Code

# Assignment 2

## Task1: Simple Linear Regression

1. Download the file ‘insurance.csv’ from our class Blackboard site.
2. Read this file into your R environment. Show the step that you used to accomplish this.

Hide

```
insurance.df <- read.csv("insurance.csv", header = T) # load data
dim(insurance.df) # view dimensions
```

```
[1] 1338    7
```

3. Filter the dataframe to create a new dataframe that only contains the records of people who are not smokers. Show the code that you used to do this.

Hide

```
library(dplyr)
non_smokers <- filter(insurance.df, smoker == "no")
dim(non_smokers)
```

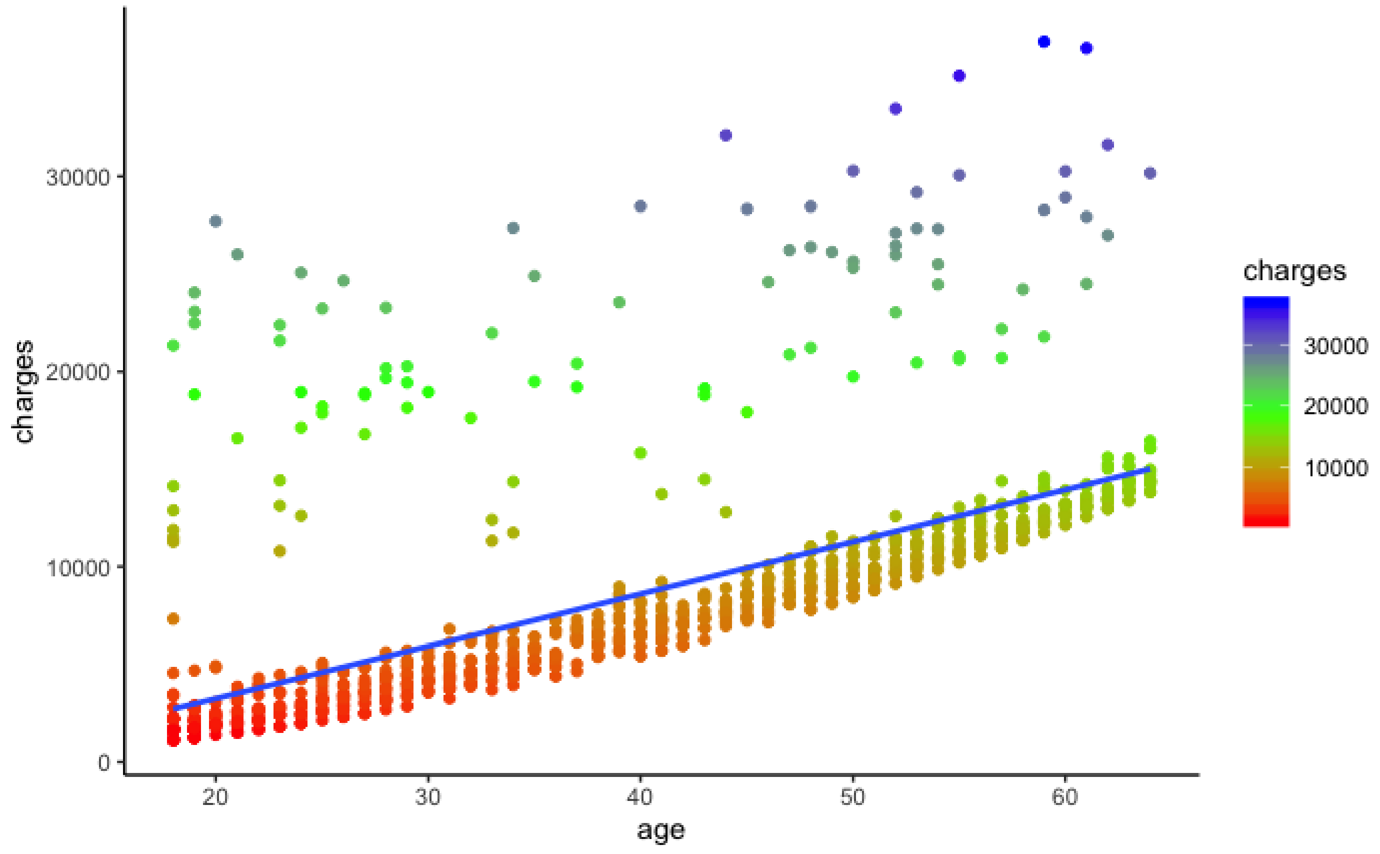
```
[1] 1064    7
```

– You will use this new dataframe for the rest of the assignment –

4. Using ggplot, create a scatterplot to depict the relationship between the input variable age and the output variable charges. Show your scatterplot, along with the code that you used to build it. What does this scatterplot suggest about the relationship between the two variables? Why (or why not) does this make intuitive sense to you?

Hide

```
library(ggplot2)
ggplot(non_smokers, aes(x = age, y = charges, color = charges)) +
  geom_point() +
  geom_smooth(method = "lm", se=FALSE) +
  scale_color_gradientn(colours = rainbow(3)) +
  theme_classic()
```



Hide

```
# There is a strong relationship between insurance charges and ages. This maske intuitiv
e sense since the older you become, the more of a deductible/premium you pay on insuranc
e policies.
```

5. Find the correlation between age and charges. Show the code that you used, and the results from your console, in a screenshot.

Hide

```
age_charges <- non_smokers[, c(1, 7)]
cor(age_charges) # Correlation table
```

```
              age   charges
age     1.0000000 0.6279468
charges 0.6279468 1.0000000
```

Hide

```
cor(non_smokers$age, non_smokers$charges, use="complete.obs") # alternative
```

```
[1] 0.6279468
```

6. Using your assigned seed value, create a data partition. Assign approximately 60% of the records to your training set, and the other 40% to your validation set. Show the code that you used to do this.

