# Lab 2: Linear Regression

An Island Never Cries

*Paul Nguyen*

**Exercise 1**

```
data(quakes)
str(quakes)
```

```
## 'data.frame':    1000 obs. of  5 variables:
##  $ lat     : num  -20.4 -20.6 -26 -18 -20.4 ...
##  $ long    : num  182 181 184 182 182 ...
##  $ depth   : int  562 650 42 626 649 195 82 194 211 622 ...
##  $ mag     : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
##  $ stations: int  41 15 43 19 11 12 43 15 35 19 ...
```

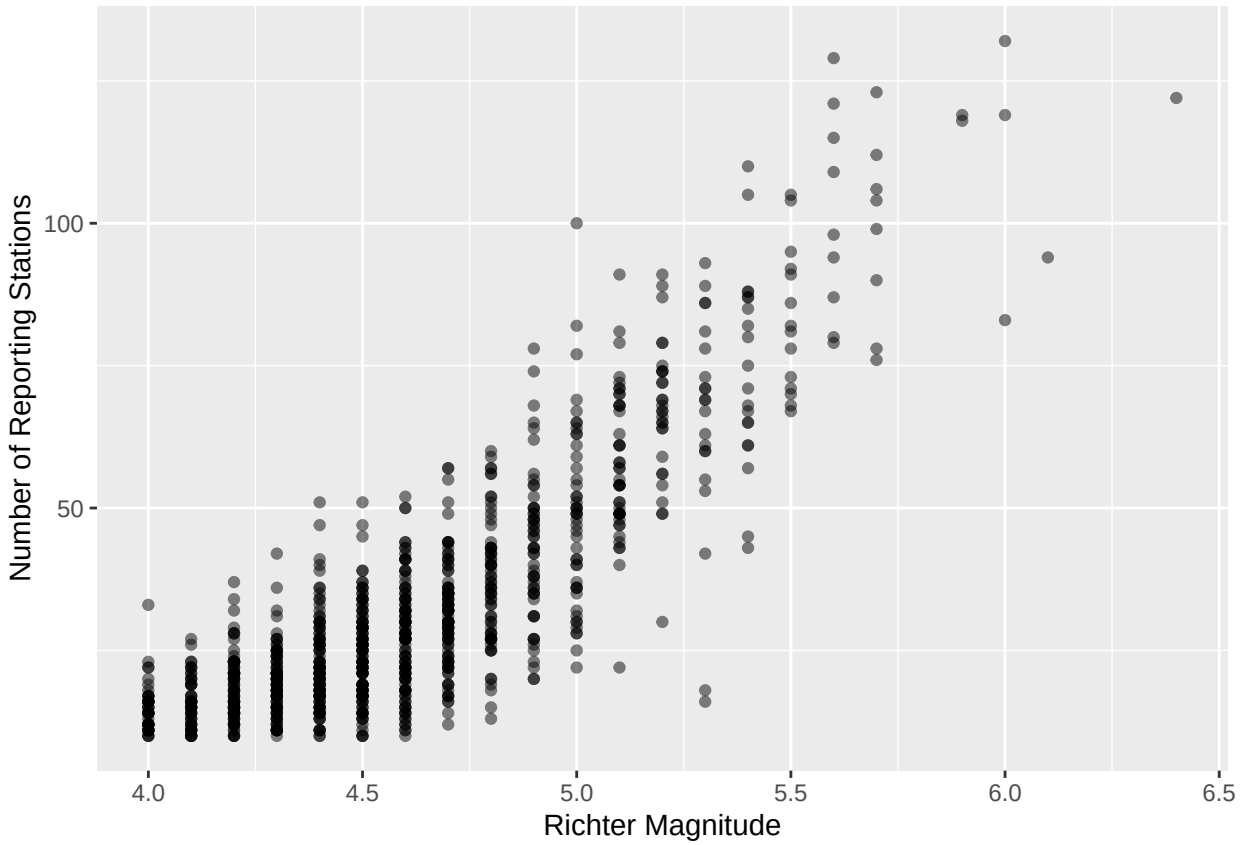
```
library(ggplot2)
library(gridExtra)
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------- tidyverse 1.2.
```

```
## v tibble  2.1.1       v purrr   0.3.2
## v tidyr   0.8.3       v dplyr   0.8.0.1
## v readr   1.3.1       v stringr 1.4.0
## v tibble  2.1.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------- tidyverse_conflicts()
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
ggplot(data = quakes, aes(x = mag, y = stations)) +
  geom_point(alpha = .49) +
  xlab ("Richter Magnitude") +
  ylab ("Number of Reporting Stations")
```



