Algoritma Academy: Regression Models

Samuel Chan

Updated: 15 October, 2024

Background

Algoritma

The following coursebook is produced by the team at Algoritma for its Data Science Academy workshops. The coursebook is intended for a restricted audience only, i.e. the individuals and organizations having received this coursebook directly from the training organization. It may not be reproduced, distributed, translated or adapted in any form outside these individuals and organizations without permission.

Algoritma is a data science education center with bootcamp programs offered in:

- Bahasa Indonesia (Jakarta campus)
- English (Singapore campus)

Lifelong Learning Benefits

If you're an active student or an alumni member, you also qualify for all our future workshops, 100% free of charge as part of your **lifelong learning benefits**. It is a new initiative to help you gain mastery and advance your knowledge in the field of data visualization, machine learning, computer vision, natural language processing (NLP) and other sub-fields of data science. All workshops conducted by us (from 1-day to 5-day series) are available to you free-of-charge, and the benefits **never expire**.

Second Edition

This coursebook is initially written in 2017.

This is the second edition, written in late August 2020. Some of the code has been refactored to work with the latest major version of R, version 4.0. I would like to thank the incredible instructor team at Algoritma for their thorough input and assistance in the authoring and reviewing process.

Libraries and Setup

We'll set-up caching for this notebook given how computationally expensive some of the code we will write can get.

```
knitr::opts_chunk$set(cache=TRUE,
                      tidy.opts=list(width.cutoff=60),
                      tidy=TRUE)
options(scipen = 9999)
rm(list=ls())
```

You will need to use `install.packages()` to install any packages that are not already downloaded onto your machine. You then load the package into your workspace using the `library()` function:

```
library(dplyr)
library(leaps)
library(GGally)
library(MASS)
library(car)
library(lmtest)
library(MLmetrics)
```

Training Objectives

This course strives for a fine balance between business applications and mathematical rigor in its treatment to regression models, one of the most essential statistical techniques in the field of machine learning. Its aim is to equip you with the knowledge to investigate relationships between variables of a data effectively and rigorously.

- **Linear Models**
- Least Squares Regression
- Simple Linear Regression
- Leverage vs Influence
- `lm` Function
- Linear Model Prediction
- **Interpretation & Application**
- Model Interpretation
- Arriving at Coefficients manually
- R-Squared
- Adjusted R-Squared
- Confidence Interval
- **Multivariate Regression**
- Multivariate Regression examples
- Model Selection

- Step-wise Regression
- Residual Plot
- Model Diagnostics
- Limitations of Regression Model

Regression Models I

Machine learning on a very basic level, refers to a sub-field of computer science that "gives computer the ability to learn without being explicitly programmed". Less-sensationally, it is concerned with the theory and application of statistical and mathematical methods to arrive at a particular objective without following a set of strictly defined and rigid pre-determined rules.

When the prediction value is numerical (think oil prices, rainfall, quarterly sales, blood pressure etc), it is generally referred to as a "regression" problem. This is in contrast with "classification" problems, a general term for when the value we're trying to predict is categorical (loan defaults, email spam collection, handwriting recognition etc).

It is important here to remind you that regression models are not just used in the machine learning context for numeric prediction. Regression, in fact, represent the "workhorse of data science"[1] and is among the most practical and theoretically understood models in statistics.

Data scientists well trained with this foundation will be able "to solve an incredible array of problems"[2]. Because regression models often lead to highly interpretable models, we can (and should) consider them as a handy statistical tool that has its place in some of the most common data science tasks:

- **Prediction**: Predict the profitability of a new product category given its pilot launch sales figure
- **Statistical Modeling**: Determining a quantitative relationship between price sensitivity and average sales unit
- **Covariation**: Determining the (residual) variation in average sales unit that appears unrelated to price levels; and to investigate the impact of other external factors beyond price points in explaining the fluctuation of average sales unit

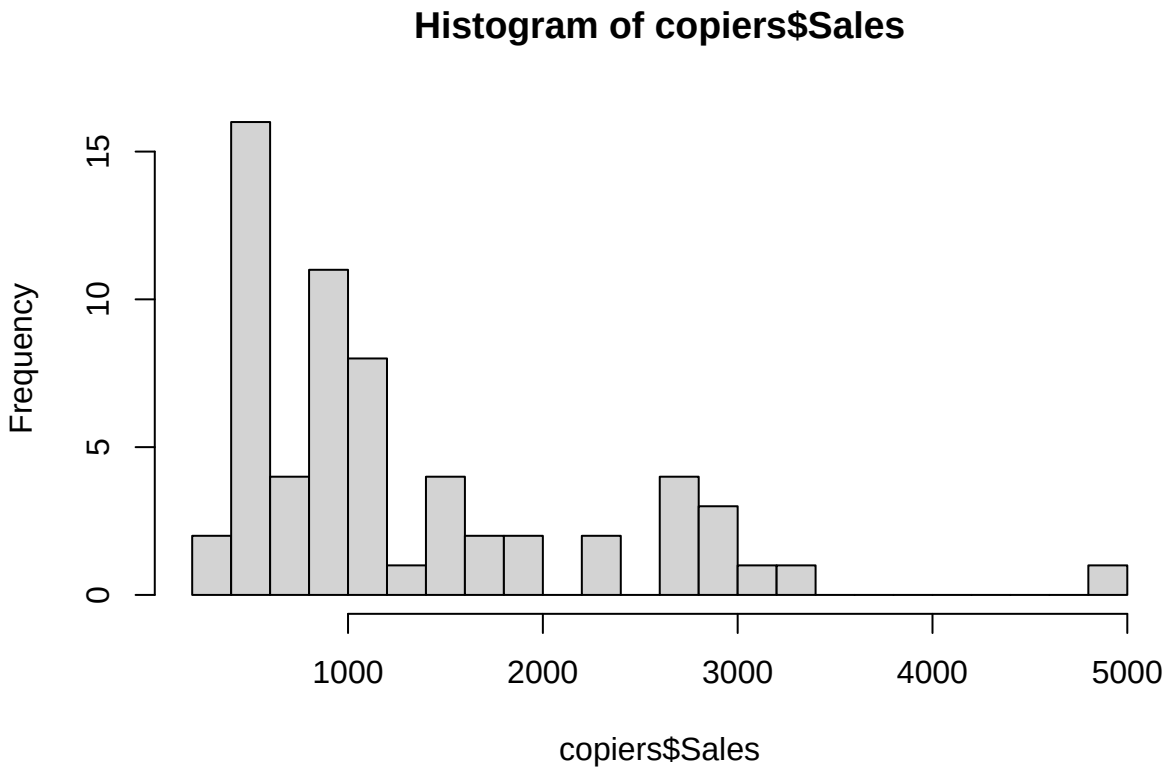
Least Squares Regression

One of the terms you'll hear a lot in the next 3 days is the **least squares**. So let's try and achieve some intuition about this important concept through the following illustration. I'm going to ahead and load some data and create a histogram from the resulting data.

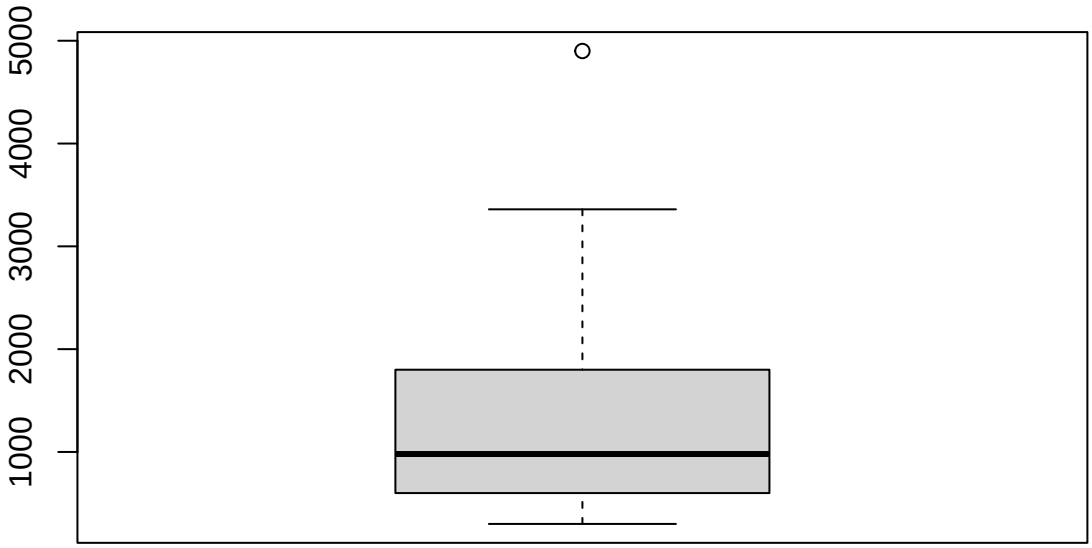
The data, as some among us may recall, is from an online retailer that specialize in the trading of office supplies and stationery:

```
copiers <- read.csv("data_input/copiers.csv")
hist(copiers$Sales, breaks=20)
```