Hide

```
sample_size <- floor(0.60 * nrow(non_smokers))
set.seed(180) # set seed for reporducing the partion
train_index <- sample(seq_len(nrow(non_smokers)), size = sample_size)

training <- non_smokers[train_index,]
validation <- non_smokers[-train_index,]
```

7. Using your training set, create a simple linear regression model, with the input variable age and the outcome variable charges. Show the step(s) that you used to do this. Include a screenshot of the summary of your model, along with the code you used to generate that summary.

Hide

```
insurance_model <- lm(charges~age, data = training)
options(scipen = 999, digits = 0)
summary(insurance_model)
```

```


Call:
lm(formula = charges ~ age, data = training)

Residuals:
   Min     1Q Median     3Q    Max 
 -3224  -1998  -1423   -647  23244 

Coefficients:
            Estimate Std. Error t value             Pr(>|t|)    
(Intercept)  -1877.5      563.6   -3.33              0.00091 ***
age            263.5       13.3   19.84 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0 ‘**’ 0 ‘*’ 0 ‘.’ 0 ‘ ’ 1

Residual standard error: 4770 on 636 degrees of freedom
Multiple R-squared:  0.382, Adjusted R-squared:  0.381 
F-statistic:  394 on 1 and 636 DF,  p-value: <0.0000000000000002
```

8. What is the regression equation generated by your model? Make up a hypothetical input value and explain what it would predict as an outcome. To show the predicted outcome value, you can either use a function in R, or just explain what the predicted outcome would be, based on the regression equation and some simple math.

Hide

```
# regression equation -> y = -1877.5 + 263.5x
# if 35 = x, then,
-1877.5 + 263.5 * 35
```

```
[1] 7345
```

Hide

```
# for being 35 years old, your charges would be $7,345
```

9. Using the accuracy() function from the forecast package, assess the accuracy of your model against both the training set and the validation set. What do you notice about these results? Describe your findings in a couple of sentences.