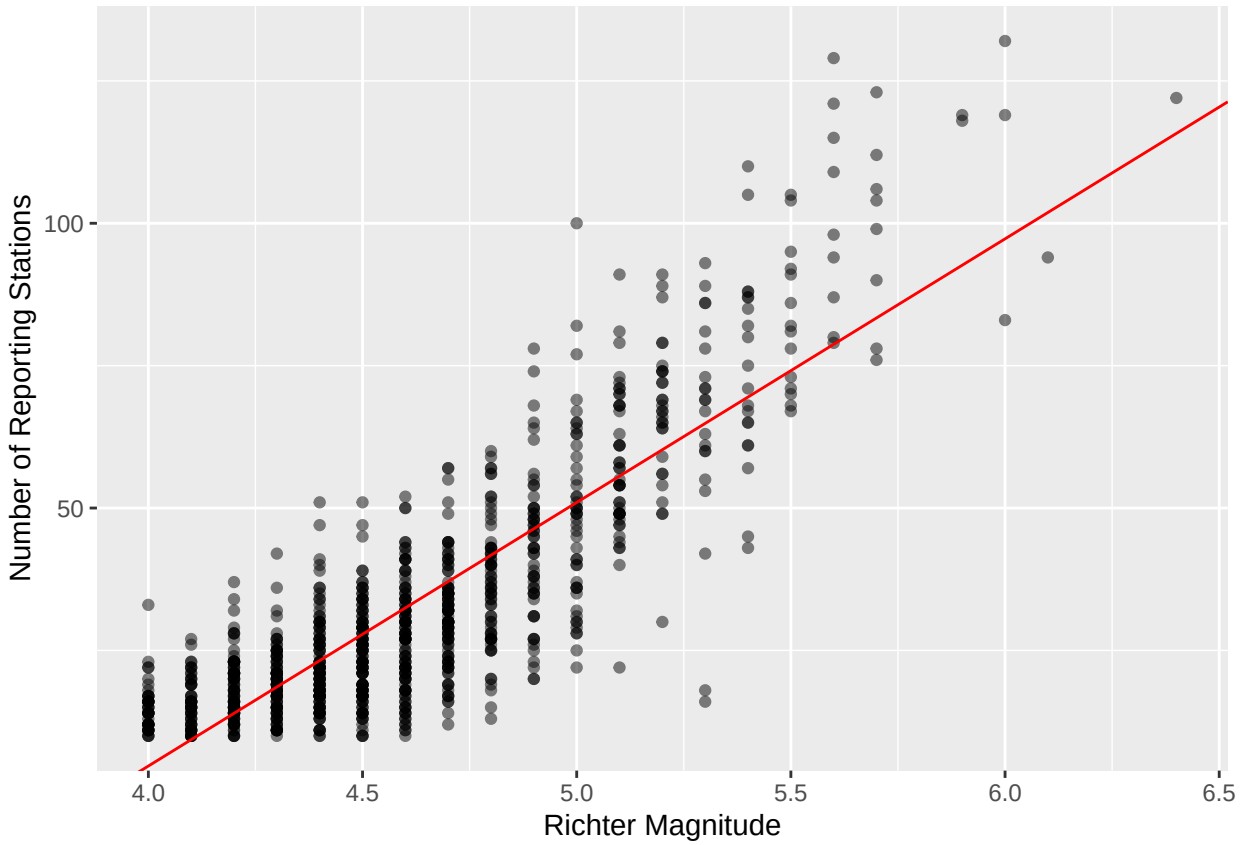

I would say that the magnitude of the Earthquake has a pretty strong positive relationship with the number of stations that reported it. The larger values the earthquake is on the Richter scale correspond to a higher value for number of stations.

### Exercise 2

If there were no relationship between these two variables, I would expected the slope of the line to be 0. I would expect the intercept of the line to be at the mean number of reporting stations.

### Exercise 3

```
m1 <- lm(stations ~ mag, data = quakes)
ggplot(data = quakes, aes(x = mag, y = stations)) +
  geom_point(alpha = .49) +
  xlab ("Richter Magnitude") +
  ylab ("Number of Reporting Stations") +
  geom_abline(slope = 46.28, intercept = -180.42, color = "red")
```




```
-180.42+(5*46.28)
```

```
## [1] 50.98
```

```
-180.42+(6*46.28)
```

```
## [1] 97.26
```

Interpreting slope: An increase in the magnitude of the earthquake by one should lead to an increase in the number of reporting stations by 46 stations, eg 5.0 to 6.0 can go from 51 reporting stations to 97 reporting stations. Interpreting intercept: Under the context for this problem, it doesn't make much sense, as you can't have a negative number of reporting stations, but if this trend were to continue so that the Richter Magnitude was 0, we would expect -180.42 stations to report it, but probably just 0 stations.

**Exercise 4**

```
#Calculating B1
magbar <- mean(quakes$mag)
stationsbar <- mean(quakes$stations)

B1 <-
  sum((quakes$mag - magbar)*(quakes$stations - stationsbar)) / (sum((quakes$mag - magbar)^2))
```

Can confirm B1 is 46.28

**Exercise 5**

```
predmag <- -180.42 + 46.28*(quakes$mag)
```

```
sxx <- sum((quakes$mag - magbar)^2)
siigmahat2 <- sum((quakes$stations - predmag)^2) / 998
SEB <- sqrt(siigmahat2)/(sqrt(sxx))



lbcf <- B1 - 1.96*(SEB)
ubcf <- B1 + 1.96*(SEB)
confint(m1, level = .95)
```

```
##                   2.5 %     97.5 %
## (Intercept) -188.64628 -172.20238
## mag           44.50944   48.05498
```

### Exercise 6

```
m1
```

```
##
## Call:
## lm(formula = stations ~ mag, data = quakes)
##
## Coefficients:
## (Intercept)          mag
##     -180.42        46.28
```

```
-180.42 +(46.28*7)
```

```
## [1] 143.54
```

I would expect ~144 stations to be able to detect an earthquake of magnitude 7.0

### Exercise 7

Goals for each questions 1. data description 2. data description 3. inference 4. inference 5. inference 6. prediction

