)), fill=colors)
```



# Data Cleaning and Preprocessing

# Data Cleaning - Q1:

Write R code to count the total number of `NA` values in a data frame, and code to count the `NA` values in each column of the data frame.

# Data Cleaning - A1:

```
dat=read.csv("../../data/titanic.csv")
```

```
sum(is.na(dat))
```

```
## [1] 177
```

```
apply(dat, 2, function(x) sum(is.na(x)))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age       SibSp        Pa
##           0           0           0           0           0         177           0
```

# Data Cleaning - Q2:

Give an example of when you would remove entire columns from your dataset, and when you would remove entire rows from your dataset. Would you ever do both?

# Data Cleaning - A2:

If a column has a high percentage of `NA` values you may want to remove the column, rather than all the rows. This is because if you have a problematic column and remove all rows effected by it, you may be losing a lot of values that are not `NA`, but if you remove the majority `NA` column then you're mostly just removing `NA` values.

# Data Cleaning - Q3:

Why does this R code return NA rather than the mean age? Re-write the code to avoid the issue.

```
mean(read.csv("../../data/titanic.csv")$Age)
```

```
## [1] NA
```

# Data Cleaning - A3:

It's happening because arithmetic operations return `NA` when there is an `NA` value in what's being operated on.

```
mean(read.csv("../../data/titanic.csv")$Age, na.rm=TRUE)
```

```
## [1] 29.69912
```

# Data Cleaning - Q4:

Given a vector `vec` write R code to scale it using unit interval scaling.

# Data Cleaning - A4:

In unit interval scaling the minimum value becomes zero, and the maximum value becomes one.

```
vec=sample(0:100, 50)
scaled=(vec - min(vec)) / (max(vec) - min(vec))
head(scaled, 7)
```

```
## [1] 0.4242424 0.4545455 0.8181818 0.9696970 0.8080808 0.7575758 0.7474747
```

```
min(scaled)
```

```
## [1] 0
```

```
max(scaled)
```

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

# Data Cleaning - Q5:

Given a vector `vec` write R code to scale it using z-scaling, and explain how you'd create your own function to do the same.

# Data Cleaning - A5:

```
vec=sample(0:100, 50)
```

```
scaled=scale(vec)
c(mean(scaled), sd(scaled))
```

```
## [1] 1.637958e-18 1.000000e+00
```

```
scaled_manually=(vec - mean(vec)) / sd(vec)
c(mean(scaled_manually), sd(scaled_manually))
```