Hide

```
library(forecast)
prediction1 <- predict(insurance_model, training) # predict training
accuracy(prediction1, training$charges)
```

```
         ME RMSE  MAE MPE MAPE
Test set -0 4767 2673 -24   36
```

Hide

```
prediction2 <- predict(insurance_model, validation) # predict validation
accuracy(prediction2, validation$charges)
```

```
           ME RMSE  MAE MPE MAPE
Test set -161 4503 2562 -27   38
```

Hide

```
# With the exception of the ME, the RMSE, MAE, MPE, and MAPE are all within range
# This makes the training and validation set pretty accurate
```

## Task 2: K-Nearest Neighbors

The model that we'll build will aim to predict which species of fish is being sold at a popular urban fish market, using only the numeric attributes as inputs. The outcome variable of this model will be Species. The numeric attributes are described below:

Weight = weight of fish in Gram g

Length1 = vertical length in cm

Length2 = diagonal length in cm

Length3 = cross length in cm

Height = height in cm

Width = diagonal width in cm

1. Download the file 'fishmarket.csv' from our class Blackboard site.
2. Read this file into your R environment. Show the step that you used to accomplish this.

Hide

```
fishmarket.df <- read.csv("fishmarket.csv", header = T) # load data
dim(fishmarket.df)
```

```
[1] 159   7
```

3. Using your assigned seed value (from Assignment 2), partition your data into training (60%) and validation (40%) sets. Show the step(s) that you used to do this.

Hide

```
library(dplyr)
set.seed(180) # set seed for reporducing the partion
fishmarket <- sample_frac(fishmarket.df, 1)
training <- slice(fishmarket, 1:95)
validation <- slice(fishmarket, 96:159)
```

4. Make up a fake fish (yes, really!)

a. Give your fish a name (there's no R code needed here, and you won't use the name when you run k-nn… but give the fish a name anyway and just write it here).

Hide

```
# balloon_molly
```

b. Use the runif() function to give your fish values for each of the six numeric attributes. Use the min and max values from your training set as the lower and upper boundaries for runif().

Hide

```
Weight <- runif(1, min(training$Weight), max(training$Weight))
Lenght1 <- runif(1, min(training$Length1), max(training$Length1))
Lenght2 <- runif(1, min(training$Length2), max(training$Length2))
Lenght3 <- runif(1, min(training$Length3), max(training$Length3))
Height <- runif(1, min(training$Height), max(training$Height))
Width <- runif(1, min(training$Width), max(training$Width))

balloon_molly <- data.frame(Weight = 172,
                            Length1 = 12,
                            Length2 = 43,
                            Length3 = 48,
                            Height = 10.841,
                            Width = 7)
```

5. Normalize your data using the preProcess() function from the caret package. Use Table 7.2 from the book as a guide for this. Show the step(s) that you used to do this.

Hide

```
library(caret)
```

```
Loading required package: lattice
Loading required package: ggplot2
Registered S3 method overwritten by 'data.table':
  method           from
  print.data.table
```

Hide

```
training.norm <- training
validation.norm <- validation
balloon_molly.norm <- balloon_molly

summary(training[, 2:7])
```

```
     Weight        Length1       Length2       Length3        Height         Width   
 Min.   :   0   Min.   : 8   Min.   : 8   Min.   : 9   Min.   : 2   Min.   :1  
 1st Qu.: 150   1st Qu.:20   1st Qu.:22   1st Qu.:24   1st Qu.: 6   1st Qu.:4  
 Median : 300   Median :26   Median :29   Median :31   Median : 8   Median :4  
 Mean   : 430   Mean   :27   Mean   :30   Mean   :33   Mean   : 9   Mean   :5  
 3rd Qu.: 668   3rd Qu.:34   3rd Qu.:37   3rd Qu.:40   3rd Qu.:12   3rd Qu.:6  
 Max.   :1650   Max.   :59   Max.   :63   Max.   :68   Max.   :19   Max.   :8  
```

Hide

```
norm.values <- preProcess(training[,2:7], method = c("center", "scale"))
training.norm[, 2:7] <- predict(norm.values, training[, 2:7])
validation.norm[, 2:7] <- predict(norm.values, validation[, 2:7])
balloon_molly.norm[, 1:6] <- predict(norm.values, balloon_molly[, 1:6])
```

6. Using the knn() function from the FNN package, and using a k-value of 7, generate a predicted classification for your fish. Show the step(s) that you used to do this, along with the output in the console. What Species was your fish predicted to belong to? Also, who were your fish's 7 nearest neighbors? What Species' did they belong to? Show the step(s) that you used to find this out, along with a screenshot of the output in the console.

