Session 13 - Summary Statistics, Linear Model, and Correlation

Alex Mounsey

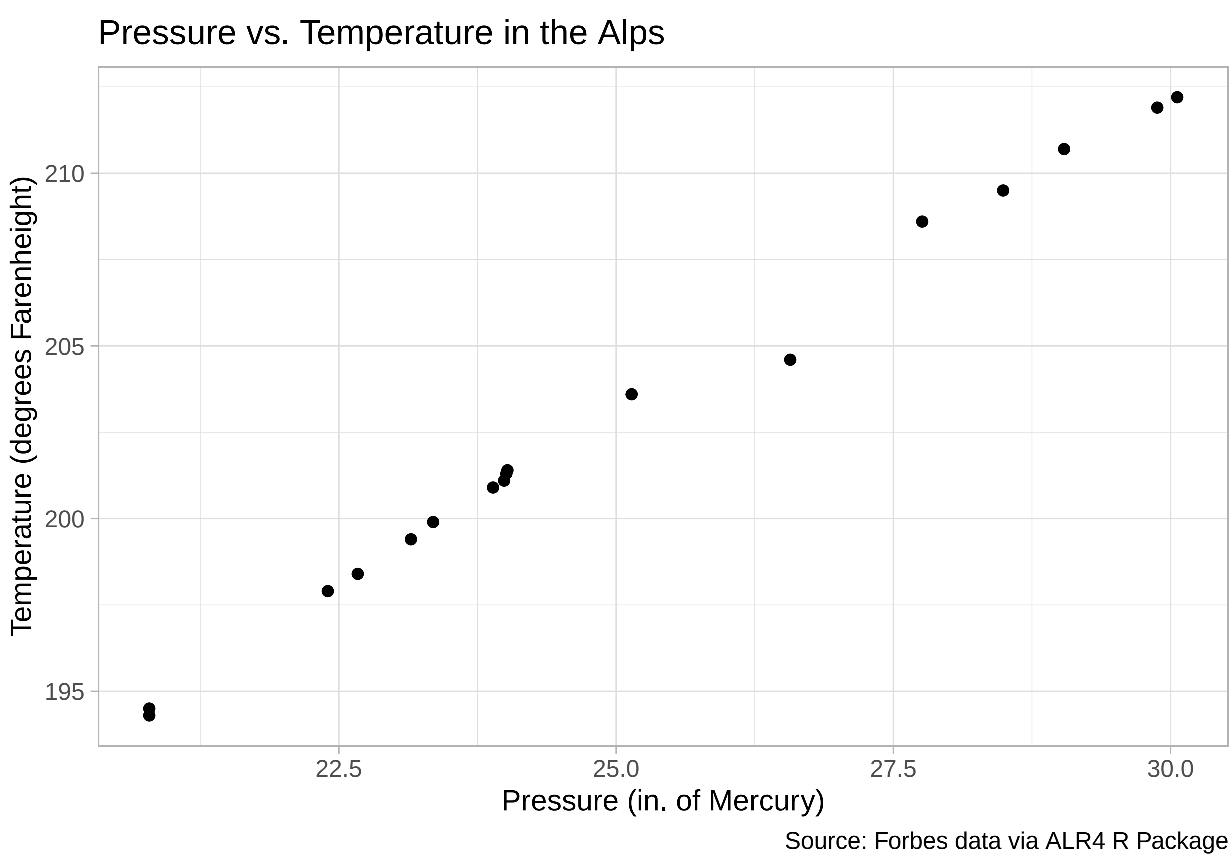
16/11/2020

Exercise 1: Simple Linear Regression

Consider the famous and scientifically important `Forbes` data. This data comprises of 17 pairs of numbers corresponding to the observed boiling point and corrected barometric pressure at locations in the Alps.

Use the `ggplot2` package to plot the data in the `Forbes` dataframe, placing the `pres` variable on the horizontal axis and the `bp` variable on the vertical axis:

```
ggplot(Forbes, aes(x = pres, y = bp)) +
  theme_light() + geom_point() +
  labs(x = "Pressure (in. of Mercury)", y = "Temperature (degrees Farenheight)",
       title = "Pressure vs. Temperature in the Alps",
       caption = "Source: Forbes data via ALR4 R Package")
```



Calculate the mean, median, variance, standard deviation, and interquartile range *(IQR)* for both `pres` and `bp`. Do this using `dplyr` and again using the `with()` function:

```
# Solution with dplyr
Forbes %>%
  summarise(mean_pres = mean(pres), median_pres = median(pres), var_pres = var(pres),
            sd_pres = sd(pres), iqr_pres = IQR(pres), mean_bp = mean(bp),
            median_bp = median(bp), var_bp = var(bp), sd_bp = sd(bp), iqr_bp = IQR(bp))
```

```
##   mean_pres median_pres var_pres  sd_pres iqr_pres  mean_bp median_bp  var_bp
## 1  25.05882       24.01 9.121111 3.020118     4.61 202.9529     201.3 33.1739
##      sd_bp iqr_bp
## 1 5.759679    9.2
```

```
# Solution with 'with()'
mean_pres <- with(Forbes, mean(pres))
```

Fit the Simple Linear Regression Model:

```
m <- lm(bp ~ pres, data = Forbes)
coef(m)
```

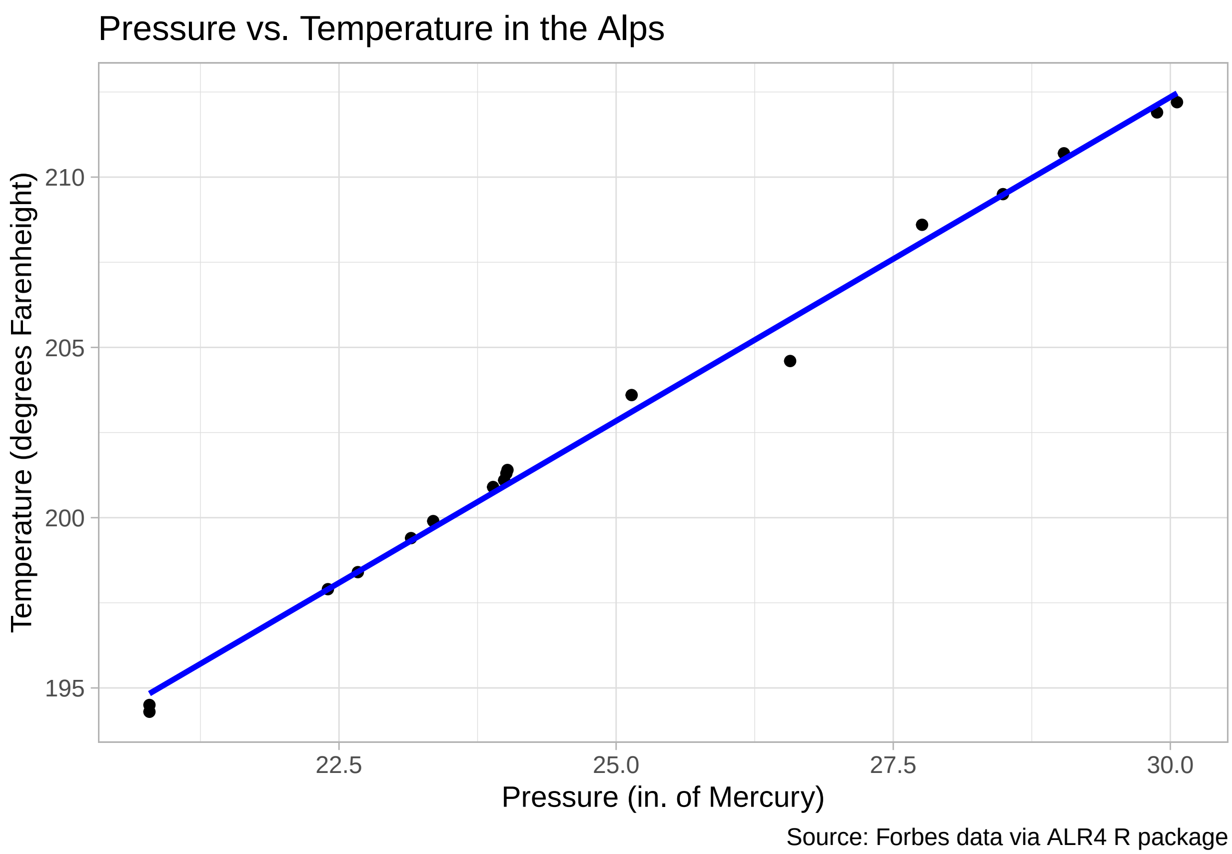
```
## (Intercept)        pres
##  155.296483    1.901784
```

Write down a mathematical expression for the model:

The estimate for α is $\hat{\alpha} = 155.3$ and the estimate for β is $\hat{\beta} = 1.902$. The fitted model is therefore $Temperature = \hat{\alpha} + \hat{\beta}Pressure = 155.3 + 1.902Pressure$.