### Exercise 9

```
xvec <- quakes$mag
```

### Exercise 10

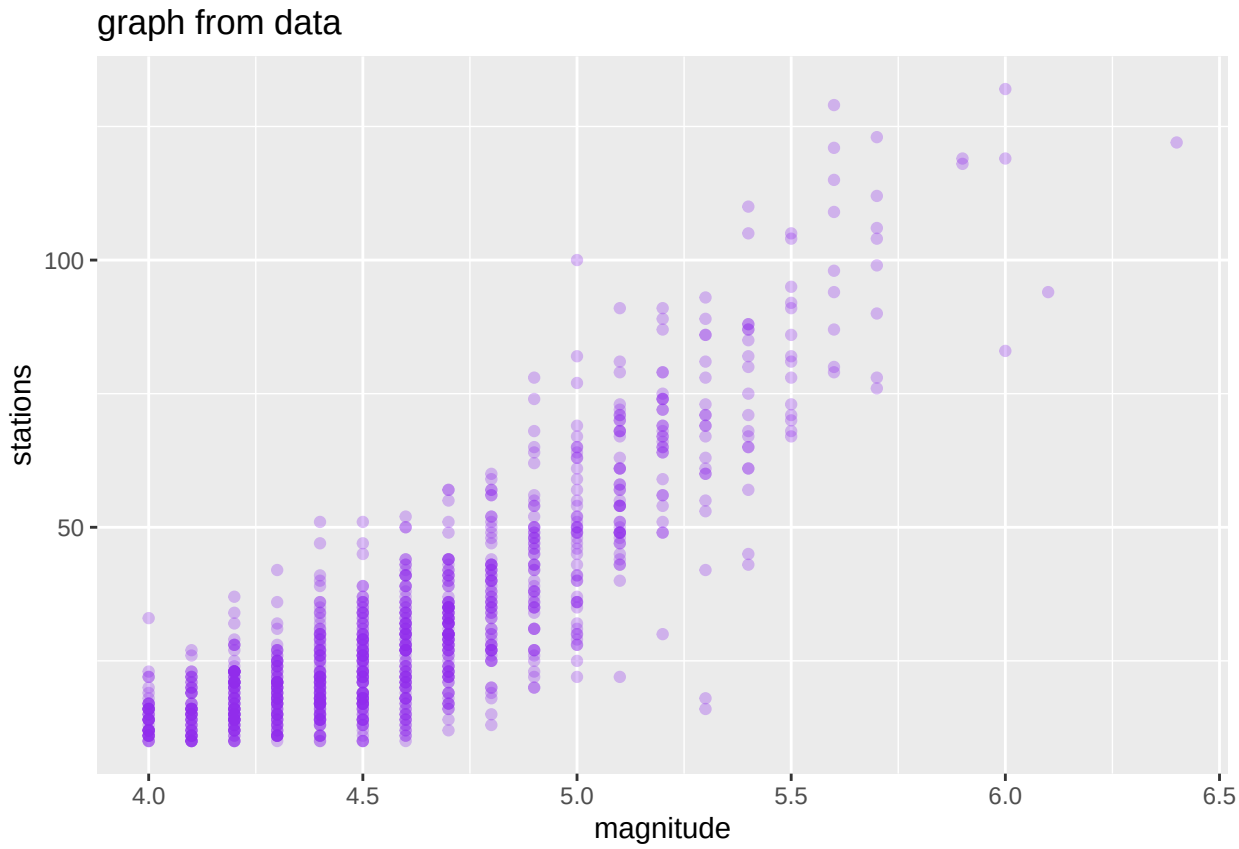
```
f_hat <- function(x) {
  y_hat <- -180.42 + 46.28*x
  return(y_hat)
}

#f_hat(xvec)
```

### Exercise 11

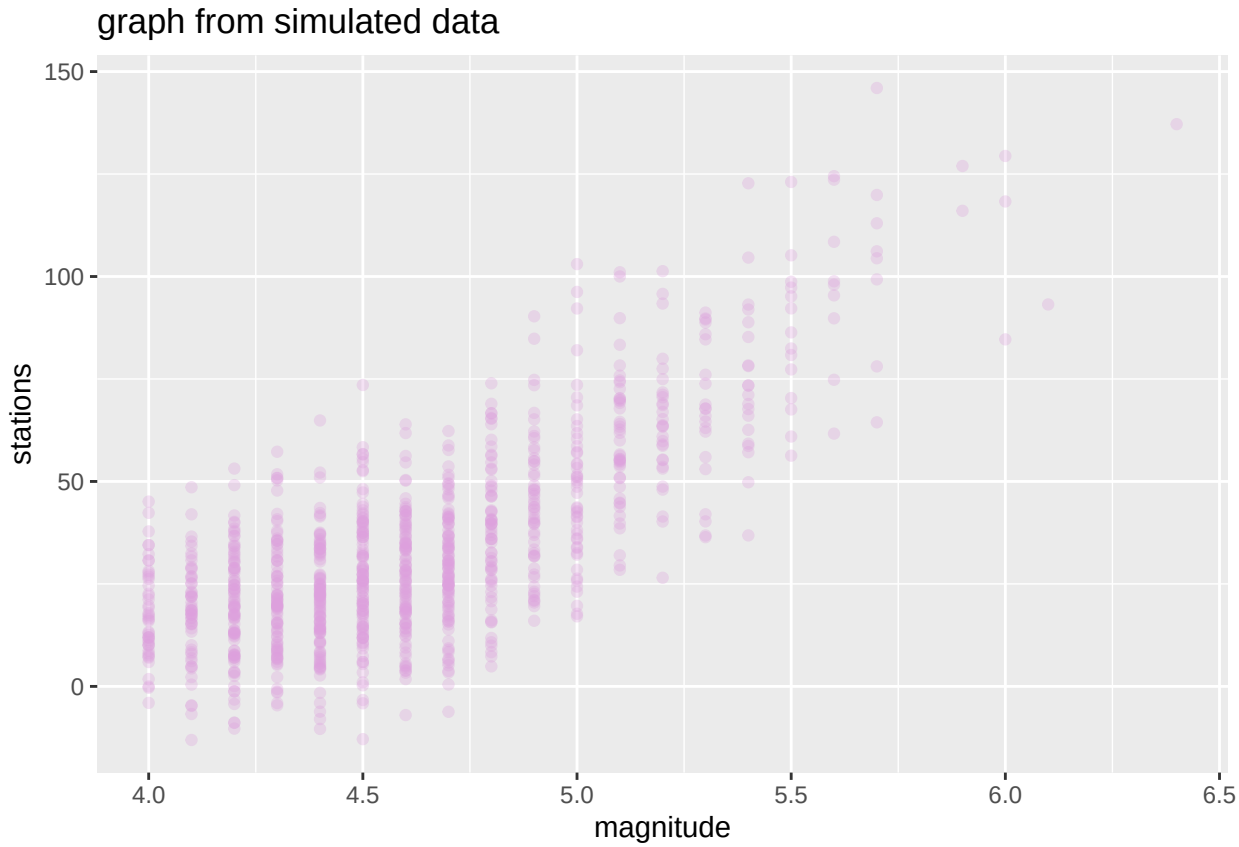
```
res <- m1$res
sigmahat2 <- sum(res^2)/998
why <- rnorm(length(xvec), mean = quakes$stations, sd = (sigmahat2)^(1/2))
```