Hide

```
library(FNN)
nn <- knn(train = training.norm[, 2:7],
          test = balloon_molly.norm[, 1:6],
          cl = training.norm[, 1],
          k = 7)
row.names(training)[attr(nn, "nn.index")]
```

```
[1] "8"  "68" "36" "51" "60" "16" "80"
```

Hide

```
nn
```

```
[1] Bream
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    8   68   36   51   60   16   80
attr(,"nn.dist")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    3    3    3    3    3    3    3
Levels: Bream
```

Hide

```
# Per knn, my fish belonged to the 'Bream' species
# Based on the training dataframe, the following neighbors were,
# 8='Whitefish', 68='Perch', 36='Roach', 51:'Perch', 60='Bream', 16='Bream', 80='Bream'
```

7a. Use your validation set to help you determine an optimal k-value. Use Table 7.3 from the textbook as a guide here. Show the step(s) that you used to do this, along with the output in the console.

Hide

```
accuracy.df <- data.frame(k = seq(1, 95, 1), accuracy = rep(0,95))

for(i in 1:95) {
  knn.pred <- knn(training.norm[, 2:7],
                  validation.norm[, 2:7],
                  cl = training.norm[, 1],
                  k = i)
accuracy.df[i, 2] <- confusionMatrix(knn.pred, validation.norm[, 1])$overall[1]
}
```

```
longer object length is not a multiple of shorter object lengthLevels are not in the sam
e order for reference and data. Refactoring data to match.longer object length is not a
multiple of shorter object lengthLevels are not in the same order for reference and dat
a. Refactoring data to match.longer object length is not a multiple of shorter object le
ngthLevels are not in the same order for reference and data. Refactoring data to match.l
onger object length is not a multiple of shorter object lengthLevels are not in the same
order for reference and data. Refactoring data to match.longer object length is not a mu
ltiple of shorter object lengthLevels are not in the same order for reference and data.
Refactoring data to match.longer object length is not a multiple of shorter object lengt
hLevels are not in the same order for reference and data. Refactoring data to match.long
er object length is not a multiple of shorter object lengthLevels are not in the same or
der for reference and data. Refactoring data to match.longer object length is not a mult
iple of shorter object lengthLevels are not in the same order for reference and data. Re
factoring data to match.longer object length is not a multiple of shorter object lengthL
evels are not in the same order for reference and data. Refactoring data to match.longer
object length is not a multiple of shorter object lengthLevels are not in the same order
for reference and data. Refactoring data to match.longer object length is not a multiple
of shorter object lengthLevels are not in the same order for reference and data. Refacto
ring data to match.longer object length is not a multiple of shorter object lengthLevels
are not in the same order for reference and data. Refactoring data to match.longer objec
t length is not a multiple of shorter object lengthLevels are not in the same order for
reference and data. Refactoring data to match.longer object length is not a multiple of
shorter object lengthLevels are not in the same order for reference and data. Refactorin
g data to match.longer object length is not a multiple of shorter object lengthLevels ar
e not in the same order for reference and data. Refactoring data to match.longer object
length is not a multiple of shorter object lengthLevels are not in the same order for re
ference and data. Refactoring data to match.longer object length is not a multiple of sh
orter object lengthLevels are not in the same order for reference and data. Refactoring
data to match.longer object length is not a multiple of shorter object lengthLevels are
not in the same order for reference and data. Refactoring data to match.longer object le
ngth is not a multiple of shorter object lengthLevels are not in the same order for refe
rence and data. Refactoring data to match.longer object length is not a multiple of shor