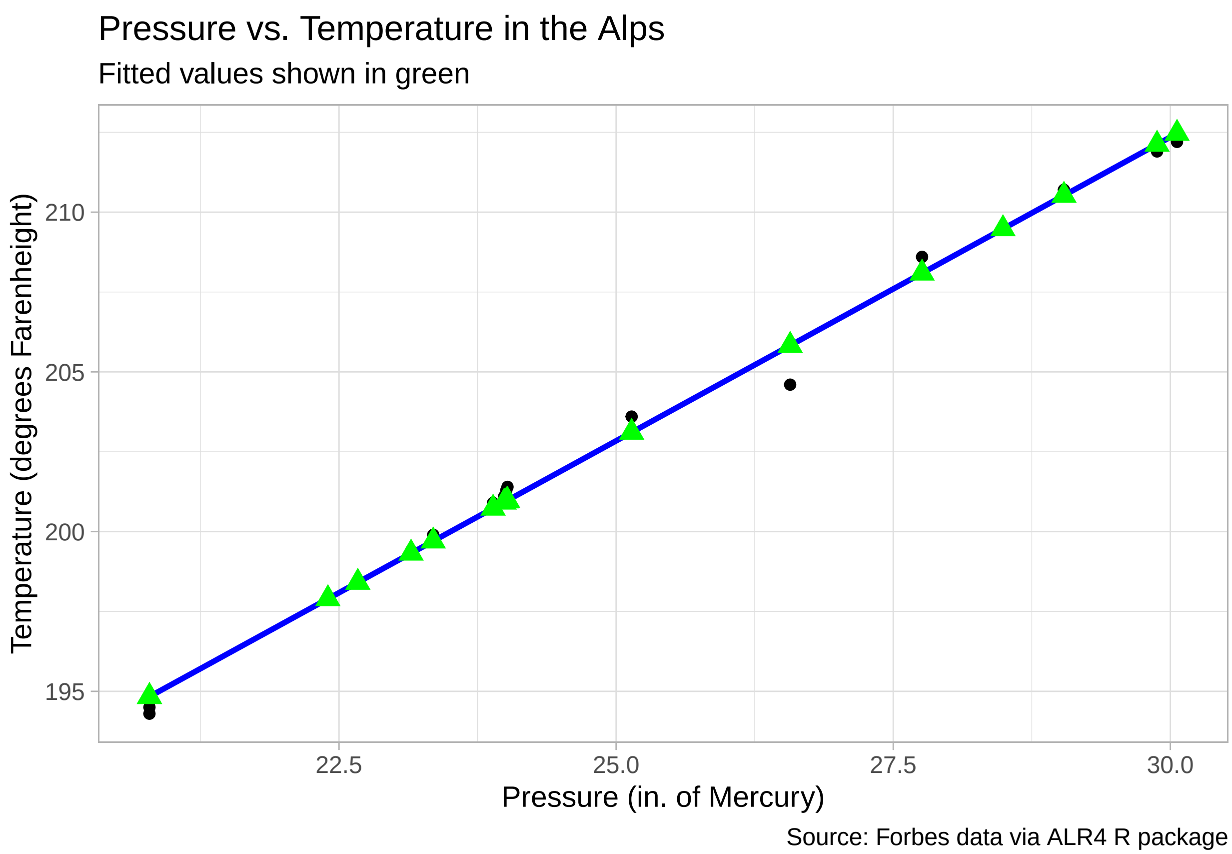
Use the `geom_smooth()` function from `ggplot2` to show, in blue, the fitted line on the plot created above:

```
ggplot(Forbes, aes(x = pres, y = bp)) +
  theme_light() + geom_point() +
  geom_smooth(method = 'lm', se = F, colour = 'blue') +
  labs(x = "Pressure (in. of Mercury)", y = "Temperature (degrees Farenheight)",
       title = "Pressure vs. Temperature in the Alps",
       caption = "Source: Forbes data via ALR4 R package")
```



Use the function `add_predictions()` from `modelr` to compute the fitted values and include them with the original data. Add these values to your plot in green:

```
Forbes %>%
  add_predictions(m) %>%
  ggplot(aes(x = pres, y = bp)) +
    theme_light() + geom_point() +
    geom_smooth(method = 'lm', se = F, colour = 'blue') +
    geom_point(aes(y = pred), col = 'green', pch = 17, size = 3) +
    labs(x = "Pressure (in. of Mercury)", y = "Temperature (degrees Farenheight)",
         title = "Pressure vs. Temperature in the Alps",
         subtitle = "Fitted values shown in green",
         caption = "Source: Forbes data via ALR4 R package")
```



Is there an underlying relationship between temperature and pressure? Justify your conclusion:

```r
p_value <- signif(summary(m)$coefficients[2,4], 3)
summary(m)
```

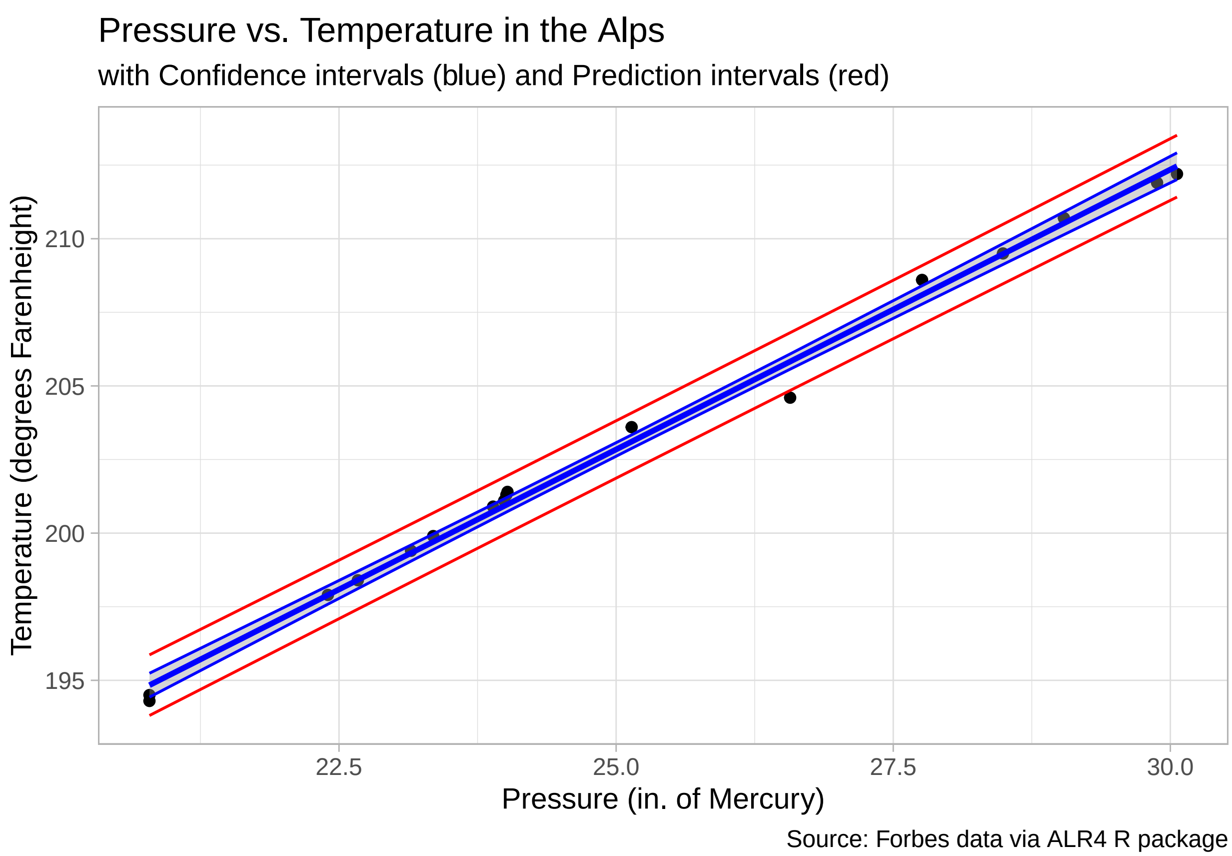
```
##
## Call:
## lm(formula = bp ~ pres, data = Forbes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.22687 -0.22178  0.07723  0.19687  0.51001
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 155.29648    0.92734  167.47   <2e-16 ***
## pres          1.90178    0.03676   51.74   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.444 on 15 degrees of freedom
## Multiple R-squared:  0.9944, Adjusted R-squared:  0.9941
## F-statistic:  2677 on 1 and 15 DF,  p-value: < 2.2e-16
```

The `p-value` is equal to 2.53×10^{-18}, which is less than `0.05` - therefore we reject the null hypothesis

$H_0 : \beta = 0$ *(there is no underlying relationship)* in favour of the alternative hypothesis $H_1 : \beta \neq 0$ *(there is an underlying relationship)*.