```
## [1] 1.637958e-18 1.000000e+00
```

# Data Cleaning - Q6:

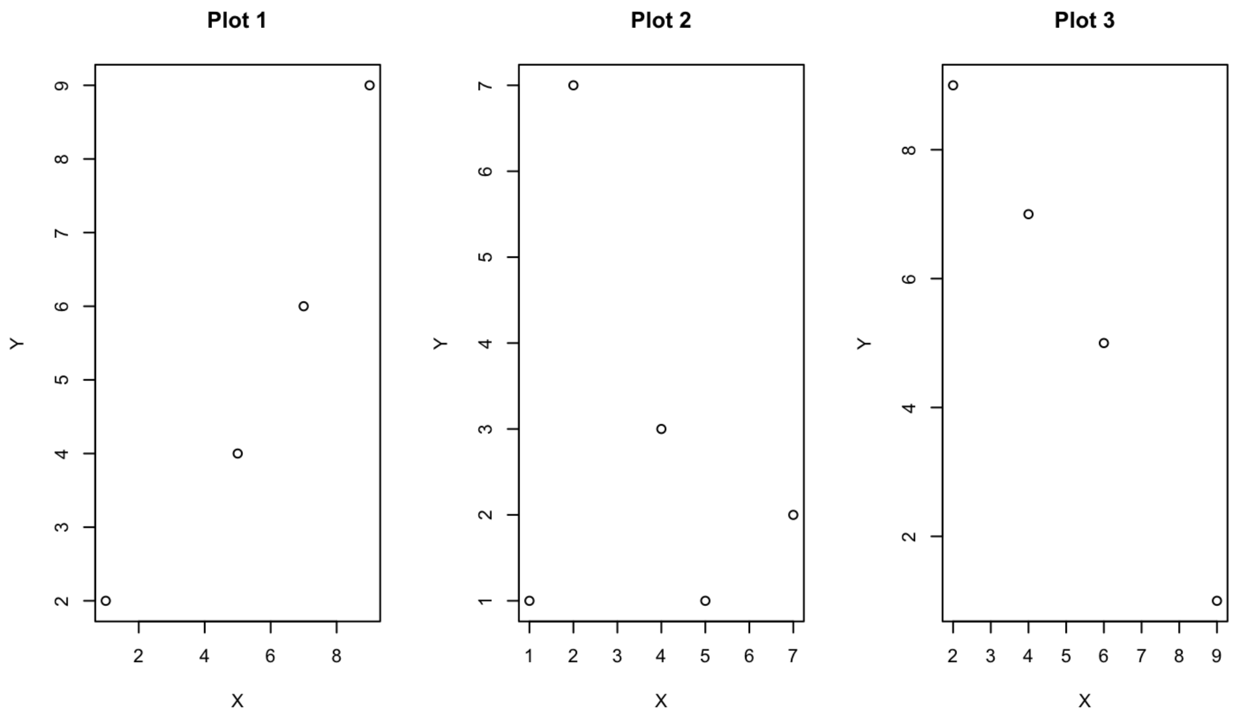
If we have a categorical value, height, with three possible labels (short, medium and tall), how many numerical features would be needed to contain the same information? Give examples of what they could be.

# Data Cleaning - A6:

Two numerical features would be needed. One could be `is.short` and the other could be `is.medium`. A `0` would indicate `FALSE` and a `1` a `TRUE`. If both are `0`, then that would mean the individual is neither short nor medium, and is therefore tall. So, to generalize, we need n - 1 numerical features to convert a categorical feature with n possible categories.

# Data Cleaning - Q7:

In the following three plots, estimate the p correlation value:



# Data Cleaning - A7:

This can be subjective, but to me *Plot 1* looks like it has a pretty strong positive correlation, so p is close to 1. *Plot 2* looks like it has no trend at all, so p=0 and *Plot 3* looks like a strong downward trend, so p is close to -1.