**Exercise 12**

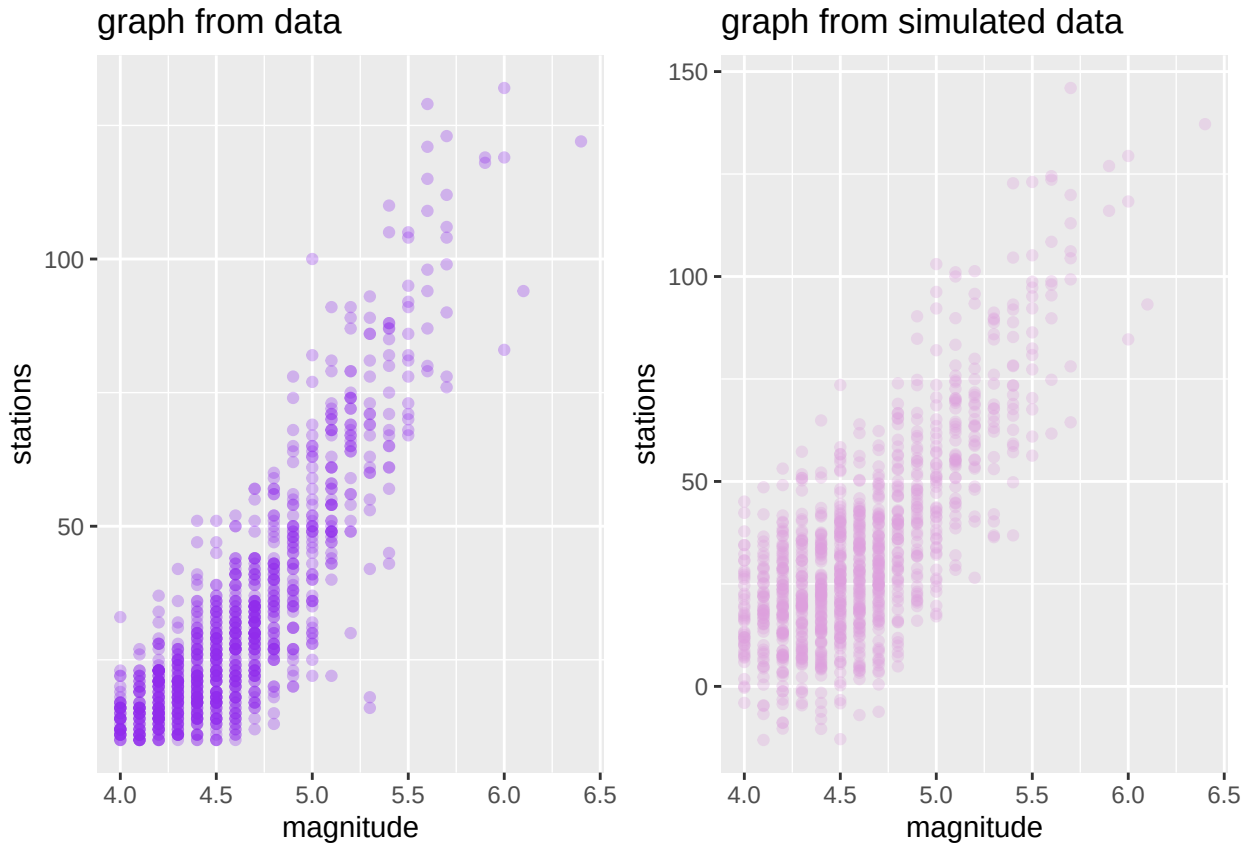
```
datagraph <- ggplot(data = quakes, mapping = aes(x = mag, y = stations)) +
  geom_point(alpha = .3, color = "purple2") +
  labs(title = "graph from data") +
  xlab("magnitude")
simgraph <- ggplot(mapping = aes(x = quakes$mag, y = why)) +
  geom_point(alpha = .3, color = "plum") +
  labs(title = "graph from simulated data") +
  xlab("magnitude") +
  ylab("stations")

datagraph
```




```
simgraph
```