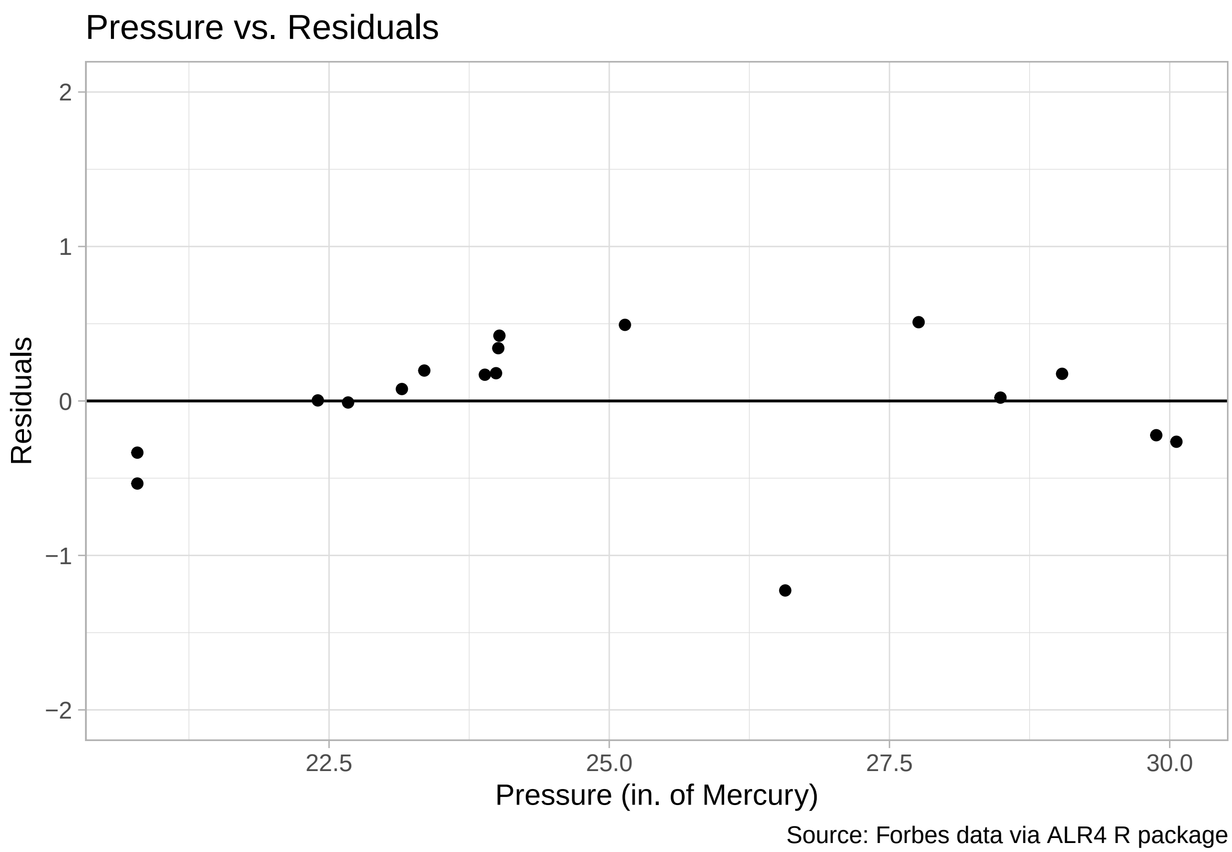
Add the confidence and prediction intervals to your graph:

```
confidence <- data.frame(Forbes, predict(m, interval = 'confidence'))
prediction <- data.frame(Forbes, predict(m, interval = 'prediction'))

Forbes %>%
  add_predictions(m) %>%
  ggplot(aes(x = pres, y = bp)) +
    theme_light() + geom_point() +
    geom_smooth(method = 'lm', se = T, colour = 'blue') +
    geom_line(data = confidence, aes(y = lwr), colour = 'blue') +
    geom_line(data = confidence, aes(y = upr), colour = 'blue') +
    geom_line(data = prediction, aes(y = lwr), colour = 'red') +
    geom_line(data = prediction, aes(y = upr), colour = 'red') +
      labs(x = "Pressure (in. of Mercury)", y = "Temperature (degrees Farenheight)",
           title = "Pressure vs. Temperature in the Alps",
           subtitle = "with Confidence intervals (blue) and Prediction intervals (red)",
           caption = "Source: Forbes data via ALR4 R package")
```

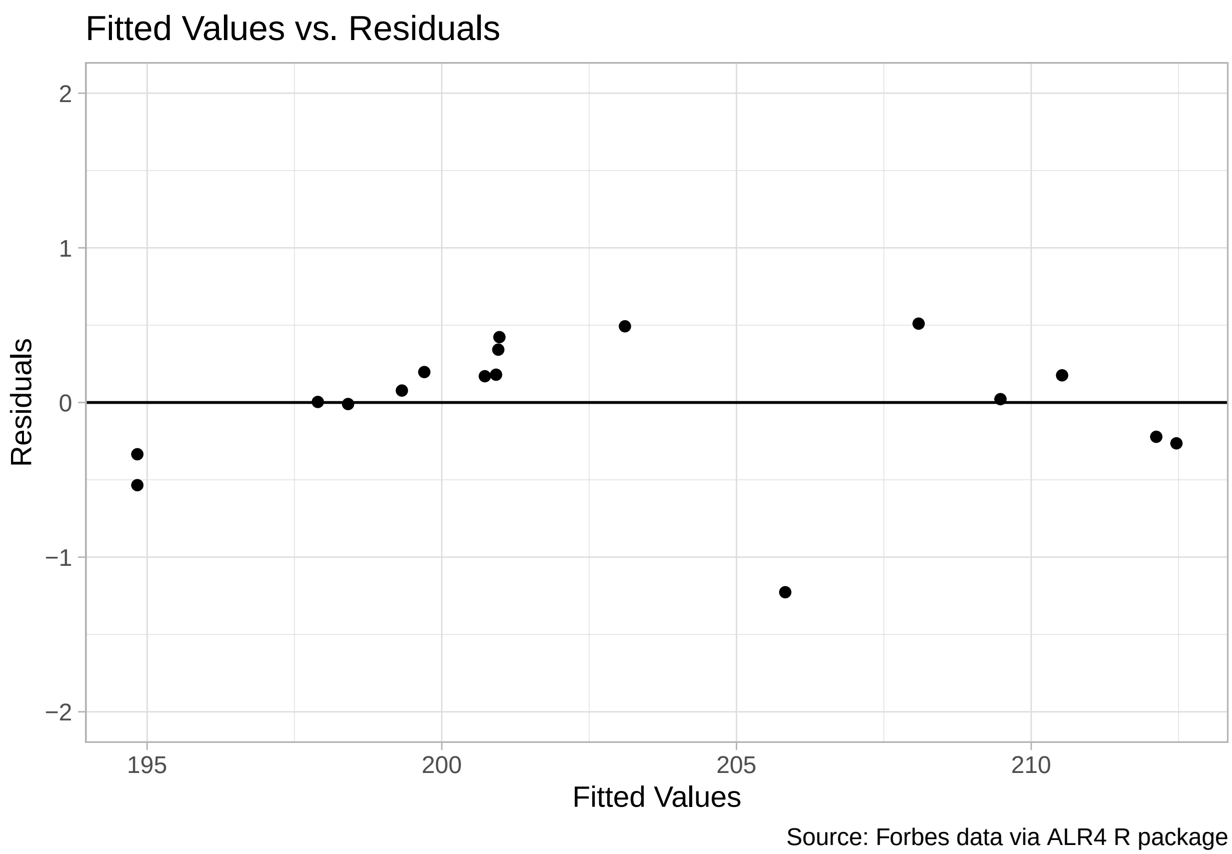


Using the `add_residuals()` function from `modelr`, compute the residuals and include them with the original data and fitted values:

```r
Forbes %>%
  add_predictions(m) %>%
  add_residuals(m) %>%
  ggplot(aes(x = pres, y = resid)) +
    theme_light() + geom_point() +
    scale_y_continuous(limits = c(-2, 2)) +
    geom_hline(aes(yintercept = 0)) +
    labs(x = "Pressure (in. of Mercury)", y = "Residuals",
         title = "Pressure vs. Residuals",
         caption = "Source: Forbes data via ALR4 R package")
```




```r
Forbes %>%
  add_predictions(m) %>%
  add_residuals(m) %>%
  ggplot(aes(x = pred, y = resid)) +
    theme_light() + geom_point() +
    scale_y_continuous(limits = c(-2, 2)) +
    geom_hline(aes(yintercept = 0)) +
    labs(x = "Fitted Values", y = "Residuals",
         title = "Fitted Values vs. Residuals",
         caption = "Source: Forbes data via ALR4 R package")
```



What is the correlation coefficient between temperature and pressure? Use a test of correlation to determine whether there is an underlying dependency between temperature and pressure:

```
p_value <- with(Forbes, cor(pres, bp))
with(Forbes, cor.test(pres, bp))
```

```
##
##  Pearson's product-moment correlation
##
## data:  pres and bp
## t = 51.741, df = 15, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9920669 0.9990205
## sample estimates:
##       cor
## 0.9972102
```

Once again, the `p-value` is below `0.05` (0.9972102). This prompts us to reject the null hypothesis $H_0 : \rho = 0$ in favor of the alternative hypothesis $H_1 : \rho \neq 0$. Therefore, we can conclude that there *is* an underlying linear dependency between temperature and pressure.