# KNN

# KNN - Q1:

KNN is used for:

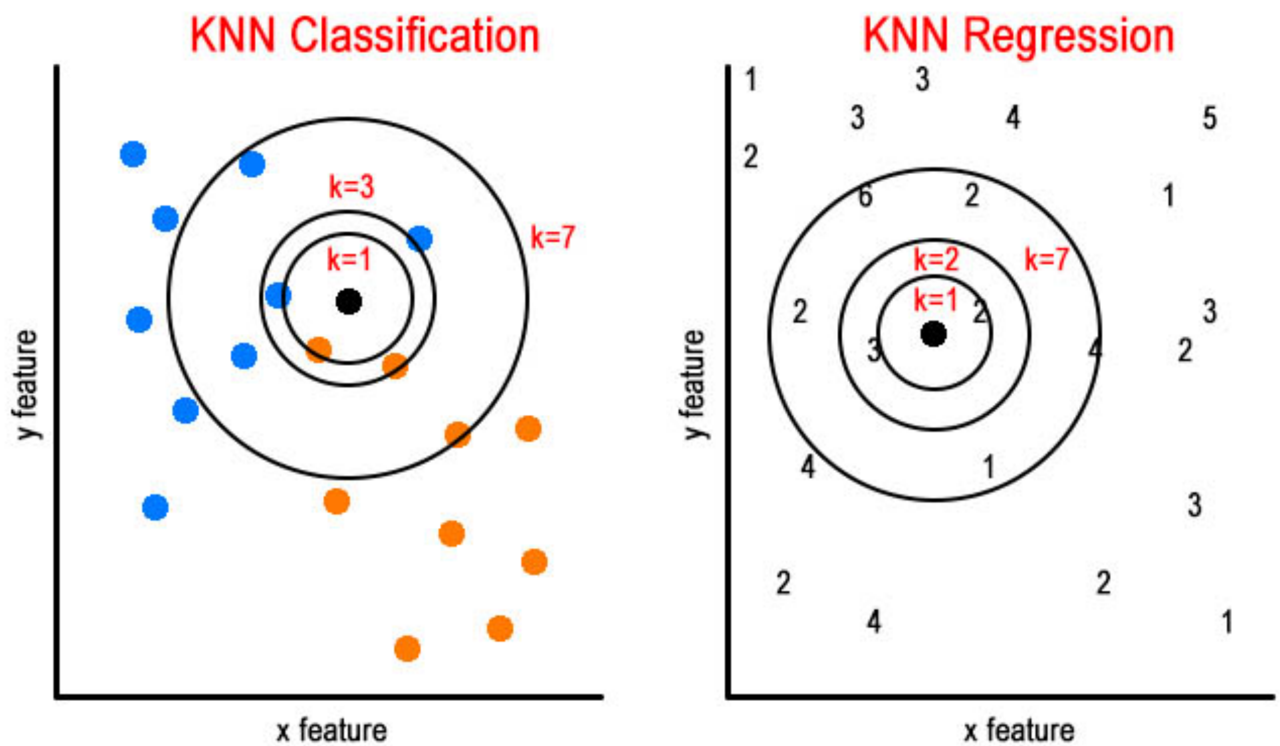
1. Classification
2. Regression
3. Both A and B
4. Anomaly detection
5. All of the above

# KNN - A1:

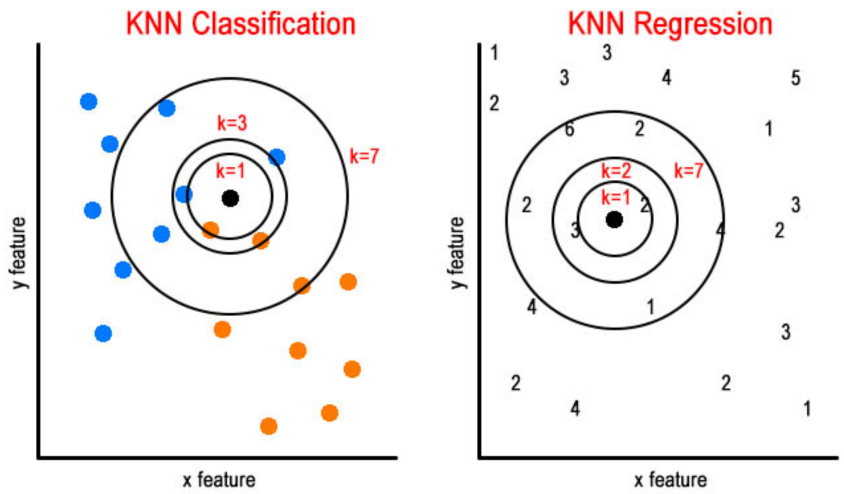
KNN can be used for classification, by looking at the label of the K nearest neighbors, regression by taking the mean of the KNN, and anomaly detection by looking at the distance between a point and the mean of the KNN or the absolute distance of the K nearest neighbor

# KNN - Q2:

Using the left side of the image below, what is the class of the black dot when `k=3`, and what is it when `k=7`? How did you come up with your answer?



# KNN - A2:



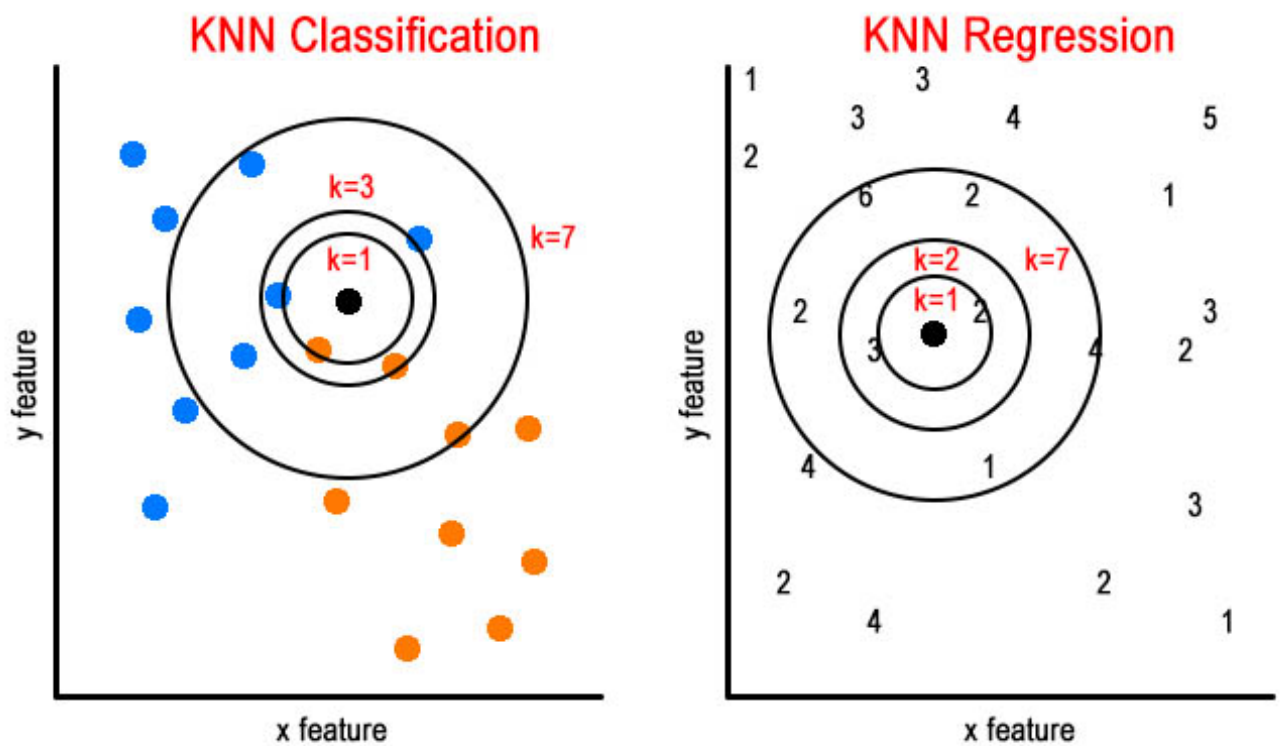

There are 2 orange and 1 blue when `k=3`, which would make the black belong to class orange.

For `k=7` there are 4 blue, and 3 orange, meaning the black would be predicted to be blue.

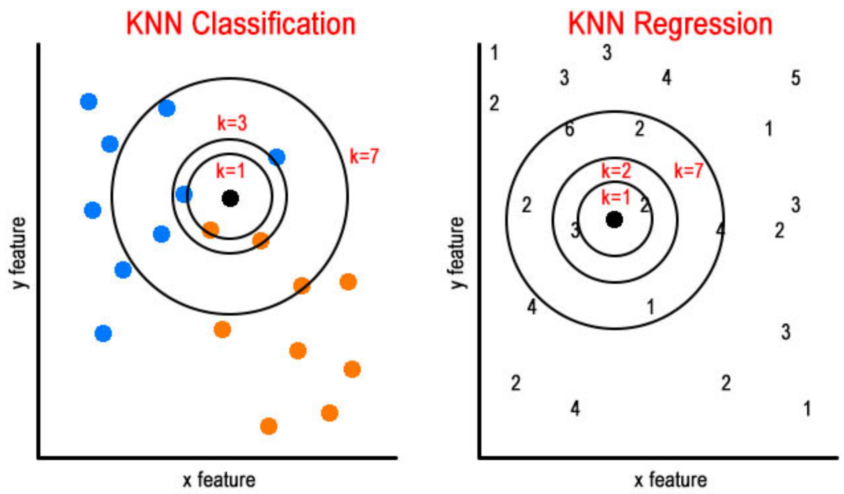
In KNN classification we take the most frequent label.

# KNN - Q3:

Using the right side of the image, what would the value of the black dot be if `k=1`? What if `k=2`? How did you come up with your answer?



# KNN - A3:




There's just one NN when `k=1` and that neighbor is `2`.

When `k=2` the 2 nearest neighbors are `2` and `3`. We take the mean of those, and get `2.5`.

In KNN regression we take the mean of the values of the k nearest neighbors.

# KNN - Q4:

True or False: when making a prediction using KNN, we need to remember all the training data. Why?

# KNN - A4:

This is true. When making a prediction we need to calculate the distance from the new point, to all the other points in order to find the k nearest points. This means we have to remember all the training points.

# KNN - Q5:

Write an R function to calculate the distance between two points. The function should take the parameters `x1`, `y1`, `x2`, `y2` and `manhattan`, which is a boolean value. If `manhattan` is *TRUE*, your function should calculate the *manhattan* distance, otherwise it should calculate the *euclidian* distance. By default, the `manhattan` parameter should be *FALSE*.

# KNN - A5:

```
dist=function(x1, y1, x2, y2, manhattan=FALSE) {
  if (manhattan) {
    return (abs(x2-x1) + abs(y2-y1))
  }
  else {
    return (sqrt((x2-x1)^2 + (y2-y1)^2))
  }
}
dist(0,0,1,1)
```

```
## [1] 1.414214
```

```
dist(0,0,1,1,TRUE)
```

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

# KNN - Q6:

True or False: whether or not you scale your data makes a big difference when using KNN. Explain why.

# KNN - A6:

Whether or not you scale your data makes a huge difference when using KNN.

If the units on one axis are much bigger than the other, then that feature will be more significant when it comes to a KNN model. If, for example, all our data lies on one vertical line, then only the Y axis feature will be used to determine the K nearest neighbors.

# KNN - Q7:

By increasing k in a KNN model, you are:

1. Decreasing the chance of overfitting
2. Increasing the chance of overfitting

Why?

# KNN - A7:

As the k of your KNN model grows, the prediction becomes less influenced by its immediate neighbors. So, as k increases, you are less likely to be overfitting your model. However, if k is too big you may be start to have underfitting.

# Linear Regression

# Linear Regression - Q1:

What are the parameters of a linear regression model?

# Linear Regression - A1:

The parameters for a linear regression model are the coefficients of the line. The equation of a line is:

```
y=mx + b
```

Where the m is the slope and b is the intercept. If we have multiple inputs:

```
Y=β0 + β1X1 + β2X2 + ... + βnXn
```

This one equation is enough to take the input and calculate the output of the model.

# Linear Regression - Q2:

How is a linear regression model trained?

# Linear Regression - A2:

The model is created by tuning the coefficients until the error of the residuals is minimized.

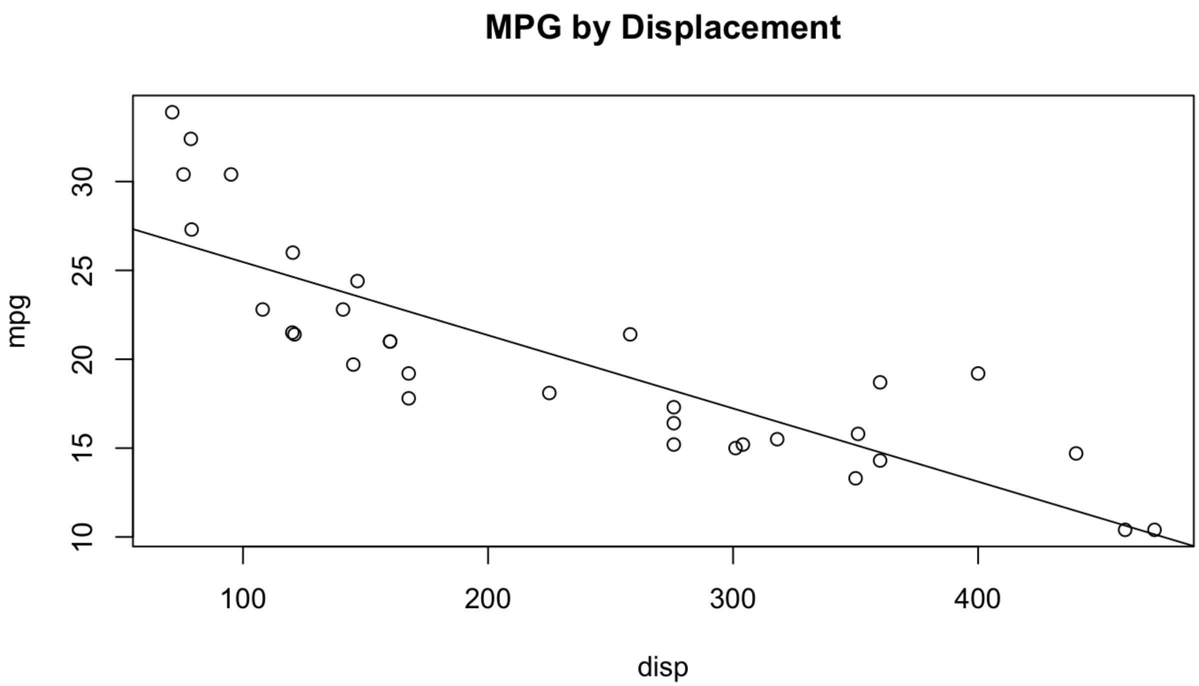
# Linear Regression - Q3:

Create a linear model in R, to predict `mpg` using `disp` using the `mtcars` data.

Plot the data using a scatterplot, and add the line from the model.

# Linear Regression - A3:

```
fit=lm(mpg ~ disp, data=mtcars)
plot(mpg ~ disp, data=mtcars, main="MPG by Displacement")
abline(fit)
```



# Linear Regression - Q4:

What is MSE and what is it used for?

# Linear Regression - A4:

MSE stands for "Mean Squared Error" and is the average of each residual value squared.

# Linear Regression - Q5:

What's the difference between RMSE and MSE. Why would you use one over the other?

# Linear Regression - A5:

RMSE is the "Root Mean Squared Error" and is the same as `sqrt(MSE)`

It can give you a more interpretable representation of the error, by providing the error in `units` rather than `units^2`.

# Linear Regression - Q6:

Given a linear model `fit` print the significance of each predictor. Which predictors would you consider removing from the model?

# Linear Regression - A6:

```
fit=lm(mpg ~ disp + qsec + drat, data=mtcars)
summary(fit)
```

```
##
## Call:
## lm(formula = mpg ~ disp + qsec + drat, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4681 -2.0867 -0.7474  1.1838  6.4843
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.524390  11.887430   0.969 0.340616
## disp        -0.031364   0.007809  -4.017 0.000402 ***
## qsec         0.403403   0.382875   1.054 0.301067
## drat         2.391842   1.637812   1.460 0.155314
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.226 on 28 degrees of freedom
## Multiple R-squared:  0.7413, Adjusted R-squared:  0.7135
## F-statistic: 26.74 on 3 and 28 DF,  p-value: 2.274e-08
```

# Linear Regression - Q7:

True or False: To when we predict using a linear model, we need to "remember" all the training data, just like in KNN.

# Linear Regression - A7:

False. When we predict using a linear model, all we need are the coefficients We can use the function of the line to make our predictions without having to remember any training data.

# Logistic Regression

# Logistic Regression - Q1:

Logistic regression is used to predict for:

1. Regression problems
2. Classification problems

# Logistic Regression - A1:

It's used for classification. Logistic regression is a way of using linear regression for classification problems.

# Logistic Regression - Q2:

Write the equation for the logistic curve or "squashing" function, and explain what it means.

# Logistic Regression - A2:

```
y=(e^x) / (1 + e^x)
```

# Logistic Regression - Q3:

In the logistic function (the "squashing" function), what do we use for the variable x?

# Logistic Regression - A3:

The variable x is the equation of the linear model. For example, if the linear model is:

$Y=\beta_0 + \beta_1 X_1 + \beta_2 X_2$

then x in the logistic equation would be substituted with:

$\beta_0 + \beta_1 X_1 + \beta_2 X_2$

# Logistic Regression - Q4:

Write R code to create a logistic regression model to predict `am` (automatic or manual) using all other features of the `mtcars` data.

# Logistic Regression - A4:

```
fit=glm(am ~ . , data=mtcars, family=binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(fit)
```

```
##
## Call:
## glm(formula = am ~ ., family = binomial, data = mtcars)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.061e-05  -6.239e-07  -2.110e-08   2.110e-08   1.309e-05
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.164e+01  1.840e+06       0        1
## mpg         -8.809e-01  2.884e+04       0        1
## cyl          2.527e+00  1.236e+05       0        1
## disp        -4.155e-01  2.570e+03       0        1
```

# Logistic Regression - Q5:

Is an RMSE value of 50 low, or is it high? Why?

# Logistic Regression - A5:

More information is needed!

The RMSE value is the Root Mean Squared Error. It's in the same units as the Y axis. For example, if we are trying to predict the age of a person by their height, then an error of 50 years is a very big error, compared to the range of the ages on the Y axis. But, if we're trying to predict the profit a company will make in Q2 of 2019 then an error of $50 is probably not a significant error at all.

# Logistic Regression - Q6:

How is the best possible equation for the logistic regression determined?

# Logistic Regression - A6:

Whereas for linear regression the coefficients were modified till the error was reduced to the smallest error possible, in logistic regression maximum likelihood principal is used. This means the model is adjusted till the probability of getting the results we should be getting is the highest.

# Logistic Regression - Q7:

Once we have the equation for a logistic model, how is it used to make predictions?

# Logistic Regression - A7:

A simple way to make predictions using a logistic model is to round the output of the logistic function. This means that if it's 0.5 or higher then round up to 1, and if it's lower round down to 0.

Does the cutoff value always have to be 0.5?

# R Code

# R Code - Q1:

Write R code to get every other, alternating, row of the data frame `mtcars`.

# R Code - A1:

```
mtcars[c(TRUE,FALSE),]
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

# R Code - Q2:

Write R code to split `mtcars` into test and training data, with a 70/30 split.

# R Code - A2:

```
tr_rows=sample(nrow(iris), nrow(iris) * 0.7)
tr_dat=iris[tr_rows,]
te_dat=iris[-tr_rows,]

nrow(tr_dat) + nrow(te_dat) == nrow(iris)
```

```
## [1] TRUE
```

# R Code - Q3:

Write two lines of R code to summarize the `mtcars` data with built in functions.

# R Code - A3:

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

# R Code - Q4:

Write R code to sample 4 coin flips, 1000 times, and find the probability of the last flip being heads.