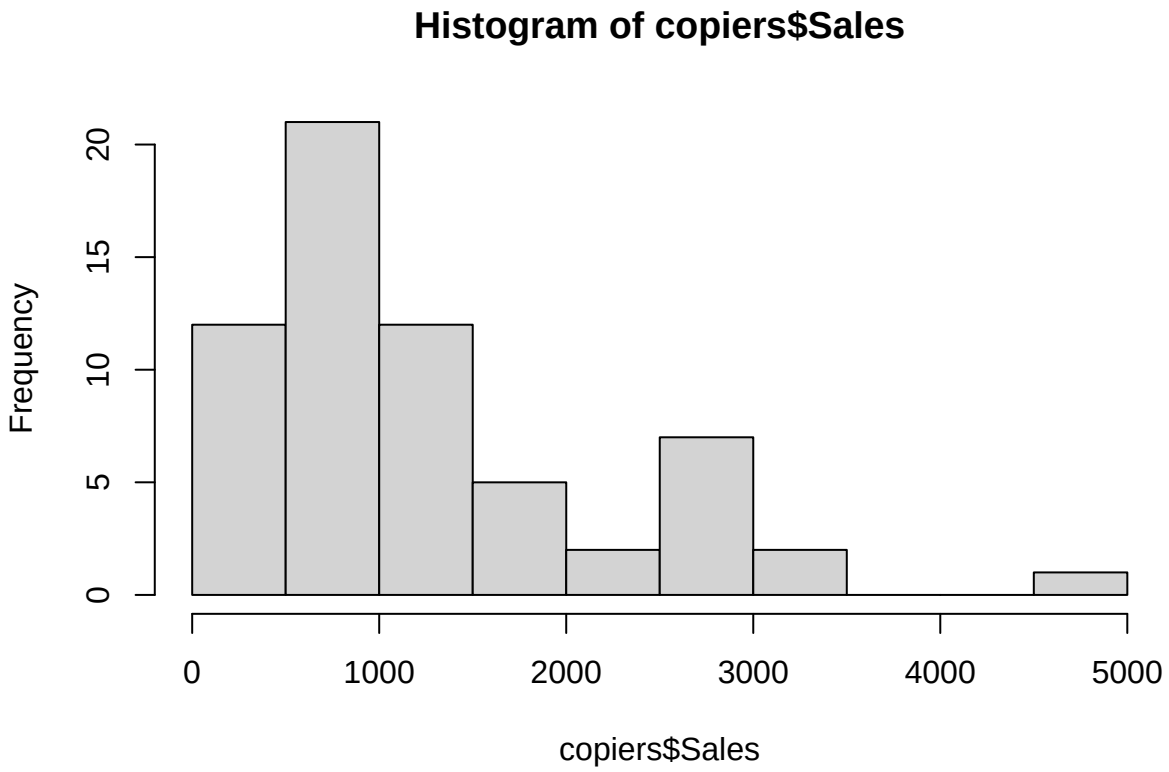
[1] L.Massaron, A.Boschetti, Regression Analysis with Python
[2] B.Caffo, Regression Models for Data Science in R




```
boxplot(copiers$Sales)
```




```
hist(copiers$Sales)
```



While the sales variable take on a rather large value (with an outlier at $5000), the idea of a least squares estimate is to identify a point in our data that minimizes the sum of the squared distances between the observed data and itself. We'll observe later that, with no predictor variables, this least squares estimate is the sample average.

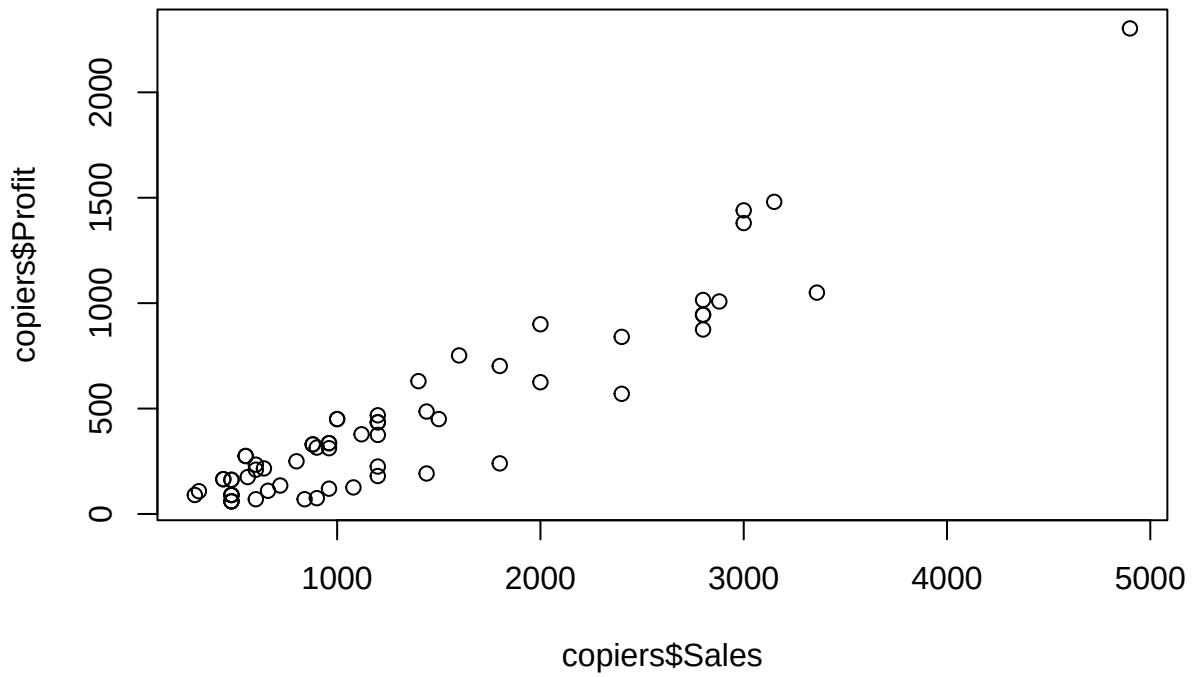
Because our estimation model isn't going to predict every observation perfectly, minimizing the average (which is equivalent to: the sum) of squared errors seem like a reasonable thing to do. If we had minimize the average absolute deviation between the data, it would lead us to the median as the least squares estimate instead of the mean. While this may seem intuitive to some, I am counting on some of you to be skeptical enough as to question me on whether the sample average would in fact lead us to the least squares estimate.