ter object lengthLevels are not in the same order for reference and data. Refactoring da
ta to match.longer object length is not a multiple of shorter object lengthLevels are no
t in the same order for reference and data. Refactoring data to match.longer object leng
th is not a multiple of shorter object lengthLevels are not in the same order for refere
nce and data. Refactoring data to match.longer object length is not a multiple of shorte
r object lengthLevels are not in the same order for reference and data. Refactoring data
to match.longer object length is not a multiple of shorter object lengthLevels are not i
n the same order for reference and data. Refactoring data to match.longer object length
is not a multiple of shorter object lengthLevels are not in the same order for reference
and data. Refactoring data to match.longer object length is not a multiple of shorter ob
ject lengthLevels are not in the same order for reference and data. Refactoring data to
match.longer object length is not a multiple of shorter object lengthLevels are not in t
he same order for reference and data. Refactoring data to match.longer object length is
not a multiple of shorter object lengthLevels are not in the same order for reference an
d data. Refactoring data to match.longer object length is not a multiple of shorter obje
ct lengthLevels are not in the same order for reference and data. Refactoring data to ma
tch.longer object length is not a multiple of shorter object lengthLevels are not in the
same order for reference and data. Refactoring data to match.longer object length is not
a multiple of shorter object lengthLevels are not in the same order for reference and da
ta. Refactoring data to match.longer object length is not a multiple of shorter object l
engthLevels are not in the same order for reference and data. Refactoring data to match.
```

```
longer object length is not a multiple of shorter object lengthLevels are not in the sam
e order for reference and data. Refactoring data to match.longer object length is not a
multiple of shorter object lengthLevels are not in the same order for reference and dat
a. Refactoring data to match.longer object length is not a multiple of shorter object le
ngthLevels are not in the same order for reference and data. Refactoring data to match.l
onger object length is not a multiple of shorter object lengthLevels are not in the same
order for reference and data. Refactoring data to match.longer object length is not a mu
ltiple of shorter object lengthLevels are not in the same order for reference and data.
Refactoring data to match.longer object length is not a multiple of shorter object lengt
hLevels are not in the same order for reference and data. Refactoring data to match.long
er object length is not a multiple of shorter object lengthLevels are not in the same or
der for reference and data. Refactoring data to match.longer object length is not a mult
iple of shorter object lengthLevels are not in the same order for reference and data. Re
factoring data to match.longer object length is not a multiple of shorter object lengthL
evels are not in the same order for reference and data. Refactoring data to match.longer
object length is not a multiple of shorter object lengthLevels are not in the same order
for reference and data. Refactoring data to match.longer object length is not a multiple
of shorter object lengthLevels are not in the same order for reference and data. Refacto
ring data to match.longer object length is not a multiple of shorter object lengthLevels
are not in the same order for reference and data. Refactoring data to match.longer objec
t length is not a multiple of shorter object lengthLevels are not in the same order for
reference and data. Refactoring data to match.longer object length is not a multiple of
shorter object lengthLevels are not in the same order for reference and data. Refactorin
g data to match.longer object length is not a multiple of shorter object lengthLevels ar
e not in the same order for reference and data. Refactoring data to match.longer object