Exercise 2: Multiple Regression

The aim of this exercise is to extend the simple linear regression model:

$$y = \alpha + \beta x + error$$

which has one explanatory variable (x) to the multiple linear regression model:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + error$$

which has more than one explanatory variable (here: x_1 and x_2).

Here is a dataset which will help us understand the multiple linear regression model. The data comes from shops in ten towns:

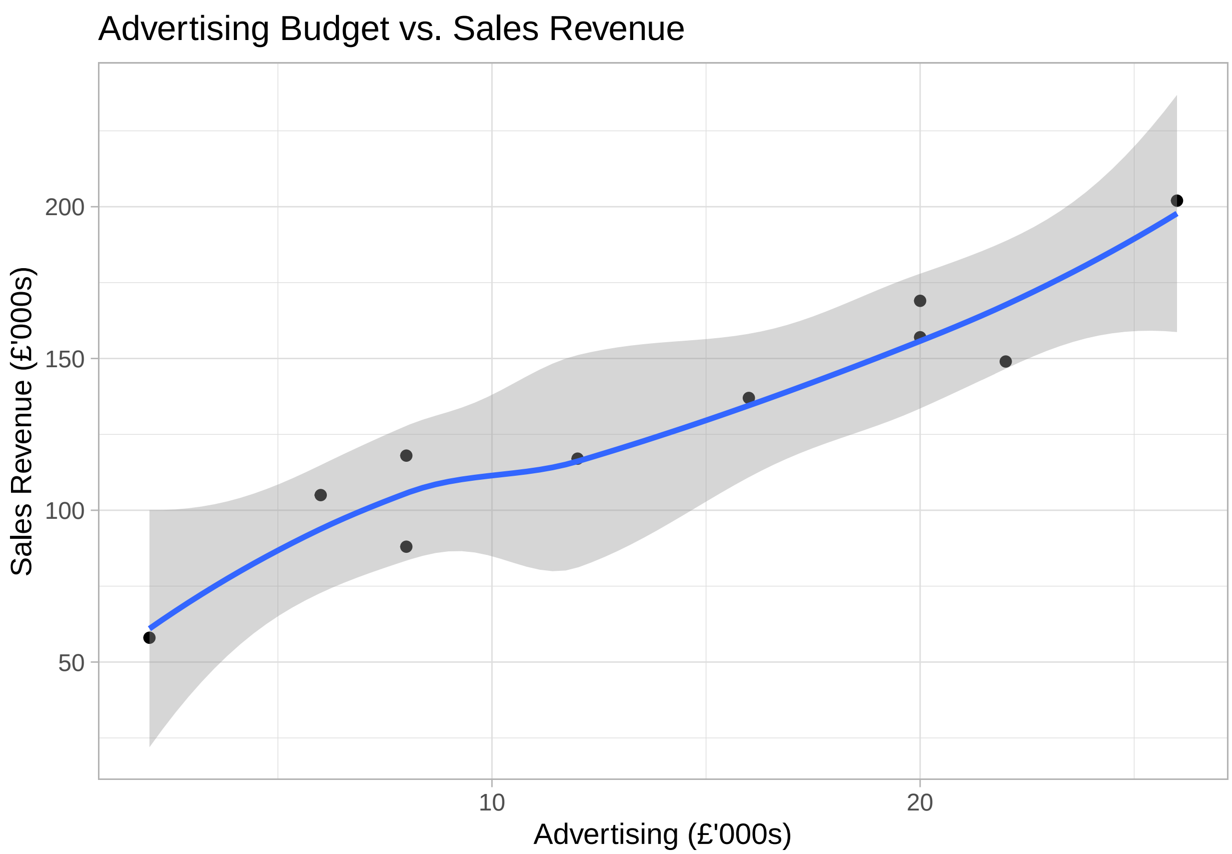
Town	Advertising (£000s)	Population (000s)	Annual Sales (£000s)
1	2	41	58
2	6	93	105
3	8	78	88
4	8	102	118
5	12	96	117
6	16	112	137
7	20	119	157
8	20	131	169
9	22	118	149
10	26	185	202

Input this data:

```
town <- 1:10
advertising <- c(2, 6, 8, 8, 12, 16, 20, 20, 22, 26)
population <- c(41, 93, 78, 102, 96, 112, 119, 131, 118, 185)
sales <- c(58, 105, 88, 118, 117, 137, 157, 169, 149, 202)

shops <- data.frame(town, advertising, population, sales)
```

Plot the annual sales against advertising, illustrating the dependence of sales on advertising, using a smooth curve:

```
ggplot(shops, aes(x = advertising, y = sales)) +
  theme_light() + geom_point() + geom_smooth() +
  labs(x = "Advertising (£'000s)", y = "Sales Revenue (£'000s)",
       title = "Advertising Budget vs. Sales Revenue")
```



Plot annual sales against population, illustrating the dependence of sales on population, using a smooth curve:

```
ggplot(shops, aes(x = population, y = sales)) +
  theme_light() + geom_point() + geom_smooth() +
  labs(x = "Population (000s)", y = "Sales Revenue (£'000s)",
       title = "Population vs. Sales Revenue")
```

In light of the linear relationships observed above, we can fit the model:

$$sales = \alpha + \beta_1 advertising + \beta_2 population + error$$

Which can be implemented in R with the following code:

```
m <- lm(sales ~ advertising + population, data = shops)
p_value <- signif(summary(m)$coefficients[2,4], 3)
summary(m)
```

```
##
## Call:
## lm(formula = sales ~ advertising + population, data = shops)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9777 -3.9884 -0.5326  3.7399 10.1376
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  26.1393     8.0014   3.267  0.01373 *
## advertising   2.0949     0.6338   3.305  0.01302 *
## population    0.6933     0.1352   5.129  0.00135 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 6.777 on 7 degrees of freedom
## Multiple R-squared:  0.9796, Adjusted R-squared:  0.9737
## F-statistic: 167.7 on 2 and 7 DF,  p-value: 1.221e-06
```

The second `p-value` (0.013) can be used to check whether β_1 is significantly different from zero. If this `p-value` is less than `0.05`, we can conclude that advertising does have an effect on sales for all similar shops.

We can make predictions in the usual way; for example, to predict sales if `£10,000` is spent on advertising in a town with a population of `140,000`:

```r
predict(m, newdata = data.frame(advertising = 10, population = 140))
```

```
##        1
## 144.1537
```

Produce a prediction for sales, together with a confidence and prediction interval, if `£10,000` is spent on advertising in a town of `140,000` people and if `£30,000` is spent on advertising in a town of `200,000` people:

```r
# Confidence intervals
predict(m, newdata = data.frame(advertising = c(10, 30), population = c(140, 200)),
        interval = 'confidence')
```

```
##        fit      lwr      upr
## 1 144.1537 127.3959 160.9115
## 2 227.6505 213.2405 242.0606
```

```r
# Prediction intervals
predict(m, newdata = data.frame(advertising = c(10, 30), population = c(140, 200)),
        interval = 'prediction')
```

```
##        fit      lwr      upr
## 1 144.1537 120.9663 167.3411
## 2 227.6505 206.0987 249.2024
```

Exercise 3: Multiple Regression

The data below shows the loading times of cargo in relation to weight, volume, and the number of items in the shipment:

Cargo	Weight	Volume	No. of Items	Loading Time
1	0.4	53	158	64
2	0.4	23	163	60
3	3.1	19	37	71
4	0.6	34	157	61
5	4.7	24	59	54
6	1.7	65	123	77
7	9.4	44	46	81
8	10.1	31	117	93
9	11.6	29	173	93

Cargo	Weight	Volume	No. of Items	Loading Time
10	12.6	58	112	51
11	10.9	37	111	76
12	23.1	46	114	96
13	23.1	50	134	77
14	21.6	44	73	93
15	23.1	56	168	95
16	1.9	36	143	54
17	26.8	58	202	168
18	29.9	51	124	99

Input the data:

```
cargo <- 1:18
weight <- c(0.4, 0.4, 3.1, 0.6, 4.7, 1.7, 9.4, 10.1, 11.6, 12.6, 10.9, 23.1, 23.1, 21.6, 23.1, 1.9, 26.8
volume <- c(53, 23, 19, 34, 24, 65, 44, 31, 29, 58, 37, 46, 50, 44, 56, 36, 58, 51)
n_items <- c(158, 163, 37, 157, 59, 123, 46, 117, 173, 112, 111, 114, 134, 73, 168, 143, 202, 124)
loading_time <- c(64, 60, 71, 61, 54, 77, 81, 93, 93, 51, 76, 96, 77, 93, 95, 54, 168, 99)

shipping <- data.frame(cargo, weight, volume, n_items, loading_time)
```