Let's do a simple exercise. I'll make a slider, and we'll go through some numbers and at each number we'll see the average least squares (`mean(sales - our_number)^2`) produced by that number. The least square estimate, as we will observe will turn out to be the sample average:

```
library(manipulate)
expr1 <- function(mn){
  mse <- mean((copiers$Sales - mn)^2)
  hist(copiers$Sales, breaks=20, main=paste("mean = ", mn, "MSE=", round(mse, 2), sep=""))
  abline(v=mn, lwd=2, col="darkred")
}
manipulate(expr1(mn), mn=slider(1260, 1360, step=10))
```

And now let's explain the importance of least squares in the context of regression models. Before I create a scatterplot of the sales data, I'd remove the far outlier (the one close to $5000) from our sample data and treat it as noise. Do note that removing outlying data (or in the general treatment of outliers) is not a decision to be taken lightly and generally involve a more methodical and lengthier treatment with respect to its implications. If done poorly, you may even be guilty of "doctoring" the data to fit your pre-determined narrative.

```
plot(copiers$Sales,copiers$Profit)
```

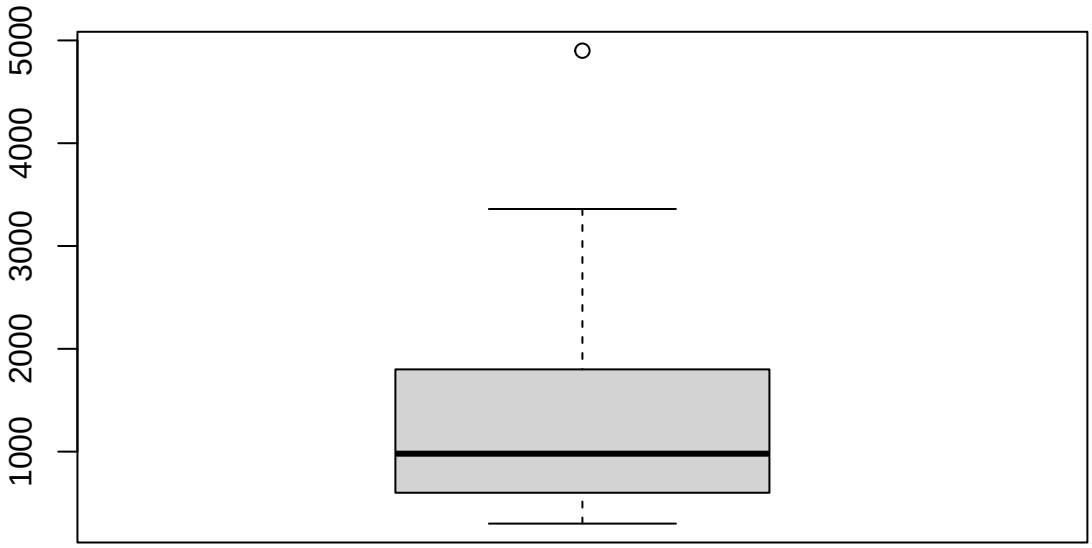


In this case, without diverting too far from the subject matter of this workshop, I think you can agree that the decision is well-justified. Another useful way to think about this particular decision in this case is to consider the following trade-off:

- Do we want the presence of a numerically distant observation (potentially $<$ ~1% of total observations) at the expense of "poorer model fit" for the rest (99%) of the observation?

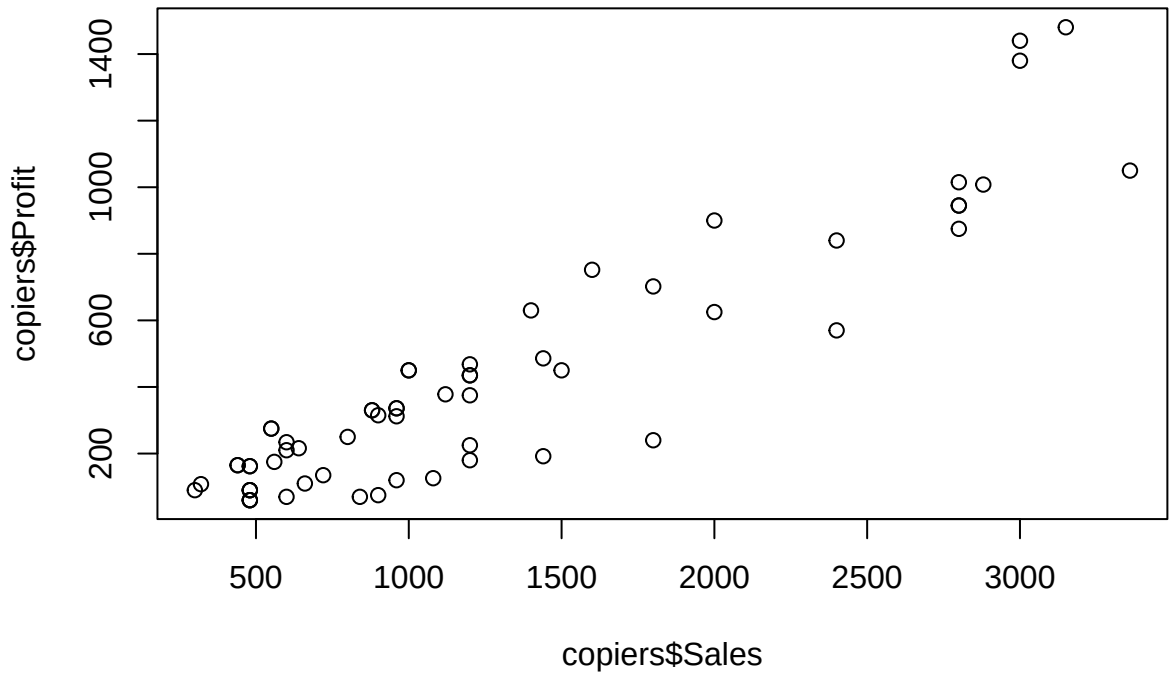
To refresh your memory, recall from your Practical Statistics class that the box plot identify an observation as an outlier if it's positioned above or below either of its "whiskers". That, we learned is 1.5 times the interquartile range above the upper quartile and bellow the lower quartile. Let's draw a box plot of our variable of interest:

```
boxplot(copiers$Sales)
```



Eliminating the outlier data from our original sample, we can now plot Sales against Profit and attain the following graph:

```
copiers <- copiers[copiers$Sales <= 3500, ]
plot(copiers$Sales, copiers$Profit)
```




```
cor(copiers$Sales,copiers$Profit)
```

```
## [1] 0.9227014
```

We can see a fairly linear relationship between the Sales and Profit variables of our `copiers` dataset, and the objective of a simple linear regression is concerned with modeling that relationship with a straight line.

Simple Linear Regression

Create a linear model in R is straightforward. We call the `lm()` function and specify two parameters: the `formula` for our linear model and the `data` from which our model is built from.

```
ols <- lm(formula = Profit ~ Sales,
          data = copiers)
```

Notice that we've saved `ols` as a linear model and we can now use the attributes of `ols`, such as its `$coefficients` to create our linear model:

```
class(ols)
```

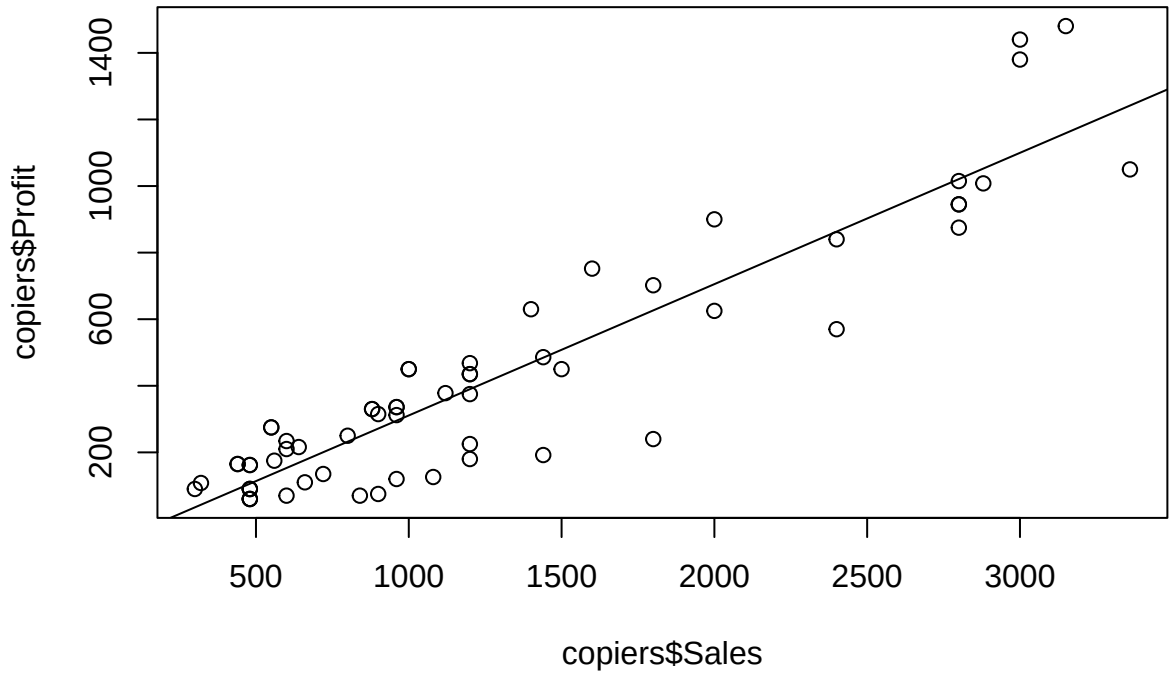
```
## [1] "lm"
```

```
ols$coefficients
```