length is not a multiple of shorter object lengthLevels are not in the same order for re
ference and data. Refactoring data to match.longer object length is not a multiple of sh
orter object lengthLevels are not in the same order for reference and data. Refactoring
data to match.longer object length is not a multiple of shorter object lengthLevels are
not in the same order for reference and data. Refactoring data to match.longer object le
ngth is not a multiple of shorter object lengthLevels are not in the same order for refe
rence and data. Refactoring data to match.longer object length is not a multiple of shor
ter object lengthLevels are not in the same order for reference and data. Refactoring da
ta to match.longer object length is not a multiple of shorter object lengthLevels are no
t in the same order for reference and data. Refactoring data to match.longer object leng
th is not a multiple of shorter object lengthLevels are not in the same order for refere
nce and data. Refactoring data to match.longer object length is not a multiple of shorte
r object lengthLevels are not in the same order for reference and data. Refactoring data
to match.longer object length is not a multiple of shorter object lengthLevels are not i
n the same order for reference and data. Refactoring data to match.longer object length
is not a multiple of shorter object lengthLevels are not in the same order for reference
and data. Refactoring data to match.longer object length is not a multiple of shorter ob
ject lengthLevels are not in the same order for reference and data. Refactoring data to
match.Levels are not in the same order for reference and data. Refactoring data to matc
h.Levels are not in the same order for reference and data. Refactoring data to match.Lev
els are not in the same order for reference and data. Refactoring data to match.Levels a
re not in the same order for reference and data. Refactoring data to match.Levels are no
t in the same order for reference and data. Refactoring data to match.Levels are not in
the same order for reference and data. Refactoring data to match.Levels are not in the s
ame order for reference and data. Refactoring data to match.Levels are not in the same o
rder for reference and data. Refactoring data to match.Levels are not in the same order
for reference and data. Refactoring data to match.Levels are not in the same order for r
eference and data. Refactoring data to match.Levels are not in the same order for refere
nce and data. Refactoring data to match.Levels are not in the same order for reference a
```

```
nd data. Refactoring data to match.Levels are not in the same order for reference and da
ta. Refactoring data to match.Levels are not in the same order for reference and data. R
efactoring data to match.Levels are not in the same order for reference and data. Refact
oring data to match.Levels are not in the same order for reference and data. Refactoring
data to match.Levels are not in the same order for reference and data. Refactoring data 
to match.Levels are not in the same order for reference and data. Refactoring data to ma
tch.Levels are not in the same order for reference and data. Refactoring data to match.L
evels are not in the same order for reference and data. Refactoring data to match.Levels
are not in the same order for reference and data. Refactoring data to match.Levels are n
ot in the same order for reference and data. Refactoring data to match.Levels are not in
the same order for reference and data. Refactoring data to match.Levels are not in the s
ame order for reference and data. Refactoring data to match.Levels are not in the same o
rder for reference and data. Refactoring data to match.Levels are not in the same order 
for reference and data. Refactoring data to match.Levels are not in the same order for r
eference and data. Refactoring data to match.Levels are not in the same order for refere
nce and data. Refactoring data to match.Levels are not in the same order for reference a
nd data. Refactoring data to match.
```