Fit the model:

$$LoadingTime = \alpha\beta_1 Weight + \beta_2 Volume + \beta_3 NumberofItems + error$$

```
m <- lm(loading_time ~ weight + volume + n_items, data = shipping)
coef(m)
```

```
## (Intercept)      weight      volume     n_items
## 43.65219779  1.78477968 -0.08339706  0.16113269
```

Which variable has the highest p-value?

```
summary(m)
```

```
##
## Call:
## lm(formula = loading_time ~ weight + volume + n_items, data = shipping)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28.35 -11.39  -2.66  12.09  48.80
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  43.6522    18.0102   2.424  0.02949 *
## weight        1.7848     0.5377   3.319  0.00506 **
## volume       -0.0834     0.4177  -0.200  0.84462
## n_items       0.1611     0.1117   1.443  0.17102
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.97 on 14 degrees of freedom
## Multiple R-squared:  0.5493, Adjusted R-squared:  0.4528
## F-statistic: 5.689 on 3 and 14 DF,  p-value: 0.009224
```

Remove the variable with the highest p-value from the model, provided that the p-value is greater than 0.05:

```r
m <- lm(loading_time ~ weight + n_items, data = shipping)
coef(m)
```

```
## (Intercept)      weight     n_items
##   41.479364    1.737438    0.154843
```

Which variable has the highest p-value?

```r
summary(m)
```

```
##
## Call:
## lm(formula = loading_time ~ weight + n_items, data = shipping)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.713 -11.324  -2.953  11.286  48.679
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.4794    13.8834   2.988  0.00920 **
## weight        1.7374     0.4669   3.721  0.00205 **
## n_items       0.1548     0.1036   1.494  0.15592
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.32 on 15 degrees of freedom
## Multiple R-squared:  0.5481, Adjusted R-squared:  0.4878
## F-statistic: 9.095 on 2 and 15 DF,  p-value: 0.002589
```

Remove the variable with the highest p-value from the model, provided that the p-value is greater than 0.05:

```r
m <- lm(loading_time ~ weight, data = shipping)
coef(m)
```

```
## (Intercept)      weight
##   59.258959    1.843436
```

Does weight have an effect on loading time for all cargo?

```r
summary(m)
```

```
##
## Call:
## lm(formula = loading_time ~ weight, data = shipping)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.486  -8.282  -1.674   5.623  59.337
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  59.2590     7.4200   7.986 5.67e-07 ***
## weight        1.8434     0.4789   3.849  0.00142 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.05 on 16 degrees of freedom
## Multiple R-squared:  0.4808, Adjusted R-squared:  0.4484
## F-statistic: 14.82 on 1 and 16 DF,  p-value: 0.001417
```

The procedure above is called *backward elimination*, and has simplified the original model:

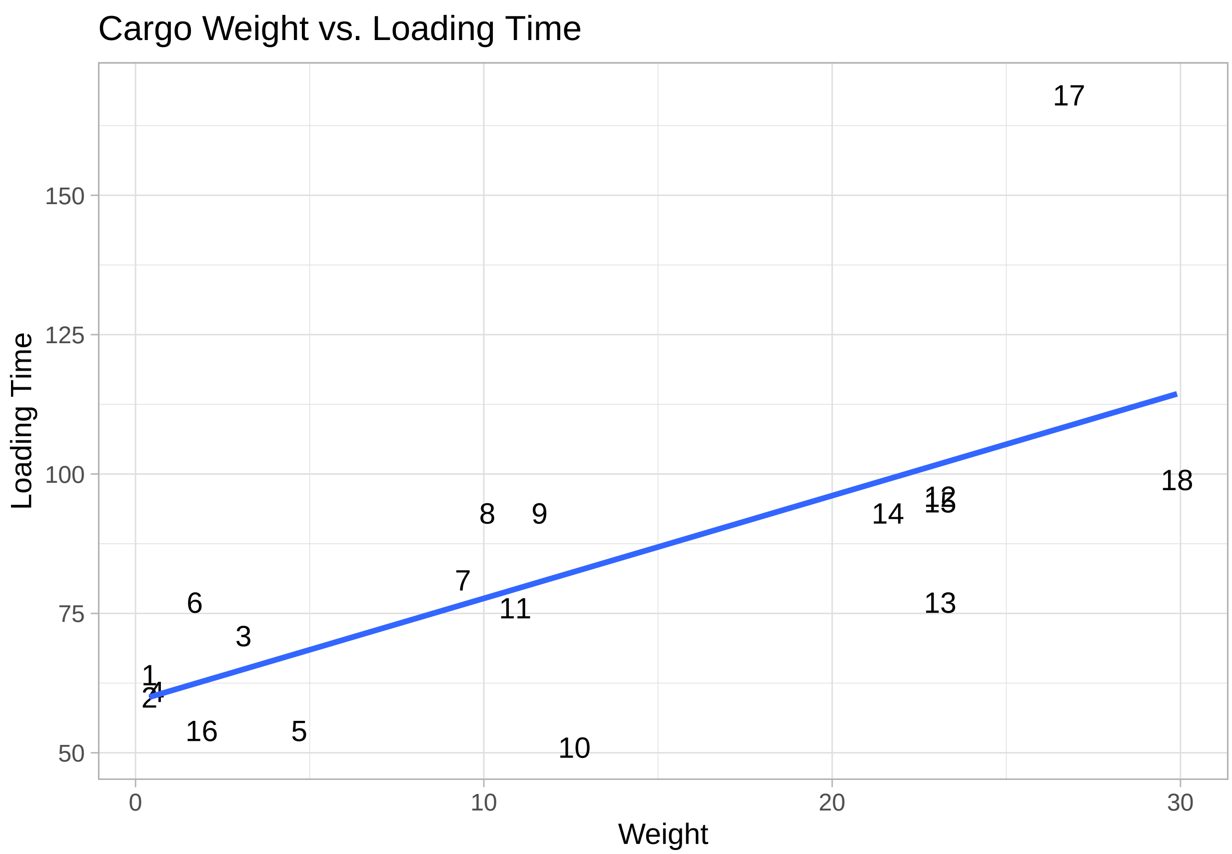
$$LoadingTime = \alpha + \beta_1 Weight + \beta_2 Volume + \beta_3 NumberofItems + error$$

to:

$$LoadingTime = \alpha + \beta Weight + error$$

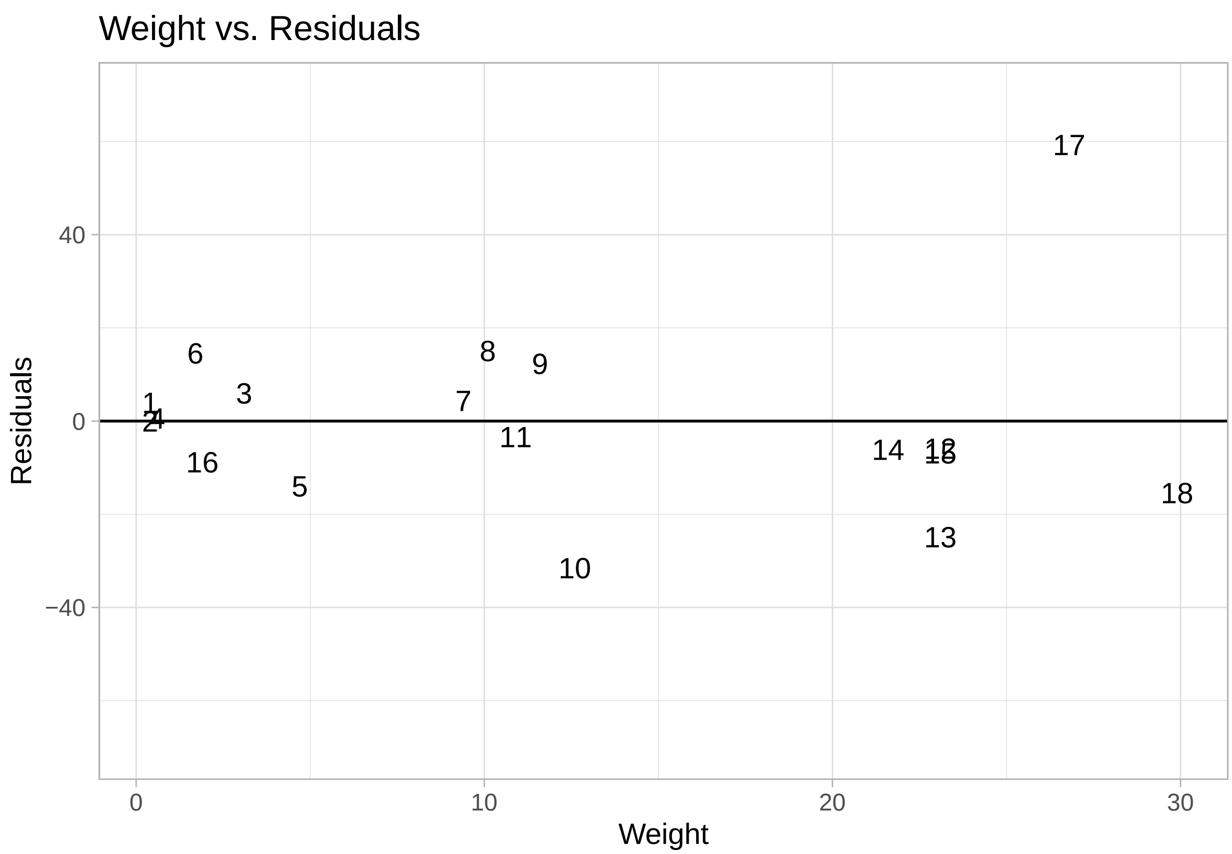
Use `ggplot2` to plot `weight` vs. `loading_time`, with the cargo number as the plotting characters:

```r
ggplot(shipping, aes(x = weight, y = loading_time, label = cargo)) +
  theme_light() + geom_text() +
  geom_smooth(method = 'lm', se = FALSE) +
  labs(x = "Weight", y = "Loading Time",
       title = "Cargo Weight vs. Loading Time")
```