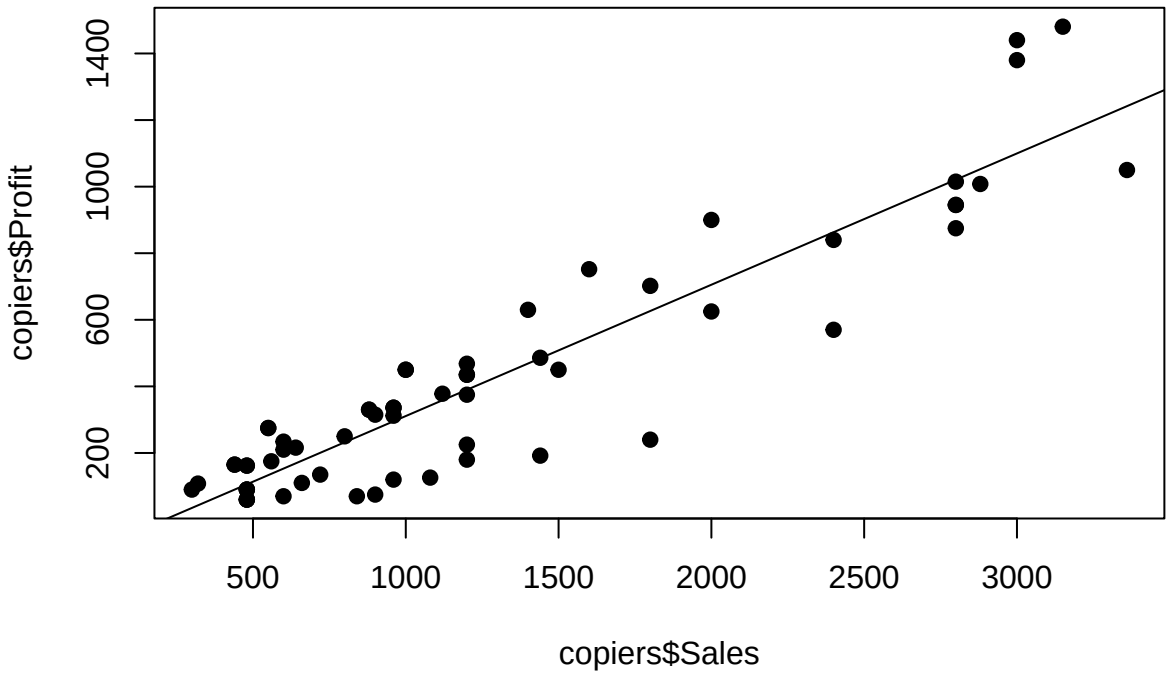
```
## (Intercept)       Sales
## -83.5481008   0.3944358
```

And with that let's create our plot again, but this time we'll also add a line that intercepts the y-axis at -83.548 and have a slope of 0.3944 degree, just as the coefficients of our model:

```
plot(copiers$Sales,copiers$Profit)
abline(ols$coefficients[1],ols$coefficients[2])
```




```
plot(copiers$Sales, copiers$Profit, pch=19)
abline(ols$coefficients[1], ols$coefficients[2])
```



That line, it turns out, is the line of best fit for our observations. This fitted line is estimated with error because the points are not perfectly predicted by the line (even though they hover around the line), and our linear model `ols` have estimated this line to minimize the least squares error.

Because of how common the `sum((actual-estimation)^2)` calculation is, we gave it a name: **sum of squared errors**, which is literal and memorable. R's `resid` function allows us to compute the difference between the predicted and actual values:

```
sum(resid(ols)^2)
```

```
## [1] 1195956
```

When I said "estimating a line", I hope it occurs to you that we're in fact estimating **two parameters**: the point at which our line cross the y-axis (known as the "intercept" term) and the slope degree (known as the "slope"). Collectively they're also referred to as **coefficients** (or "beta coefficients") of the linear model.

For most machine learning models we created in R, we can use generic functions such as `summary()` and `predict()` to obtain more information about the model, or on new values, respectively. Let's try and apply `summary()` on the model we just created:

```
summary(ols)
```

```
##
## Call:
## lm(formula = Profit ~ Sales, data = copiers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -386.43  -66.78   18.00   66.44  340.24
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
```

```
## (Intercept) -83.54810   32.82978  -2.545              0.0136 *
## Sales         0.39444    0.02146  18.384 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 142.4 on 59 degrees of freedom
## Multiple R-squared:  0.8514, Adjusted R-squared:  0.8489
## F-statistic:   338 on 1 and 59 DF,  p-value: < 0.00000000000000022
```

From the output of the `summary()` call, we see the two coefficients that represent our regression line. The point at which our line crosses the y-intercept is -83.548 and the slope is 0.3944. The coefficient of the slope is estimated so as to minimize the vertical distance between the observed points (truth) and the prediction. Because predictions are rarely perfect, the distance between each of these points and are captured in an attribute named `residuals`. Hence, to get the five number summary of the `residuals`, we would call `fivenum(ols$residuals)` or equivalently `summary(ols$residuals)`.

```
summary(ols$residuals)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -386.43  -66.78   18.00    0.00   66.44  340.24
```

The standard error that was printed from the `summary()` function is an estimate of the standard deviation of the coefficient, which, recall, is an indication of how much our estimate varies across samples (due to random sampling). The t-value we obtain from the summary is just the coefficient divided by the standard error:

```
summary(ols)$coefficients[,"Estimate"] / summary(ols)$coefficients[,"Std. Error"]
```

```
## (Intercept)       Sales
##   -2.544887   18.384226
```

Verifying that they're the same:

```
summary(ols)$coefficients[,"t value"]
```

```
## (Intercept)       Sales
##   -2.544887   18.384226
```

The last column of the summary section give us a probability of obtaining a t-value as extreme as what we just observed, a concept that is familiar to what we've learned in Practical Statistics. It could have been obtained from `2*pt(-2.544887, n)` where `n` is the number of samples. From the earlier `summary()` call we also see significance codes denoted with `***`, `**`, `.` etc. That is just a visual representation of the p-value (`Pr(>|t|)`), so a p-value that is lesser than 0.001 is denoted with `***`, and a p-value between 0.01 and 0.05 is denoted with `*`, and so on and so forth.

The reason why we're concerned with these statistics is that in regression modeling, we're trying to investigate if there's sufficient evidence that our independent variables (or "predictor variables") are different from 0.

That is to say, we like to know if variables like Sales, Branch ID, Time of Day have any real effect on the profitability of our Profit, or if variables such as humidity level, temperature, or duration of outdoor exposure has any real effect on machinery corrosion level. Through a regression model, we want to see if these variables are genuinely affecting the target variable ("profit") or that any apparent differences in profit are just due to random chance and sampling; The null hypothesis, recall, is that the reason we observe a

difference in profitability across stores for example, is ultimately due to random chance from sampling - if we have sampled a different number of months, we would have observed a completely or even contradictory scenario!

Also recall from our practical statistics class that the rule of thumb to reject the null hypothesis (no effect) and favor the alternative hypothesis is 0.05. That is, if the chance of us observing a profit distribution such as these provided that they are in fact independent from store branches is only 5%, then we'd reject the null hypothesis.

Note: If you need a refresher on the statistical concepts I've mentioned above, I encourage you to quickly glance over the Practical Statistics coursebook again and do some revision with your assigned academic mentor. If you'd like an in-depth treatment of the subject, I can recommend the free PDF copy of The Elements of Statistical Learning: Data Mining, Inference, and Prediction - download link in annotation[3].

```
summary(ols)[[4]]
```

```
##                Estimate  Std. Error   t value
## (Intercept) -83.5481008 32.82978329 -2.544887
## Sales         0.3944358  0.02145512 18.384226
##                                     Pr(>|t|)
## (Intercept) 0.013566993750767503135179481432 71
## Sales       0.000000000000000000000000004215473
```

Now that we have the coefficients, what does that tell us? Well the size of the coefficients tell us the effect that variable has on our target variable. We observed here that Sales have a coefficient of 0.39 on Profit, and because 0.39 is a positive number we know that the effect is positive: the higher the Sales, the higher the Profit. A negative coefficient will indicate the opposite, and an example of that would be Profit vs Market Saturation: the increasingly saturated market leads to decreasing profit.