Hide

```
accuracy.df
```

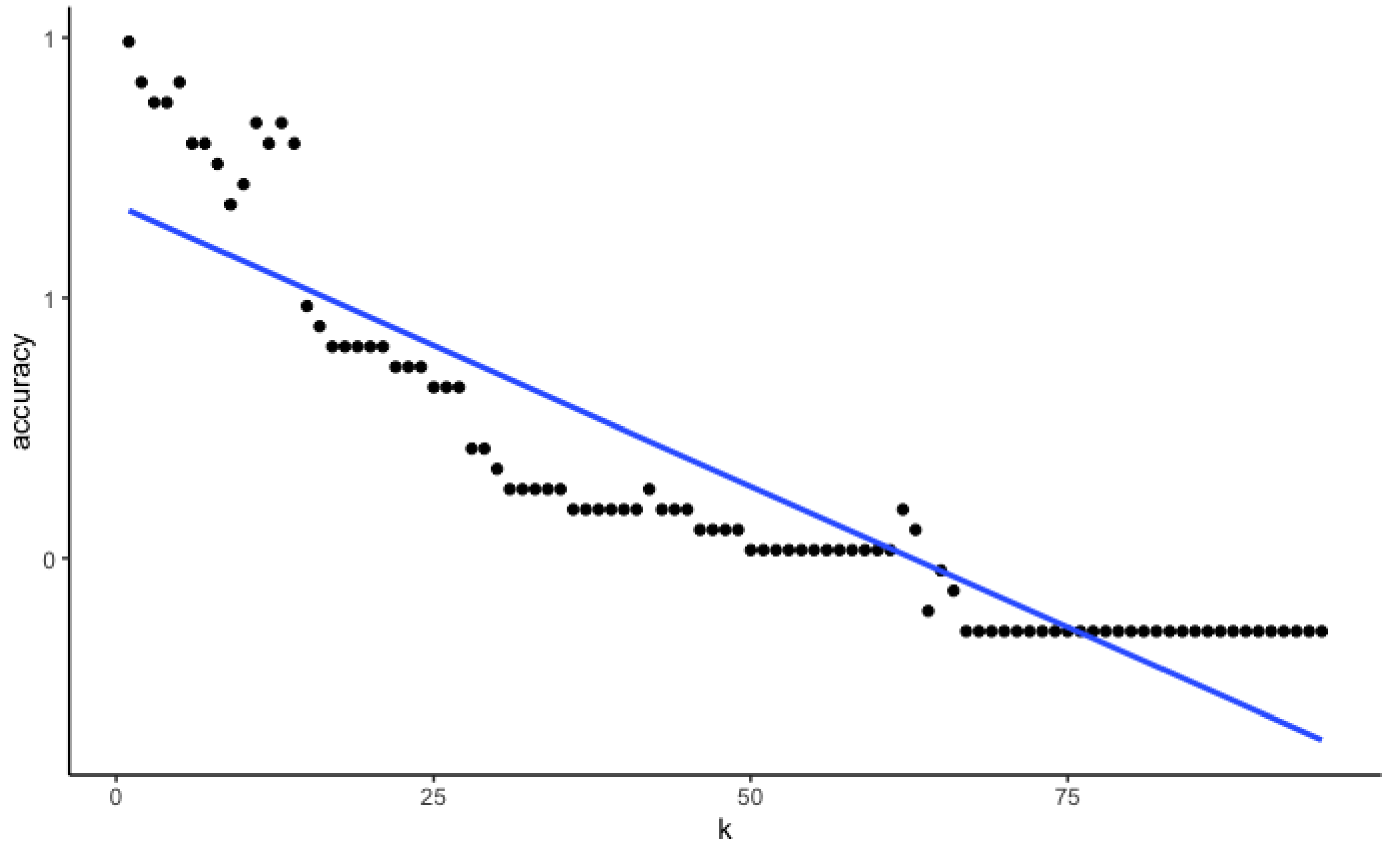
| k <dbl> | accuracy <dbl> |
|---:|---:|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |
| 8 | 1 |
| 9 | 1 |
| 10 | 1 |

1-10 of 95 rows Previous **1** 2 3 4 5 6 ... 10 Next

7b. Using either the base graphics package or ggplot, make a scatterplot with the various k values that you used in 7a on your x-axis, and the accuracy metrics on the y-axis.

Hide

```
library(ggplot2)
ggplot(accuracy.df, aes(x=k, y=accuracy)) +
  geom_point() +
  geom_smooth(method = "lm", se=FALSE) +
  theme_classic()
```



8. Re-run your knn() function with this new k-value. What result did you obtain? Was it different from the one you saw in Step 9? Show the step(s) that you used to do this, along with the output in the console. Also, what were the outcome classes (Species) for each of your fish's k-nearest neighbors?

Hide

```
nn.new <- knn(train = training.norm[, 2:7],
              test = balloon_molly.norm[, 1:6],
              cl = training.norm[, 1],
              k = 3)
nn.new
```

```
[1] Perch
attr(,"nn.index")
     [,1] [,2] [,3]
[1,]    8   68   36
attr(,"nn.dist")
     [,1] [,2] [,3]
[1,]    3    3    3
Levels: Perch
```

Hide

```
# The new outcome of new knn is now that 'balloon molly' is classfied as a Perch
# Using K=3 would indicate a new species. However, the index would never change from 1:9
5.
```