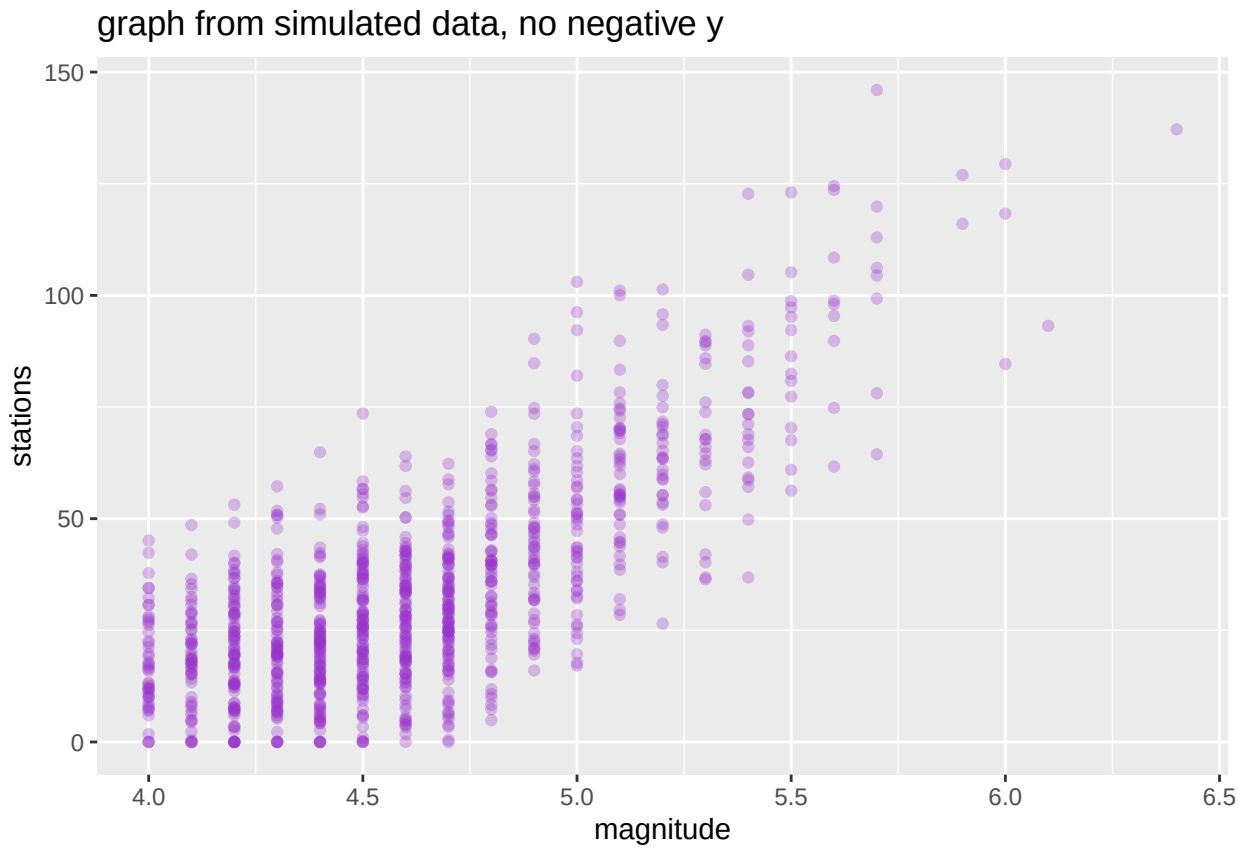



```
grid.arrange(datagraph, simgraph, nrow = 1)
```