As a quick knowledge check – can you think of another example where we might observe a negative coefficient in a regression model?

In addition to the above information, we've also derived the profit equation from our linear model directly. It takes the form:
$\hat{Y} = \beta_0 + \beta_1 X_1$

Which in plain English means: Estimated Profit = Intercept + Slope * Sales

Substituting the beta coefficients into the formula hence yield: Estimated Profit = -83.5481008 + 0.3944358 * Sales

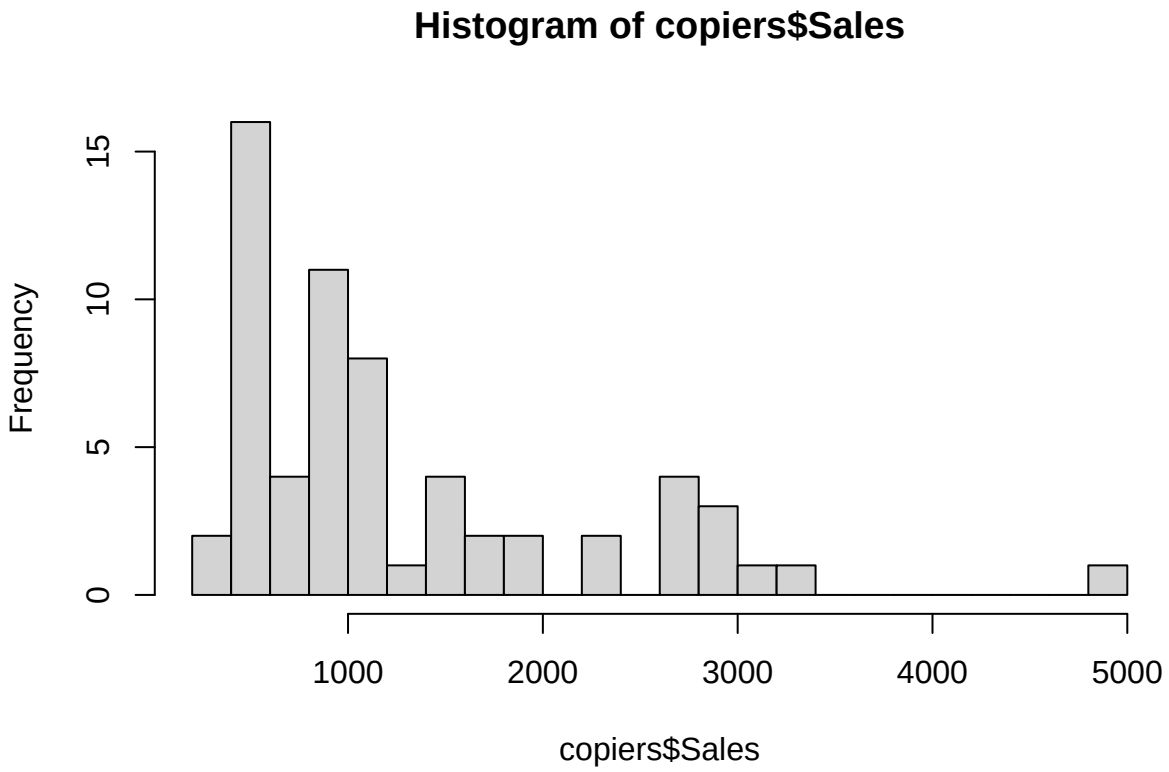
That tells us that the profit is expected to increase by $0.39444 when the sales price of our Copiers machine increase by $1, and decrease by $0.39444 as the sales price of our Copiers machine decrease by $1. For a Copiers machine with a listed price of $0, the profit is predicted to be negative (incurring a loss of approximately $-83.55).

Leverage vs Influence

Recall earlier that we left out the outlying data point; To illustrate the idea of leverage and power we'll override the `copiers` dataset with the original data read from our csv again.

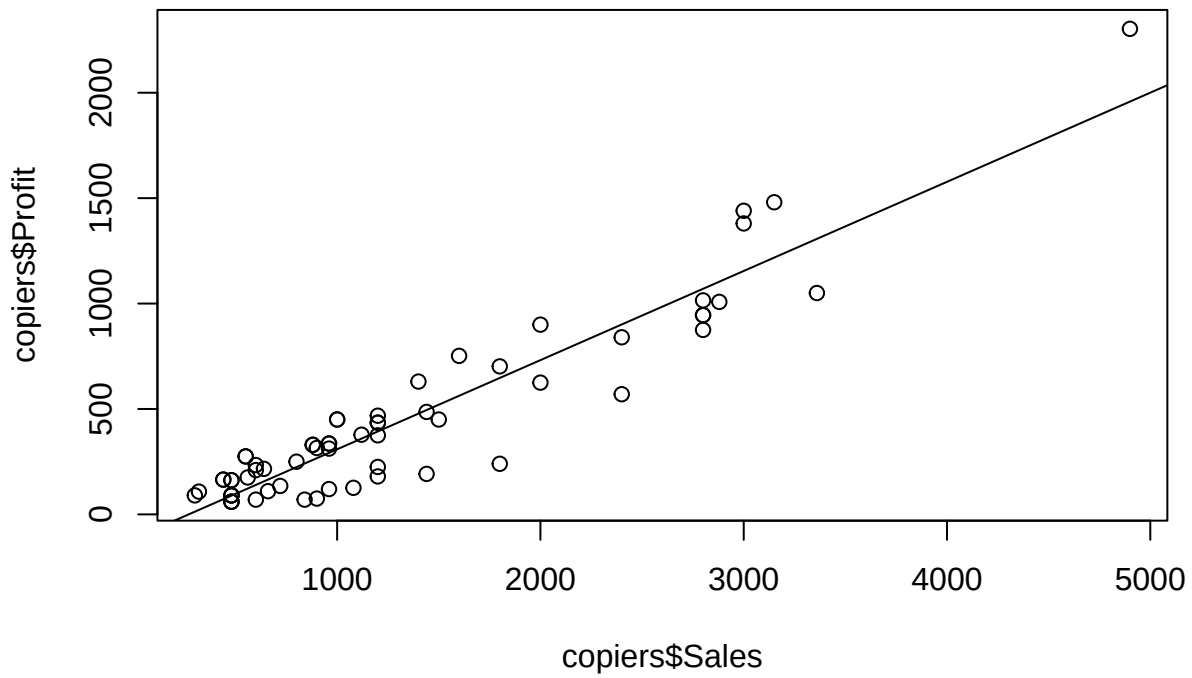
```
copiers <- read.csv("data_input/copiers.csv")
hist(copiers$Sales, breaks=20)
```

[3] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction



Now let's create a linear model named `ols_outlier` and plot the regression line:

```
ols_outlier <- lm(Profit ~ Sales, copiers)
plot(copiers$Sales, copiers$Profit)
abline(ols_outlier$coefficients[1], ols_outlier$coefficients[2])
```



Notice the presence of that outlier have influenced our regression line but not by a degree that makes it immediately apparent (at least visually). Let's inspect the impact our outlier data point has had on our

model by printing the coefficients of this model with our outlier-exempted model:

```
ols_outlier$coefficients
```

```
##  (Intercept)        Sales
## -114.0625136    0.4228588
```