# R Code - A4:

```
mean(replicate(1000, sample(0:1, 4, TRUE))[4,])
```

```
## [1] 0.509
```

# R Code - Q5:

Write R code to calculate RMSE using the two vectors below:

```
actual=c(5,6,4,3,7,5,5,6)
predicted=c(4,6,3,4,6,4,7,6)
```

# R Code - A5:

```
rmse=sqrt(mean((actual - predicted)^2))
rmse
```

```
## [1] 1.06066
```

# R Code - Q6:

What is the expected output of this R snippet?

```
sum(rep(sample(0:1, 1), 1000))
```

# R Code - A6:

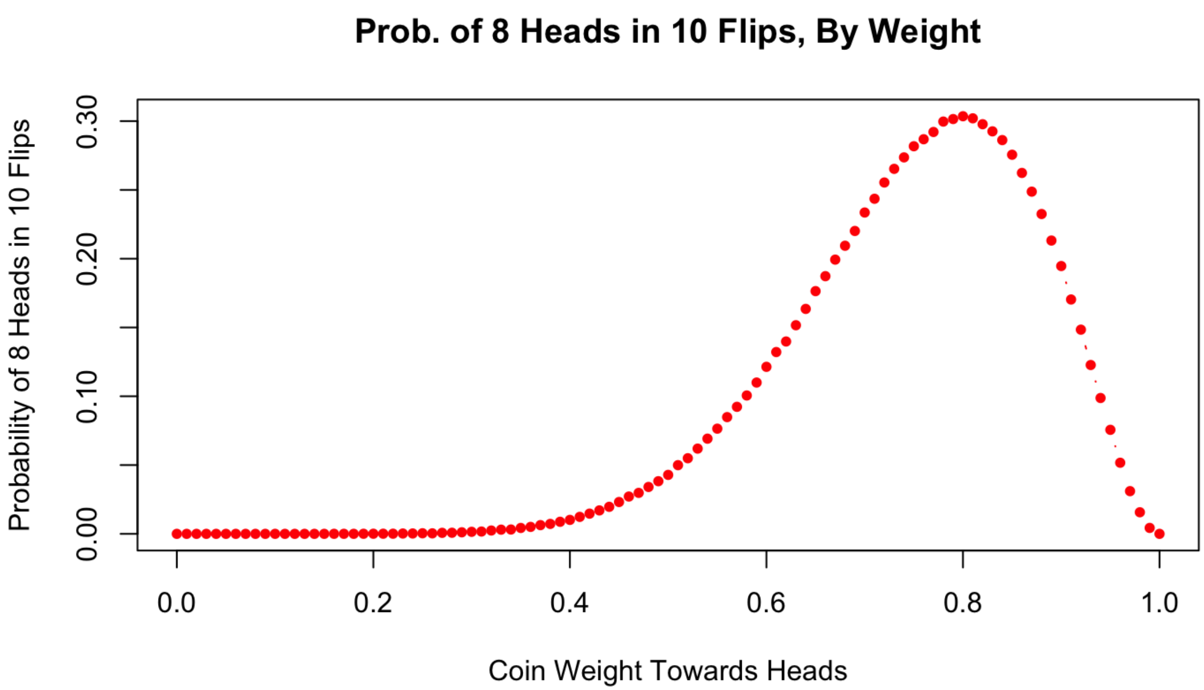
```
sum(rep(sample(0:1, 1), 1000))
```

```
## [1] 1000
```

It's going to be either 0 or 1000. This is because unlike the function `replicate()` the `rep()` function only evaluates the expression it's repeating one time. So, the result of the sample will be either 1 or 0, and that value will then be repeated 1000 times, adding to either 1000 or to 0.

# R Code - Q7:

Recreate this plot. The colors and labels don't have to match. You've seen it before!



# R Code - A7:

```
weights=seq(0,100,1)/100

get_probs=function(weight) {
  mean(replicate(100000,sum(sample(0:1,10,replace=TRUE,prob=c(1-weight,weight)))) == 8)
}

probs=sapply(weights, function(x) get_probs(x))

plot(
  weights,
  probs,
  col="red", type="b", pch=20,
  main="Prob. of 8 Heads in 10 Flips, By Weight",
  xlab="Coin Weight Towards Heads", ylab="Probability of 8 Heads in 10 Flips"
)
```