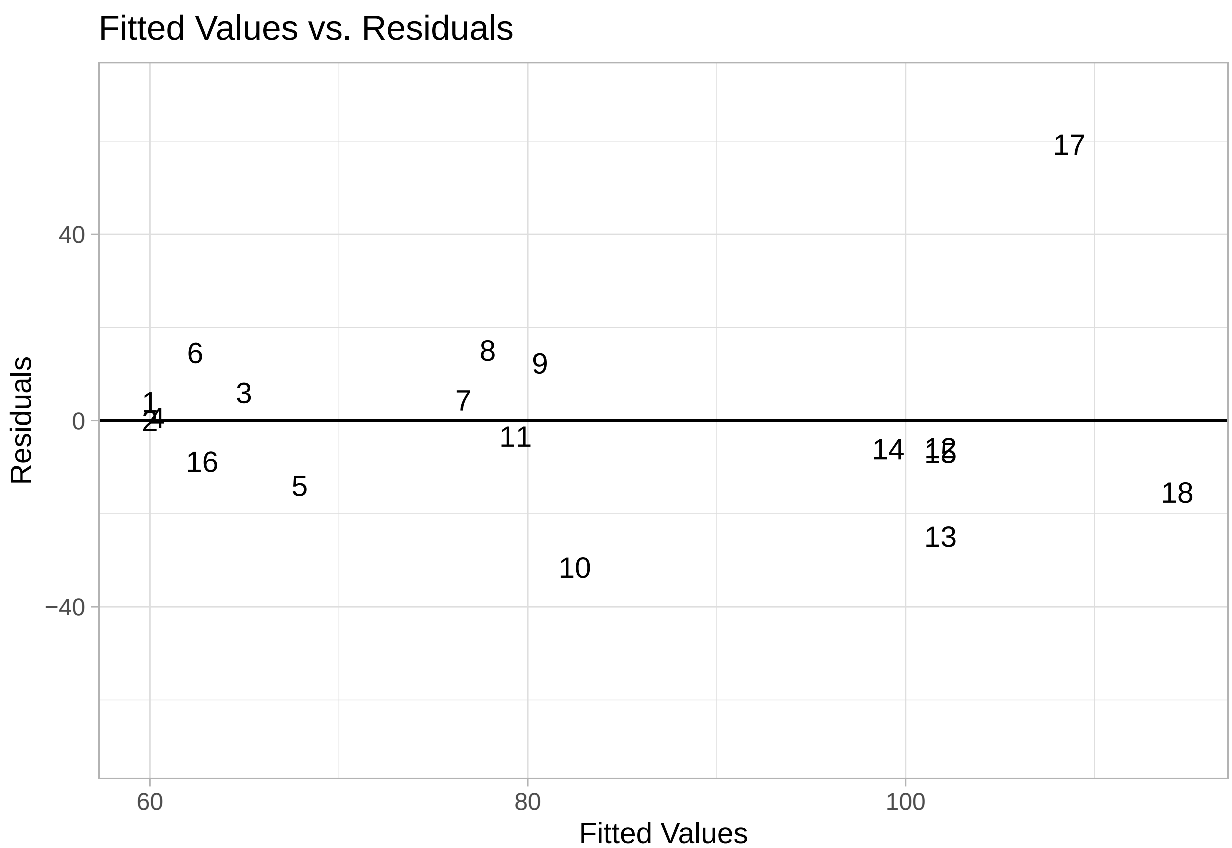


Produce residual plots:

```
shipping %>%
  add_predictions(m) %>%
  add_residuals(m) %>%
  ggplot(aes(x = weight, y = resid, label = cargo)) +
    theme_light() + geom_text() +
    scale_y_continuous(limits = c(-70, 70)) +
    geom_hline(aes(yintercept = 0)) +
    labs(x = "Weight", y = "Residuals",
         title = "Weight vs. Residuals")
```




```r
shipping %>%
  add_predictions(m) %>%
  add_residuals(m) %>%
  ggplot(aes(x = pred, y = resid, label = cargo)) +
    theme_light() + geom_text() +
    scale_y_continuous(limits = c(-70, 70)) +
    geom_hline(aes(yintercept = 0)) +
    labs(x = "Fitted Values", y = "Residuals",
         title = "Fitted Values vs. Residuals")
```

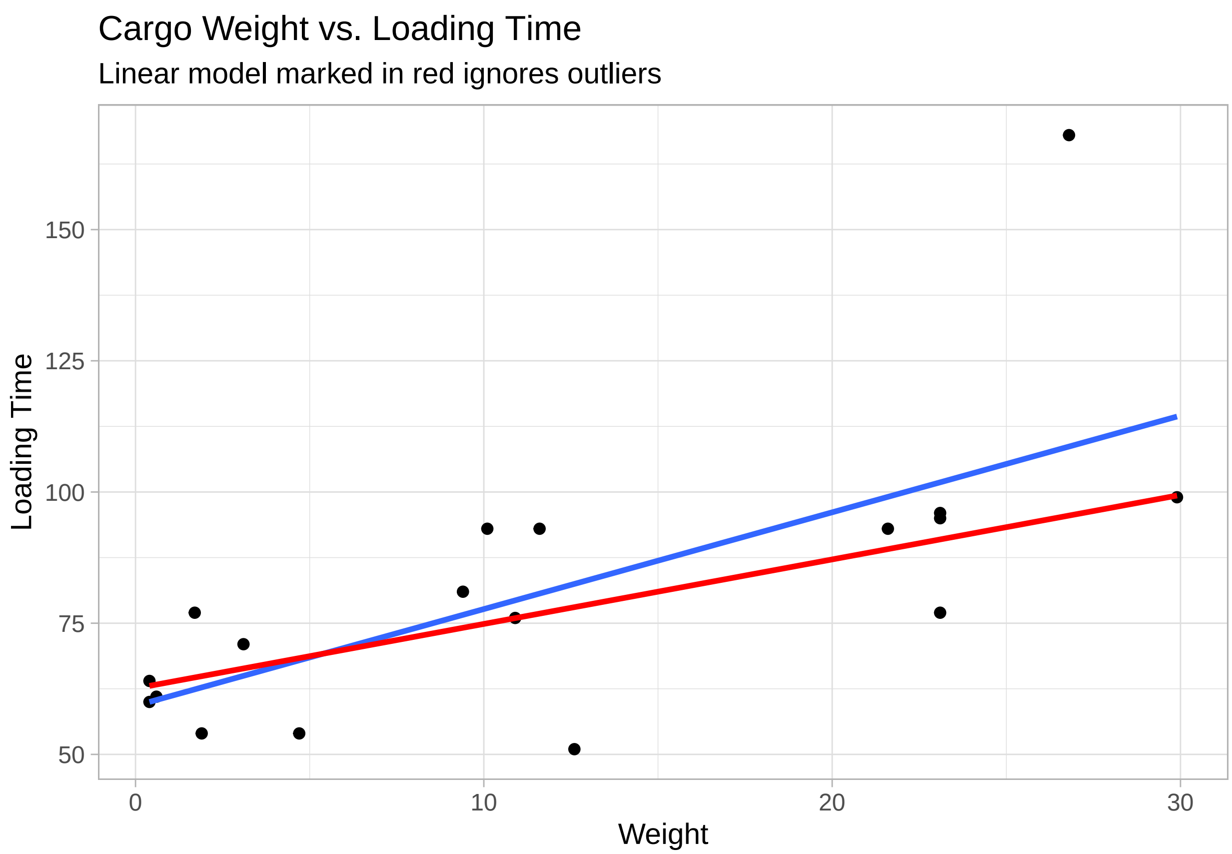


Comment on how well the simple linear regression model fits the data:

The residuals appear to be evenly scattered about `0`, with no distinct pattern. This indicates that the variance is constant and the simple linear regression fits the data well.