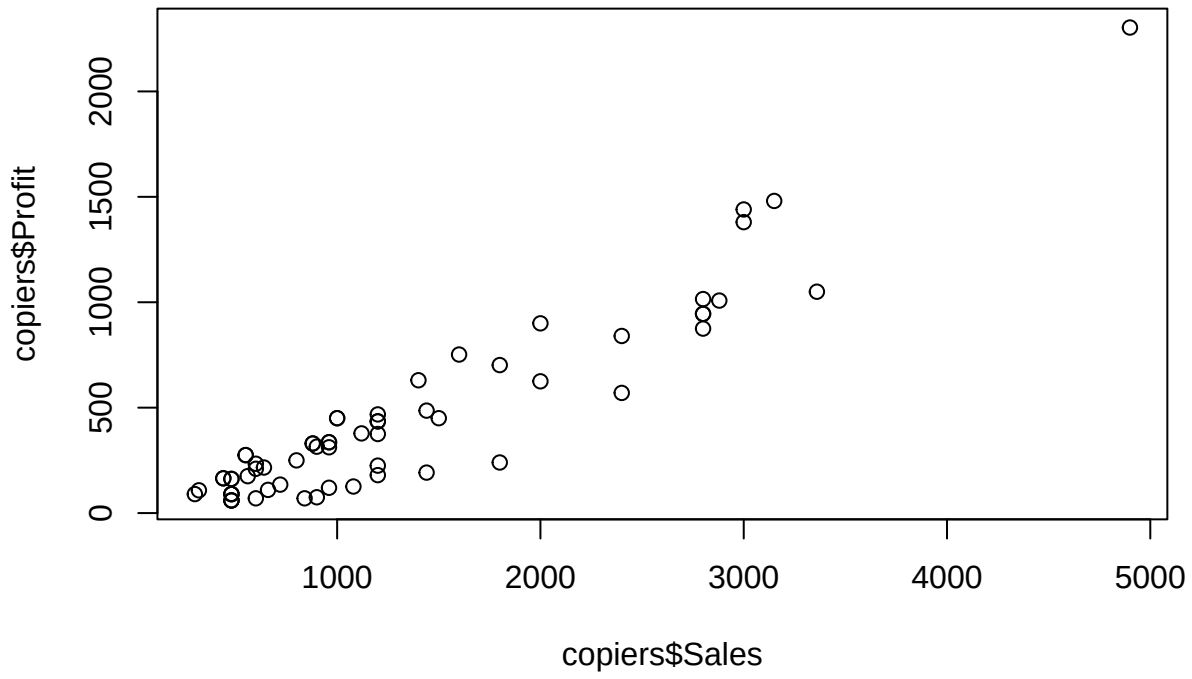
```
ols$coefficients
```

```
## (Intercept)       Sales
## -83.5481008   0.3944358
```

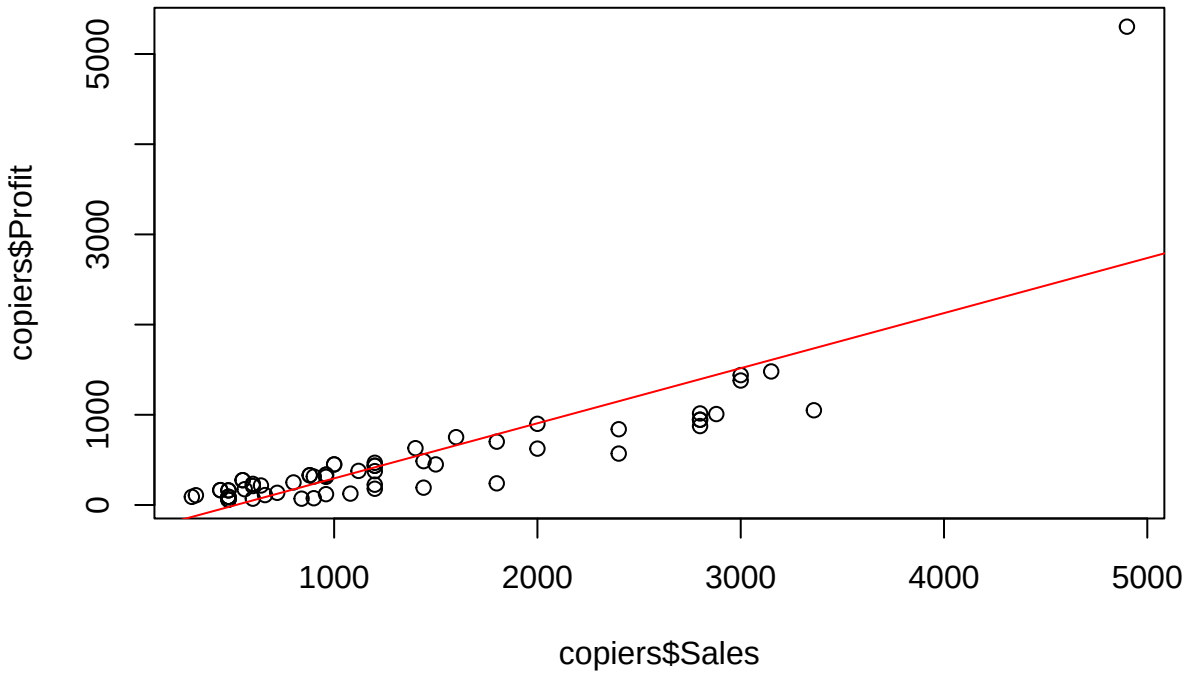
Our outlier there is said to have **high leverage** if its predictor value (x-axis) is **far away from the mean of the predictor values for all observations**. In other words, if a point is far away from where the rest of the data, we say that the point has high leverage. This point has the potential to greatly influence the coefficients of our regression model, but we have yet to establish whether it did.

Supposed that the point with high leverage is positioned in a way that largely fits the pattern observed in the rest of the data, then it is not particularly influential. To see this in action, I've arbitrarily increase the y-value of our outlier point by a certain margin, and plot the regression line to see how the outlier has in fact "exercised" its influence:

```
plot(copiers$Sales,copiers$Profit)
```

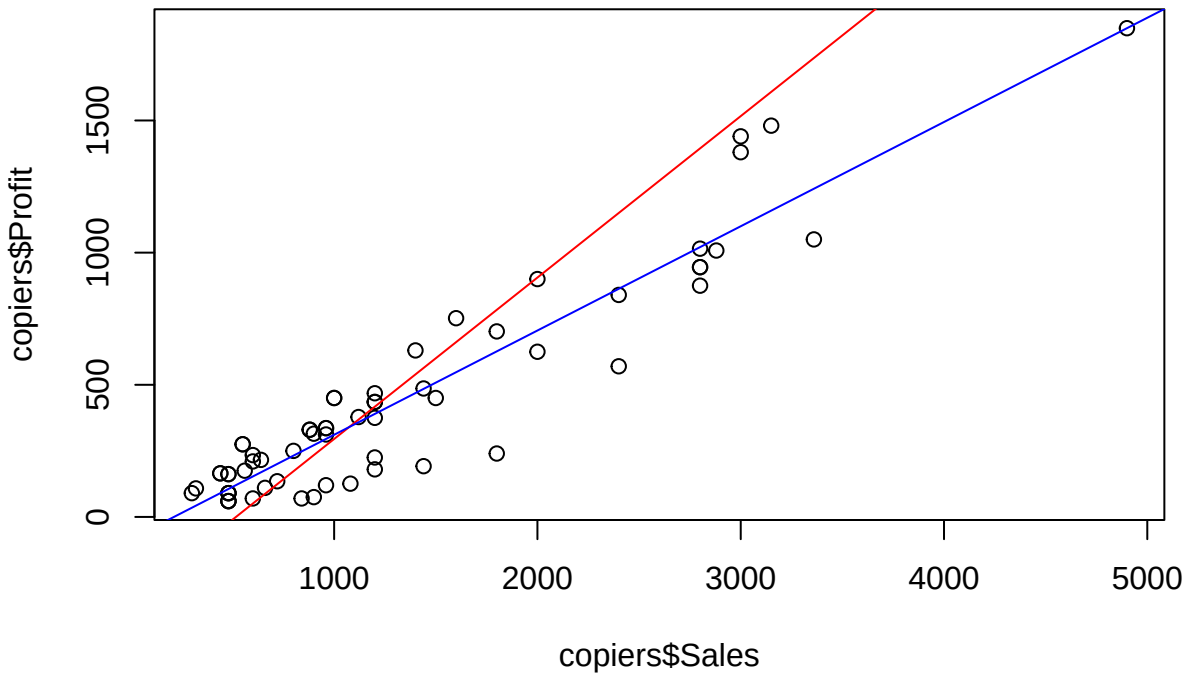



```
copiers[59,"Profit"] <- copiers[59,"Profit"] + 3000
ols_outlier2 <- lm(Profit ~ Sales, copiers)
plot(copiers$Sales, copiers$Profit)
abline(ols_outlier2$coefficients[1], ols_outlier2$coefficients[2], col="red")
```