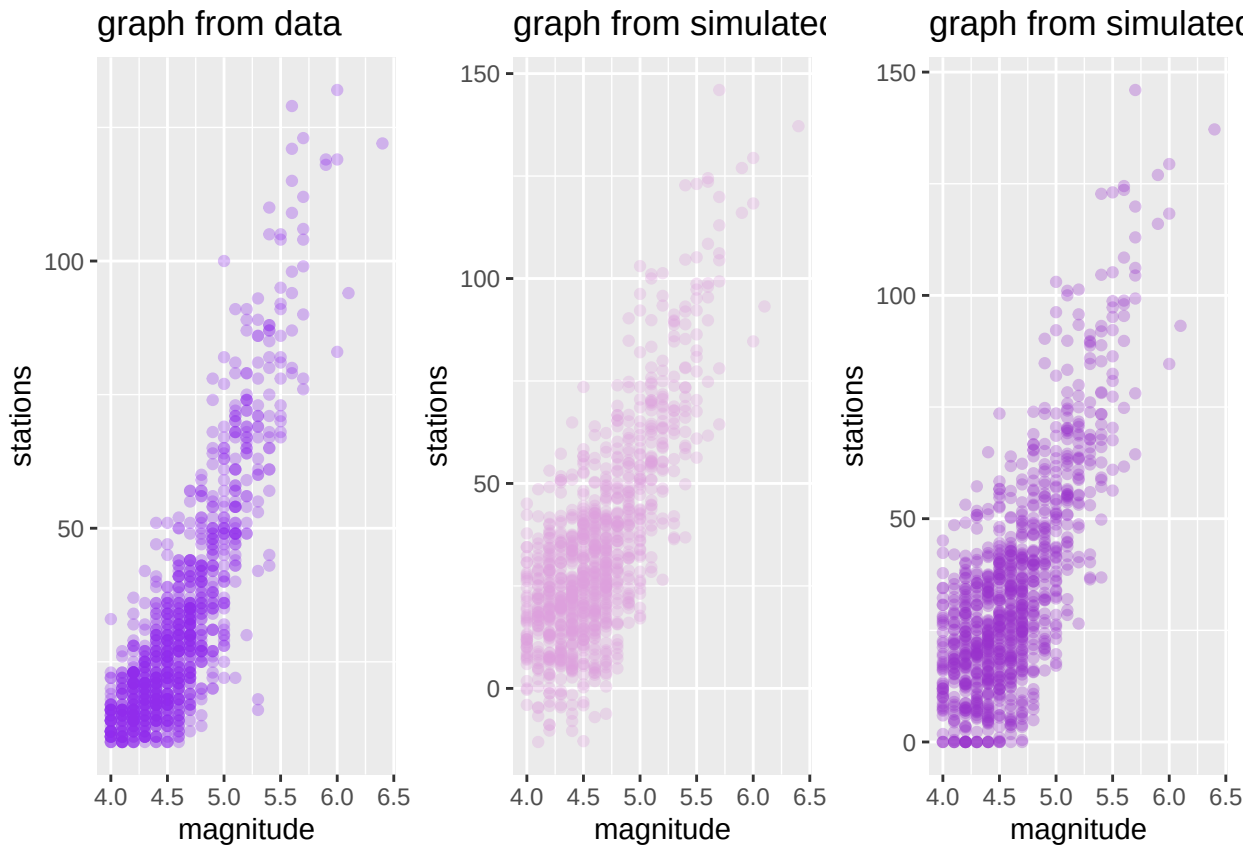



I would say that the shape of the simulated data is pretty similar to the actual data. The upward trend is similar, along with the less frequent high magnitude earthquakes, which makes sense, because I used the actual x values from the data to simulate the y values. It is different in that for the low magnitude earthquakes, in my simulated data, it is possible to have a negative amount of reporting stations, which is not good. If we restrict our simulated Y's so that they can only be greater or equal than 0, that would alleviate this problem.

```
whynoneg <- why
whynoneg[whynoneg < 0] <- 0

noneggraph <- ggplot(mapping = aes(x = quakes$mag, y = whynoneg)) +
  geom_point(alpha = .3, color = "darkorchid") +
  labs(title = "graph from simulated data, no negative y") +
  xlab("magnitude") +
  ylab("stations")
noneggraph
```



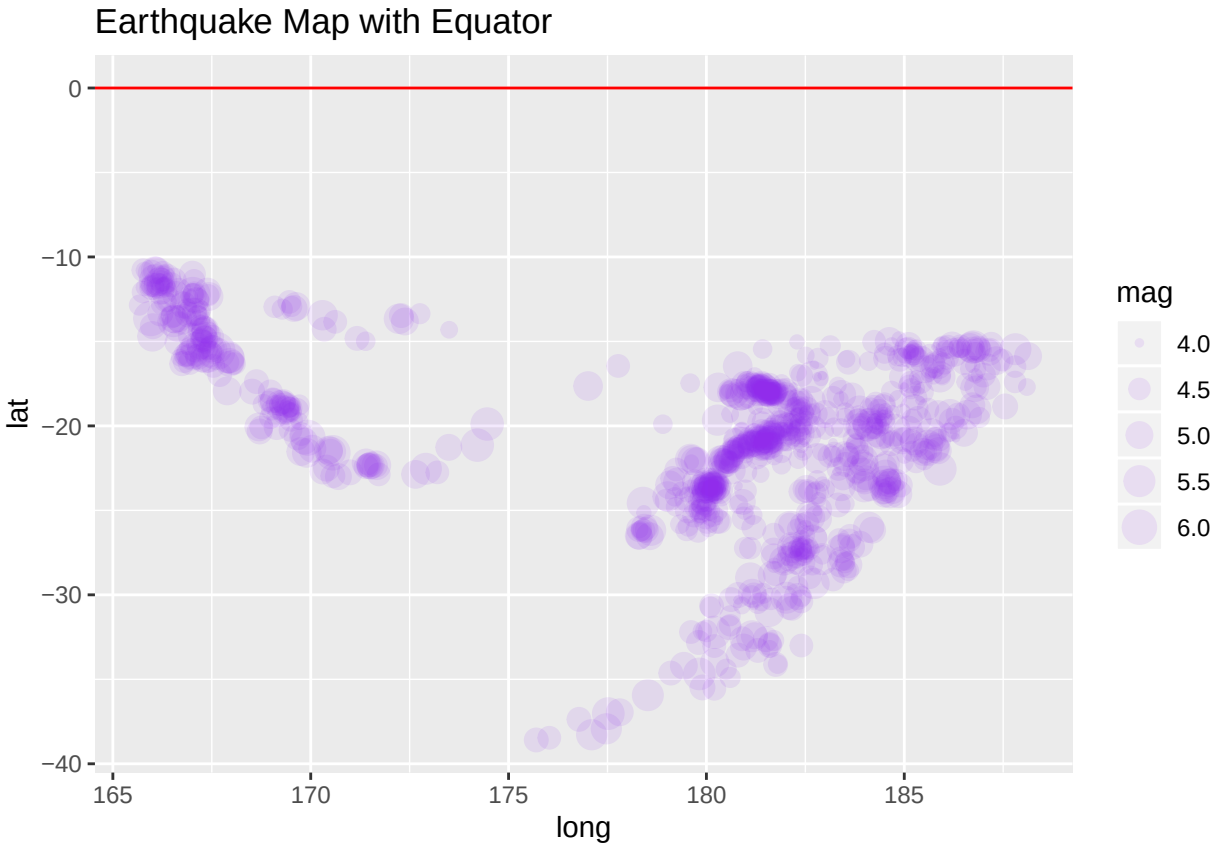

```
grid.arrange(datagraph, simgraph, noneggraph, nrow = 1)
```




**Challenge**

Use the latitude and longitude data to plot each of these earthquakes in quakes on a map with their magnitude mapped to the size of the plotting character. You may need to add some transparency to prevent overplotting.

```
ggplot(data = quakes, mapping = aes(x = long, y = lat, size = mag)) +
  geom_point(alpha = .1, color = "purple2") +
  geom_hline(yintercept = 0, color = "red") +
  labs(title = "Earthquake Map with Equator")
```




**Problem Set**

## Problem 1

| | Coefficient | Std. error | t-statistic | p-value |
|---|---|---|---|---|
| Intercept | 2.939 | .3119 | 9.42 | <.0001 |
| TV | .046 | .0014 | 32.81 | <.0001 |
| radio | .189 | .0086 | 21.89 | <.0001 |
| newspaper | -.001 | .0059 | -.018 | .8599 |