Add, in red, the simple linear regression line based on all data except the `17th` data point *(the outlier)*:

```
ggplot(shipping, aes(x = weight, y = loading_time)) +
  theme_light() + geom_point() +
  geom_smooth(method = 'lm', se = F) +
  geom_smooth(data = shipping[-17,], method = 'lm', se = F, colour = 'red') +
  labs(x = "Weight", y = "Loading Time",
       title = "Cargo Weight vs. Loading Time",
       subtitle = "Linear model marked in red ignores outliers")
```



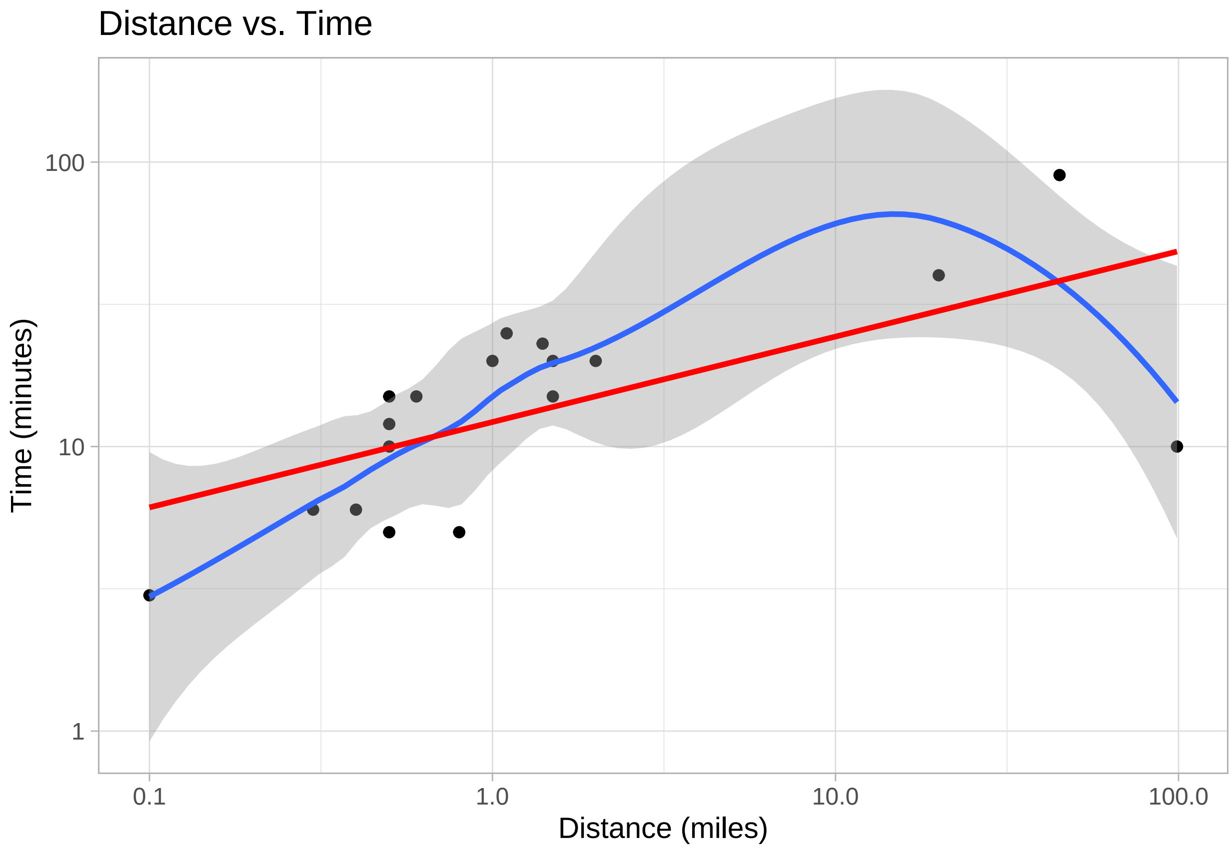
What is the effect of removing the `17th` data point?

The linear model which excludes the `17th` data point in its calculation appears to fit the data more appropriately, and should produce more accurate predictions.

Exercise 4: Transformations

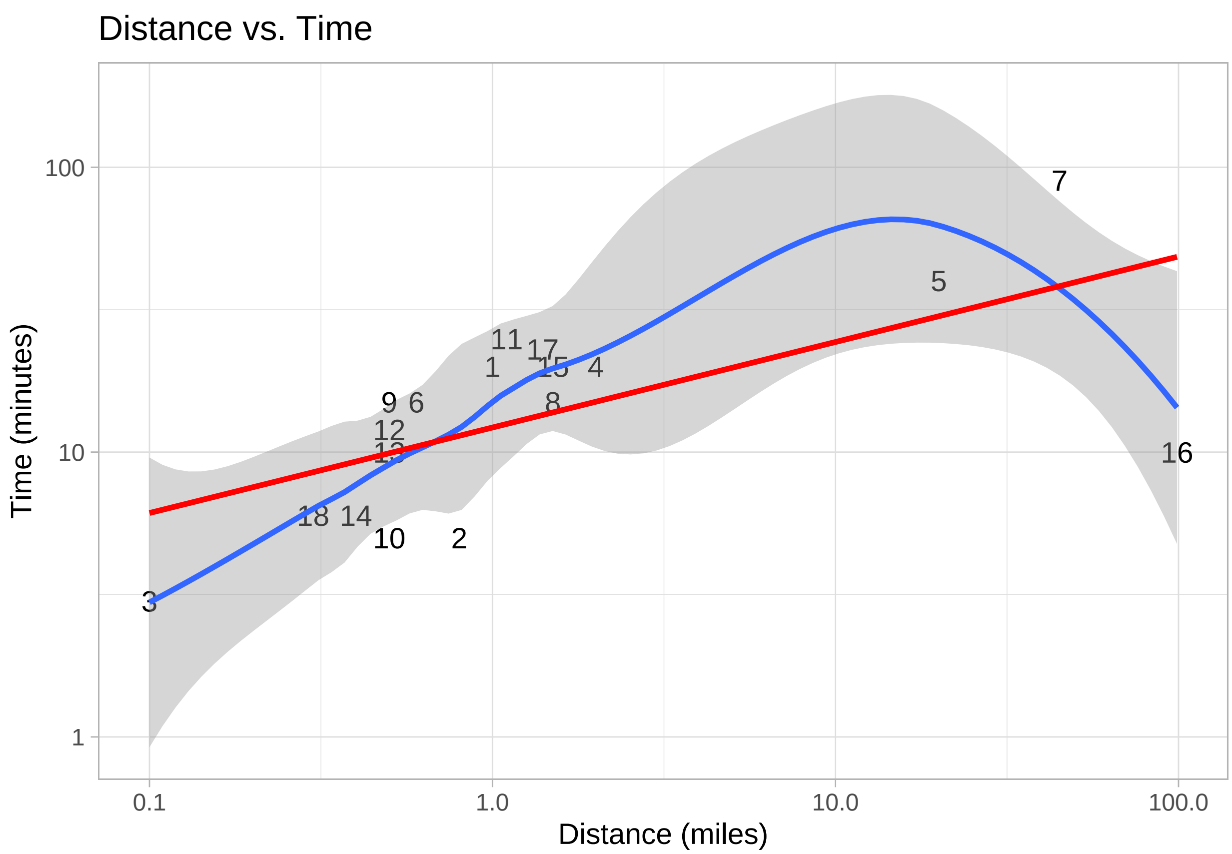
In previous sessions, we have worked with the questionnaire data. In particular, we used this code to produce the following plot:

```
q_data <- read_csv('../data/MATH513_Questionnaire_Data.csv')

ggplot(q_data, aes(x = Distance, y = Travel_time)) +
  theme_light() + geom_point() + geom_smooth() +
  geom_smooth(method = 'lm', se = F, colour = 'red') +
  scale_x_log10() + scale_y_log10() +
  labs(x = "Distance (miles)", y = "Time (minutes)",
       title = "Distance vs. Time")
```



Add labels for each of the data points' index, and use them as plotting characters:

```
q_data <- q_data %>%
  mutate(data_point = 1:nrow(q_data))

ggplot(q_data, aes(x = Distance, y = Travel_time, label = data_point)) +
  theme_light() + geom_text() + geom_smooth() +
  geom_smooth(method = 'lm', se = F, colour = 'red') +
  scale_x_log10() + scale_y_log10() +
  labs(x = "Distance (miles)", y = "Time (minutes)",
       title = "Distance vs. Time")
```



Fit the model:

$$log(TravelTime = \beta_0 + \beta_1 log(Distance) + error)$$

Note that as it's not possible to compute the log of zero, we add a small value (`0.01`) to `Distance` in the case that there are some zero distances.

```r
m_log <- lm(log(Travel_time) ~ log(Distance + 0.01), data = q_data)
summary(m_log)
```

```
##
## Call:
## lm(formula = log(Travel_time) ~ log(Distance + 0.01), data = q_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5771 -0.4146  0.2407  0.4051  0.8575
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.49588    0.15726  15.871 3.27e-11 ***
## log(Distance + 0.01)  0.30115    0.09023   3.338  0.00417 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.6561 on 16 degrees of freedom
## Multiple R-squared:  0.4105, Adjusted R-squared:  0.3736
## F-statistic: 11.14 on 1 and 16 DF,  p-value: 0.004174
```

Re-fit the model, omitting the `16th` data point:

```
m_log <- lm(log(Travel_time) ~ log(Distance + 0.01), data = q_data, subset = -16)
```

Use this model to produce confidence intervals for β_0 and β_1 using the `cofint()` function:

```
confint(m_log)
```

```
##                          2.5 %   97.5 %
## (Intercept)          2.353922 2.795413
## log(Distance + 0.01) 0.357343 0.671532
```

These intervals give us an indication of the reliability of the estimates for β_0 and β_1. The wider the interval, the less reliable the estimate is.