Have the y-value (`Profit`) of that outlier point been largely consistent with the pattern we've observed in the rest of the data (blue line), we would have said that the outlier point has large leverage but little influence - in other words, the inclusion or exclusion of this outlier does not seem to change the coefficient estimates by much.

```
copiers[59,"Profit"] <- 1849.16
ols_outlier3 <- lm(Profit ~ Sales, copiers)
plot(copiers$Sales, copiers$Profit)
abline(ols_outlier2$coefficients[1], ols_outlier2$coefficients[2], col="red")
abline(ols_outlier3$coefficients[1], ols_outlier3$coefficients[2], col="blue")
```



```
c("Model"="Without Outliers", round(ols$coefficients,2), "R-Squared"=round(summary(ols)$r.squared,2))
```

```
##              Model        (Intercept)              Sales          R-Squared
## "Without Outliers"           "-83.55"             "0.39"             "0.85"
```

```
c("Model"="High Influence (Red)", round(ols_outlier2$coefficients,2), "R-Squared"= round(summary(ols_ou
```

```
##                  Model            (Intercept)                  Sales
## "High Influence (Red)"              "-315.79"                "0.61"
##              R-Squared
##                 "0.67"
```

```
c("Model"="Low Influence (Blue)", round(ols_outlier3$coefficients,2), "R-Squared"= round(summary(ols_ou
```

```
##                  Model            (Intercept)                  Sales
## "Low Influence (Blue)"              "-83.55"                "0.39"
##              R-Squared
##                 "0.88"
```

Interestingly, we hardly observe any changes in the coefficients as well as the R-squared between the outlier-exempted model and the third model – in fact the inclusion of that outlier point even fractionally increase our R-squared value, from 0.85 to 0.87.

However, another outlier point that share the same leverage (its x-value is equally distant from the mean of the other observations) have substantially decreased our R-squared, and we can observe from the coefficients and from the visual plot that the inclusion of this outlier has strong influence on our coefficients.

As a data scientist, it is important we learn to recognize when our regression models are unduly influenced by rare but influential data points - the inclusion of one or more influential outliers could strongly bias our regression model (intercept changes from -83.55 to -315.79; slope changes 0.39 to 0.61 in the earlier exercise) and render our analysis null.

Linear Model Prediction

Earlier we learned to estimate our profit from sales amount using the coefficients we obtained in the outlier-excluded model. As a reminder, let's inspect the coefficients of our first linear model again:

```
ols$coefficients
```

```
## (Intercept)       Sales
## -83.5481008   0.3944358
```

Substituting the beta coefficients into the formula hence yield: Estimated Profit = -83.5481008 + 0.3944358 * Sales

Supposed we're expecting a sales transaction by the end of day amount $1,000. What would our linear model predict its profit to be?

```
-83.5481008 + 0.3944358 * 1000
```

```
## [1] 310.8877
```

Estimated Profit = -83.5481008 + 0.3944358 * 1000 = 310.8877

It turns out that our linear model, `ols` would predict a profit of $310.8877. Let's compare this result to using R's built-in `predict()` function. This function allow us to obtain predictions given some input data. `predict` expects a machine learning model as its first parameter, and in this case a data frame to predict on. We already have the model object (`ols`) so let's create the new data for us to work with.

```
transaction_today <- data.frame(Sales=c(1000,500,700,5000))
```

And with that we can apply the `predict()` function rather easily:

```
predict(ols,transaction_today)
```

```
##         1         2         3         4
##  310.8877  113.6698  192.5570 1888.6309
```

The result is exactly the same as what we obtained manually (by hand).

[Optional] Estimating the beta coefficients manually

This section is optional - you can *skip* this section entirely and the concepts presented here will not be graded.

I want to convince you that the coefficients we obtained using `lm()` is really nothing "magical". In fact, recall from the beginning of the workshop where I've mentioned that regressions are among the most theoretically understood models in all of machine learning!

Let's start by revisiting some of the concepts in our Practical Statistics course. First, the variance of things - that's how *varied* observations are on the average (hence the name "variance"). Variance can be obtained by taking the squares of differences between each observation and the mean and then taking the average of that statistic:

```
var(copiers$Sales)
```

```
## [1] 934109.1
```

```
sum((copiers$Sales - mean(copiers$Sales))^2)/(length(copiers$Sales)-1)
```

```
## [1] 934109.1
```

So variance measures how much a variable *varies* around its mean. Now let's take a look at covariance. Covariance measures how much two variables *vary* together. In mathematical notation, consider we have a dataset with pairs of data, (X_i, Y_i), their empirical covariance would then be

$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$, also:

$Cov(X,Y) = \frac{1}{n-1}(\sum_{i=1}^{n} X_i Y_i - n\bar{X}\bar{Y})$

If the formula or covariance looks confusing, it's helpful to again recall the formula of **variance**, and understand that covariance is just a measure of how our variables **co-vary**. Hence, our variance:

- $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$

And our covariance formula:
- $Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$

The covariation is a measure of limited utility since its units are the product of the units of the two variables; A more useful definition normalizes the two variables first, which leads us to the correlation definition:
$Cor(X,Y) = \frac{Cov(X,Y)}{S_x S_y}$

Some facts about correlation:

- Cor(X,Y) = Cor(Y,X)
- -1 <= Cor(X,Y) <= 1
- Cor(X,Y) = 1 and Cor(X,Y) = -1 only when the X or Y observations fall perfectly on a positive or negative sloped line, respectively.
- Cor(X,Y) measures the strength of the linear relationship between the X and Y data, with stronger relationships as Cor(X,Y) heads towards -1 or 1
- Cor(X,Y) = 0 implies no linear relationship

As an exercise, call `cov()` and `cor()` on the `mtcars` dataset below and observe the range of values that are returned to you:

[Optional] Least Squares Estimations

With the above new knowledge, consider our profit estimator regression again. If we were to find the best line to describe the profit using sales as a prediction, one equation we can consider is to find a "baseline" (we can call it "beta 0") that represent profit at $0 sales, and then add beta 0 to a second parameter named beta 1 where it represent the increment of profit at each additional $1 of sales.

How do we know if the beta 0 and beta 1 we have chosen has indeed result in the most optimal line? Well that we've already learned earlier: we can use the least squares criteria! Specifically, we can find the combination of beta 0 and beta 1 that minimizes the squared vertical distances between the data points for actual profit and the points on the fitted line for each predicted profit:

$$\sum_{i=1}^{n}\{Y_i - (\beta_0 + \beta_1 X_i)\}^2$$

Additionally, the way we would apply linear regression using our least squares model above, is where we're trying to fit the line $Y = \hat{\beta}_0 + \hat{\beta}_1 X$ where:

- The est. slope: $\hat{\beta}_1 = Cor(Y,X)\frac{Sd(Y)}{Sd(X)}$
- Est. intercept: $\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$

So it is shown that what `lm()` is doing is in fact ordinary least squares regression and the coefficients it obtains could have been estimated from traditional statistical methods as well. Using the covariance or correlation of `they` (our target / dependent variable) and `x` (our predictor / independent variable), we could have obtained the estimate of the slope:

```
ols
```

```
##
## Call:
## lm(formula = Profit ~ Sales, data = copiers)
##
## Coefficients:
## (Intercept)        Sales
##    -83.5481       0.3944
```

```
slope <- cor(copiers$Sales, copiers$Profit) * (sd(copiers$Profit)/sd(copiers$Sales))
slope
```

```
## [1] 0.3944358
```

And using that slope estimate, we could then obtain an estimate of our intercept:

```
#y = mx + c
c = mean(copiers$Profit) - mean(copiers$Sales) * slope
c
```

```
## [1] -83.54812
```

And verify that the slope and intercept we obtain is in fact the same as what `lm()` returns:

```
summary(lm(Profit ~ Sales, copiers))$coefficients
```

```
##                Estimate Std. Error   t value
## (Intercept) -83.5481183 30.6806773 -2.723151
## Sales         0.3944358  0.0187033 21.089103
##                                          Pr(>|t|)
## (Intercept) 0.008455272733379431623901290038247 68
## Sales       0.000000000000000000000000000001949109
```

Let's talk about another important concept of a regression model: the R-squared.