The first null hypothesis here could be $H_o : B_0 = 0$. This would mean that when $x = 0$, when $x$ is money put into advertising for TV, radio, or newspaper, the number of units sold would be 0. The next row's null hypothesis is $H_0 : \beta_1 = 0$. This means that there is no relationship between the amount of money spent on TV advertising and sales. You could put in \$10,000 for your TV advertisements, and it would not have an impact on sales. Similarly applied to $\beta_2 = 0, \beta_3 = 0$, the null hypothesis for radio and newspaper advertising is there is no relationship between advertising and the sale of the product. However, based on these p values, we see that the probability of $\beta_0 \geq 2.939$ is extremely low if the null hypothesis is indeed true. So if we put in \$0 into all advertising for TV, radio, and newspaper, we would start sales would start at 2.9 units. The probability of $\beta_1 \geq .046$ is also extremely low if the null hypothesis for $\beta_1$ is true as well, so if we put in \$1000 into TV advertising, we should see a 46 unit increase in sales. The probability of $\beta_2 \geq .189$ is also extremely low if the null hypothesis for $\beta_2$ is true as well, so if we put in \$1000 into radio advertising, we should see a 189 unit increase in sales. However, for newspaper, the p-value is pretty high at .8599, so if $\beta_3 = 0$, we should not be surprised to see this coefficient of -.001. We should not include $\beta_3$ in our model.

## Problem 4

a) If the true relationship between X and Y is linear, then the training residual sum of squares for a linear regression would be higher than the residual sum of squares for the cubic regression. The reason for this is because the higher polynomial means that the cubic model is more tailored specifically for the training data. The cubic model captures the twists and turns of the training data, lowering the value of the residuals, better than the linear regression, even if the true relationship is linear.

b) However, for the test RSS, the cubic regression is too fine-tuned for this data. It will see trends where there is none that it got from the training data. It is too variable, and so the test RSS for the cubic regression will be higher than the test RSS for linear regression, since the true relationship is linear.

c) If the true relationship between X and Y is not linear, for the test data, we would see the cubic regression have a lower residual sum of squares than the linear regression. Again, this is because the cubic linear regression will be more sensitive to the data than the linear regression, and since this is the test data, it will be able to trace the data points pretty closely.

d) There isn't really enough information to tell which model will have the lower test RSS, because this depends on the true relationship between X and Y. I could see a case where the linear regression model could have a lower RSS, such as if the true relationship takes the form of a quadratic equation, especially if the $B_2$ term for the $B_2x^2$ is small. However, if the true relationship between X and Y is further from linear, I can see the cubic regression once again having a lower RSS.

## Problem 5

$$\hat{y_i} = x_i\hat{\beta}_i$$

where

$$\hat{\beta} = (\sum_{i=1}^{n} x_i y_i)/(\sum_{i'=1}^{n} x_{i'}^2)$$

$$\hat{y_i} = x_i * (\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i'=1}^{n} x_{i'}^2})$$

$$\hat{y_i} = (\frac{x_i}{\sum_{i'=1}^{n} x_{i'}^2}) * \sum_{i=1}^{n} x_i y_i$$

since we know the values of $x_i$, we can write $\frac{x_i}{\sum_{i'=1}^{n} x_{i'}^2}$ as a constant $p_i$. Then,

$$\hat{y_i} = \sum_{i'=1}^{n} p_{i'} * x_{i'} * y_{i'}$$

$$\hat{y_i} = \sum_{i'=1}^{n} a_{i'} * y_{i'}$$

$$a_{i'} = p_{i'} * x_{i'} = (\frac{x_i}{\sum_{i'=1}^{n} x_{i'}^2}) * x_{i'}$$

$a_{i'}$ is a constant equal to the $x_i^2$ value for each i divided by the sum of each i'th value from 1 to i. We interpret this result by saying the fitted values from linear regression are linear combinations of the response values.

## Additional Exercise

K-nearest neighbor regression defined as: $[\hat{f}(x) = \frac{1}{k}\sum_{x_i \in \mathcal{N}(x)} y_i]$.

$$Test\ MSE = [(y_0 - \hat{f}(x_0)^2)] = Var(\hat{f}) + [E(f - \hat{f})]^2 + Var(\varepsilon)$$

as shown in class. Now, substituting K-nearest neighbor regression into $\hat{f}$:

$$MSE = Var(\frac{1}{k}\sum_{x_i \in \mathcal{N}(x)} y_i) + [E(f - \frac{1}{k}\sum_{x_i \in \mathcal{N}(x)} y_i])]^2 + Var(\varepsilon)$$

$$= \frac{1}{k^2}Var(\sum_{x_i \in \mathcal{N}(x)} y_i) + [f - E(\frac{1}{k}\sum_{x_i \in \mathcal{N}(x)} y_i)]^2 + \sigma^2$$

where sigma squared is a constant.

$$= \frac{1}{k^2}\sum_{x_i \in \mathcal{N}(x)} Var(y_i) + [f - \frac{1}{k}\sum_{x_i \in \mathcal{N}(x)} f(x)]^2 + \sigma^2$$

$$= \frac{1}{k^2}k\sigma^2 + [f - \frac{1}{k}\sum_{x_i \in \mathcal{N}(x)} f(x)]^2 + \sigma^2$$

where $\frac{1}{k}\sum_{x_i \in \mathcal{N}(x)} f(x)$ is constant for a specific neighborhood.