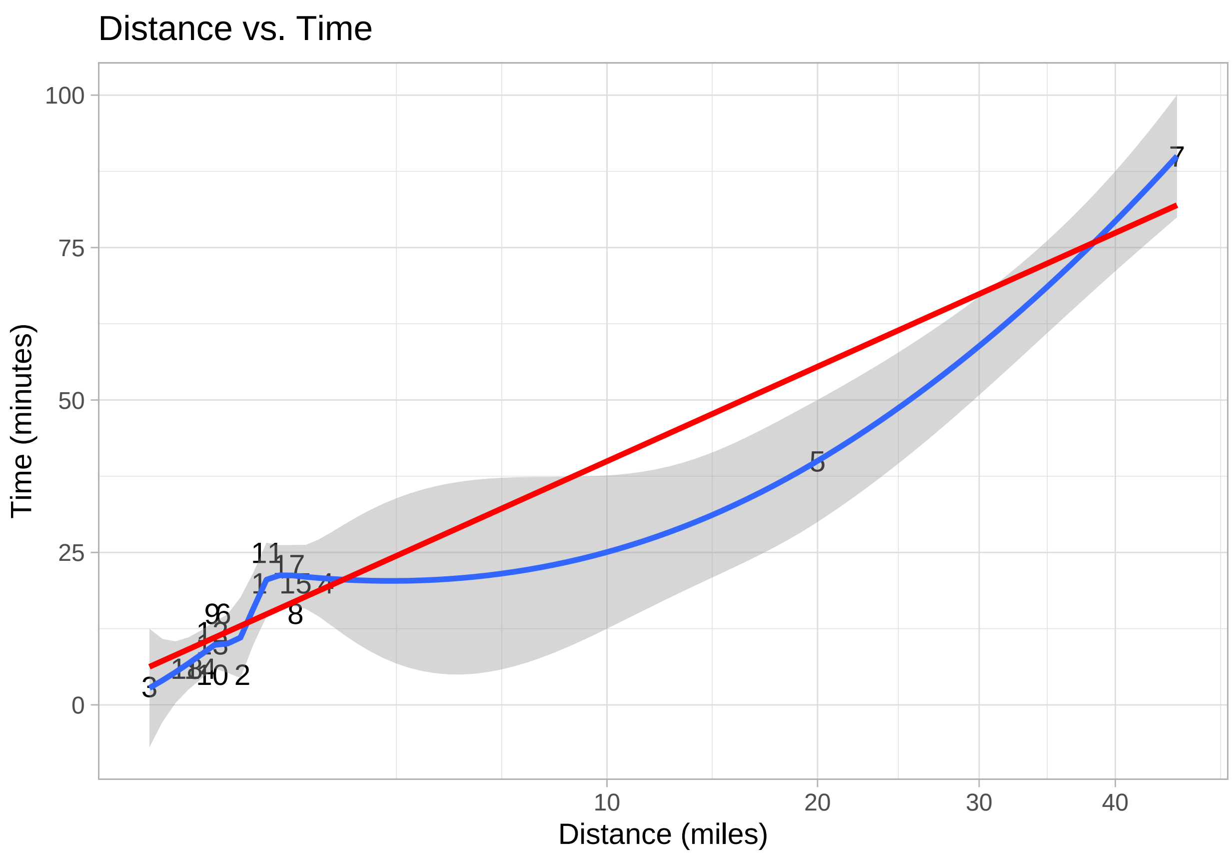
A mathematical argument that we do not discuss here tells us that the second of these confidence intervals, here *(0.357343, 0.671532)*, contains `0.5`, then the travel time may depend on the square root of the distance.

To check this out, we should plot the data using a square root scale on the `x`-axis, with a standard linear scale on the `y`-axis. A square root scale spreads out values less than 1 and squashes values greater than 1.

In `ggplot2`, a square root scale on the `x`-axis can be produced using `scale_x_sqrt()` instead of `scale_x_log10()`.

Produce a plot which uses a square root scale on the x-axis and a standard linear scale on the y-axis:

```
ggplot(q_data[-16,], aes(x = Distance, y = Travel_time, label = data_point)) +
  theme_light() + geom_text() + geom_smooth() +
  geom_smooth(method = 'lm', se = F, col = 'red') +
  scale_x_sqrt() +
  labs(x = "Distance (miles)", y = "Time (minutes)",
       title = "Distance vs. Time")
```

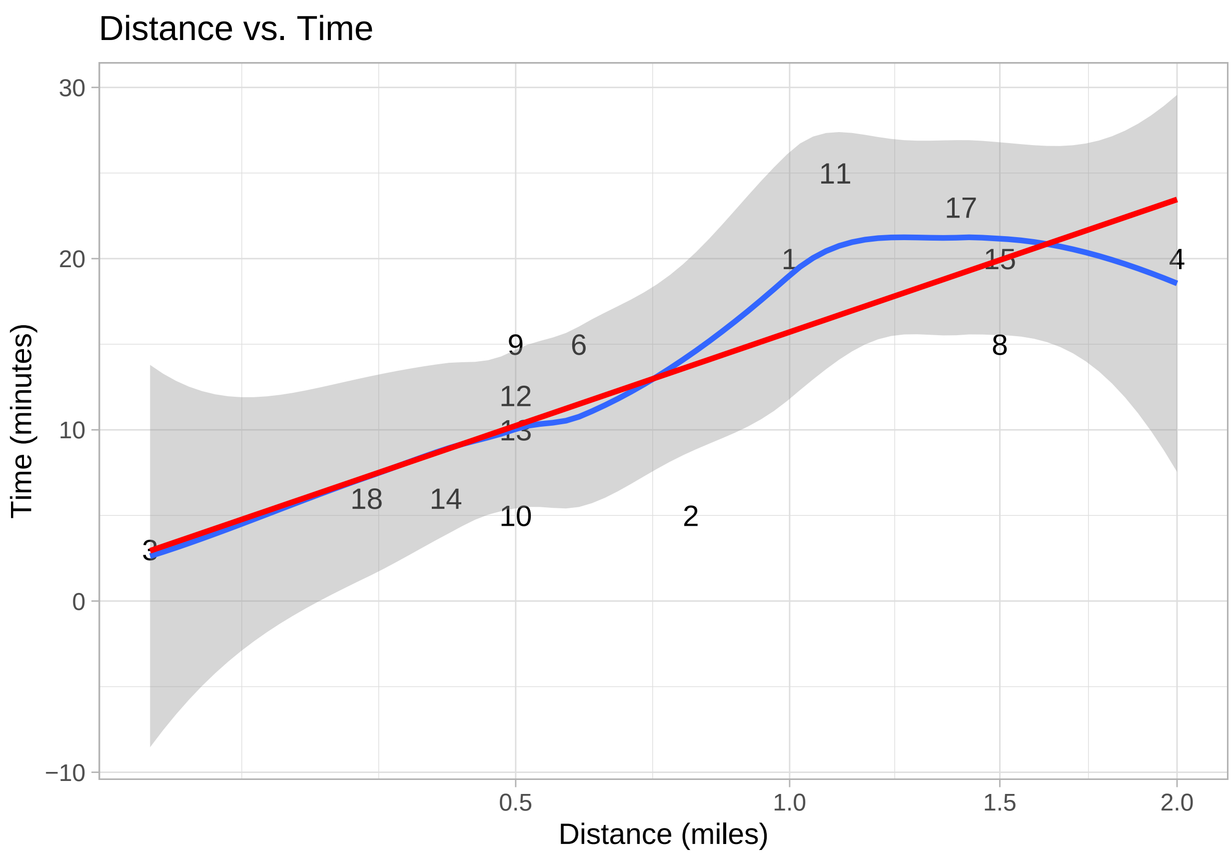


Comment on this plot:

All of the data points, excluding the `5th` and `7th`, are bunched together.

Produce the same plot, omitting the `5th` and `7th` data points:

```
ggplot(q_data[-c(5, 7, 16),], aes(x = Distance, y = Travel_time, label = data_point)) +
  theme_light() + geom_text() + geom_smooth() +
  geom_smooth(method = 'lm', se = F, col = 'red') +
  scale_x_sqrt() +
  labs(x = "Distance (miles)", y = "Time (minutes)",
       title = "Distance vs. Time")
```



Comment on this plot:

The travel time appears to depend on the square root of the distance.