$$= \frac{\sigma^2}{k} + [f - \frac{1}{k}\sum_{x_i \in \mathcal{N}(x)} f(x)]^2 + \sigma^2$$

which gives us $\frac{\sigma^2}{k}$, which is a constant over k, decreasing as k grows larger.

```
x1 <- c(1:3, 5:12)
y1 <- c(-7.1, -7.1, .5, -3.6, -2, -1.7,
        -4, -.2, -1.2, -1.2, -3.5)
df_train <- tibble(x1, y1)

sigma2 <- 1
varfun <- function(k) {
  var1 <- 1/k
  var1
}

vardata <- varfun(1:11)
vardata
```

```
##  [1] 1.00000000 0.50000000 0.33333333 0.25000000 0.20000000 0.16666667
##  [7] 0.14285714 0.12500000 0.11111111 0.10000000 0.09090909
```

```
knn <- function(x, k, training_data) {
  n <- length(x)
  y_hat <- rep(NA, n)
  for (i in 1:n) {
    dists <- abs(x[i] - training_data$x1)
    neighbors <- order(dists)[1:k]
    y_hat[i] <- mean(training_data$y1[neighbors])
  }
  y_hat
```

```
}

bias2funconstant <- function(k) {#x = 5,
  bias2 <- ((-9.3 + (2.6 * 5) - (0.3 * 5^2) + (.01 * 5^3)) - (1/k)*(knn(5,k,train)))^2
  bias2
}
train <- tibble(x1,y1)

bias2funconstant(2)
```

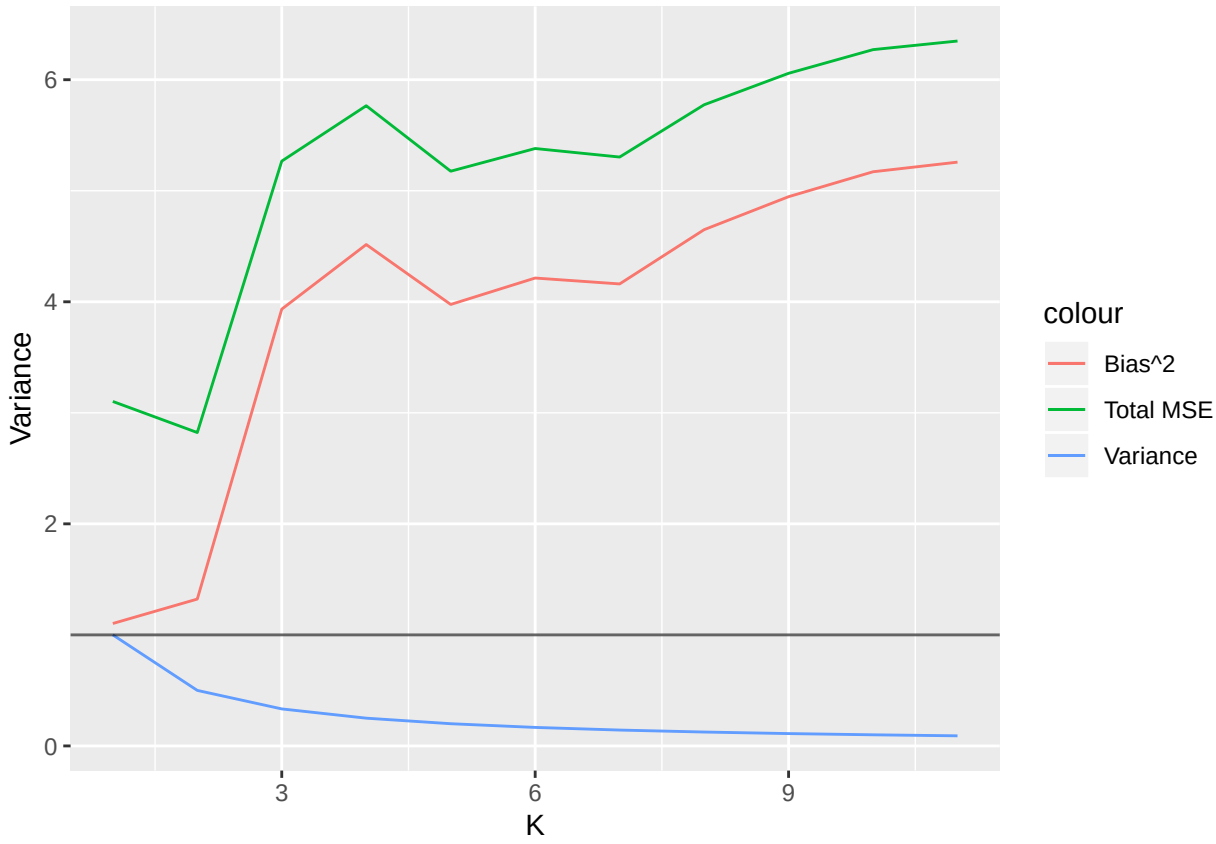
```
## [1] 1.3225
```

```
biasdata2 <- c(bias2funconstant(1),bias2funconstant(2),bias2funconstant(3),bias2funconstant(4),
               bias2funconstant(5),bias2funconstant(6),bias2funconstant(7),bias2funconstant(8),
               bias2funconstant(9),bias2funconstant(10),bias2funconstant(11))
biasdata2
```

```
##  [1] 1.102500 1.322500 3.933611 4.515625 3.976036 4.213897 4.160767
##  [8] 4.649414 4.946505 5.171076 5.257735
```

```
#i chose to write this all about because bias2funconstant(1:11) was not working, the bias2funconstant(2
```

```
kdata <- 1:11

tibbletibble <- tibble(biasdata2, vardata, kdata, x1, y1)
ggplot(data = tibbletibble, aes(x = kdata)) +
  geom_line(data = tibbletibble, aes(x = kdata, y = biasdata2, color = "Bias^2")) +
  geom_line(data = tibbletibble, aes(x = kdata, y = vardata, color = "Variance")) +
  geom_hline(yintercept = 1, alpha = .6) +
  geom_line(data = tibbletibble, aes(x = kdata, y = biasdata2 + vardata + 1, color = "Total MSE"))+
  xlab("K") +
  ylab("Variance")
```


6
4
2
0
Variance
3
6
9
K
colour
Bias^2
Total MSE
Variance