R-squared

R squared by definition is the percentage of the total variability that is explained by the linear relationship with the predictor (Regression Variation / Total Variation):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

In other words, R squared can be thought of as a quantity that represents the % of the total variation that's represented by the model. We simply take the regression variation and divide it by the total variation. In our case, it is the % of the variation in profit *that is explained by the regression relationship with sales*. Some facts about R^2:

- R^2 is the percentage of variation explained by the regression model
- $0 \leq R^2 \leq 1$
- If we define R as the sample correlation between the predictor and the outcome, R^2 is simply the sample correlation squared

Because R-squared is a statistical measure of how close the data are to the fitted line, we want our model to achieve a high R-squared as it means our model has fit the data well (not always the case, but we'll get to that later).

When we use `summary()` on our least squares regression model, its output include the r-squared (in some statistical software / machine learning domains this is also called the coefficient of determination). We can find the R-squared directly from the summary:

```
ols <- lm(Profit ~ Sales, copiers)
summary(ols)$r.squared
```

```
## [1] 0.8811293
```

Now let's try and manually arrive at the R-squared value, just as we did with the coefficients of the slope and intercept. Plucking our example into the formula specified above:

```
predict <- predict(ols)
actual <- copiers$Profit
r2 <- (sum((predict - mean(copiers$Profit))^2))/(sum((actual - mean(copiers$Profit))^2))
r2
```

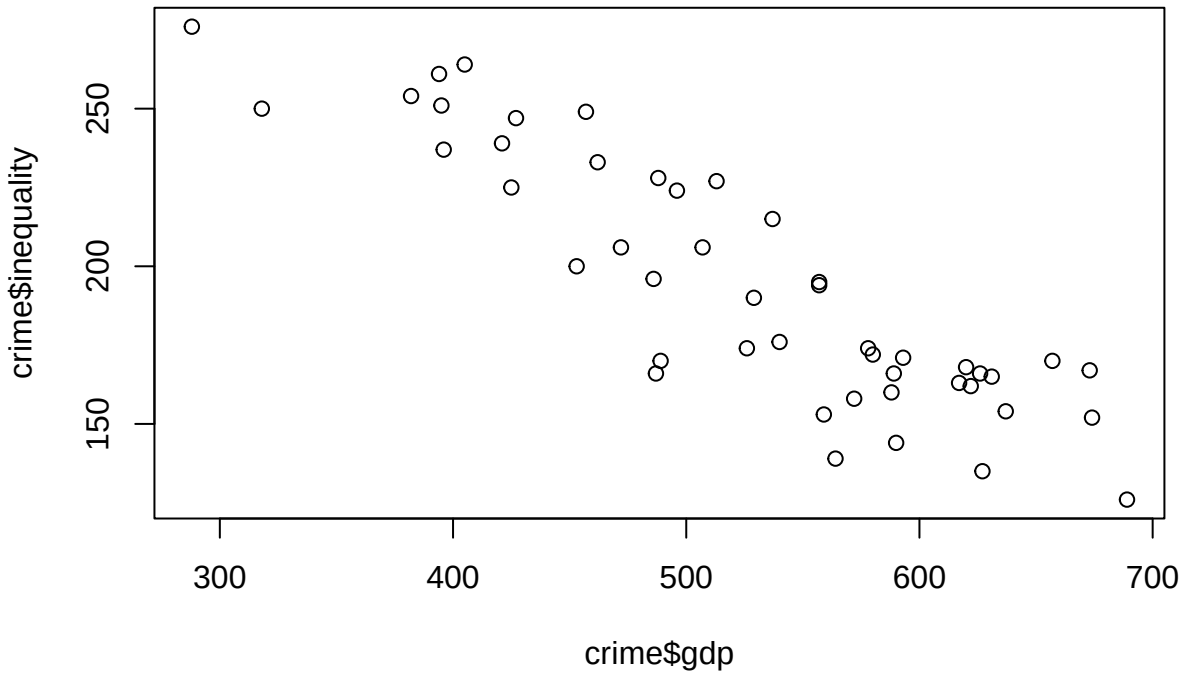
```
## [1] 0.8811293
```

You can see from our manual calculations above that R squared really is a ratio and ranges from 0 to 1. This is observed from the third line of the code. Specifically, it is a ratio that describes the amount of error explained by two different models.

Graded Assignment: Investigating GDP, Income Inequality and Crime
This part of the assignment is graded for Academy students. Please fill up your answers in the provided answer sheet. Every correct answer is worth **(1) Point**.

In the following code we take a peek at a dataset used by criminologists to study the effect of punishment regimes on crime rates. We'll read the dataset and rename the columns. We're particularly interested in the effect of the `gdp` variable (gross domestic product per head) on income inequality so we went ahead and plot that.

```
crime <- read.csv("data_input/crime.csv")
crime <- crime[,-1]
names(crime) <- c("percent_m", "is_south", "mean_education", "police_exp60", "police_exp59", "labour_pa
plot(crime$gdp, crime$inequality)
```



Extra explanations for your `crime` dataset

The dataset was collected in 1960 and a full description of the dataset was available in here. I use the description I gathered from the authors of the MASS package. After you rename the dataset (in your coursebook, around line 410 to line 420), the variables are:
- `percent_m`: the number of males aged 14 to 24 years per 1000 of total state population - `is_south`: whether it is in a Southern state. 1 for Yes, 0 for No.
- `mean_education`: mean number of years of schooling times 10 of the population 25 years old and over
- `police_exp60`: police expenditure in 1960
- `police_exp59`: police expenditure in 1959 - `labour_participation`: labour force participation rate
- `m_per1000f`: number of males per 1000 females
- `state_pop`: state population
- `nonwhites_per1000`: number of non-whites resident per 1000 people
- `unemploy_m24`: Unemployment rate of urban males per 1000 aged 14-24
- `unemploy_m39`: unemployment rate of urban males per 1000 aged 35-39
- `gdp`: gross domestic product per head
- `inequality`: income inequality (the number of families per 1000 earning below one half of the median income) - `prob_prison`: probability of imprisonment
- `time_prison`: avg time served in prisons
- `crime_rate`: crime rate in an unspecified category Produce a linear model (`inequality ~ gdp`) and inspect the model using `summary()`, then proceed to answer the graded assignments.

Graded Quizzes

Question 1: Which of the following best described the slope?

(A) • It's a **negative** slope, and is statistically **significant** (P-value lower than 0.05)
(B) • It's a **positive** slope, and is statistically **significant** (P-value lower than 0.05)
(C) • It's a **negative** slope, and is statistically **insignificant** (P-value higher than 0.05)
(D) • It's a **positive** slope, and is statistically **insignificant** (P-value higher than 0.05)

Question 2: Given a GDP of 600, what would the inequality measure be? Use `predict()` on the new value

Enter the predicted value (up to 2 decimal points, if any) into the input field.

Question 3: What is the most fitting conclusion from the regression model above?
(A) - The R-squared approximates 0.78, indicating a reasonable fit (the closer to 1 the better) (B) - The R-squared approximates 0.78, indicating a poor fit (the closer to 0 the better)
(C) - The R-squared does not tell us about the quality of our model fit. That is the job of the p-value.

```
library(GGally)
ggcorr(crime, label = T,label_size = 2.9)
```


```
cor(crime)
```

```
##                       percent_m    is_south mean_education police_exp60
## percent_m             1.00000000  0.58435534    -0.53023964  -0.50573690
## is_south              0.58435534  1.00000000    -0.70274132  -0.37263633
## mean_education       -0.53023964 -0.70274132     1.00000000   0.48295213
## police_exp60         -0.50573690 -0.37263633     0.48295213   1.00000000
## police_exp59         -0.51317336 -0.37616753     0.49940958   0.99358648
## labour_participation -0.16094882 -0.50546948     0.56117795   0.12149320
## m_per1000f           -0.02867993 -0.31473291     0.43691492   0.03376027
## state_pop            -0.28063762 -0.04991832    -0.01722740   0.52628358
## nonwhites_per1000     0.59319826  0.76710262    -0.66488190  -0.21370878
## unemploy_m24         -0.22438060 -0.17241931     0.01810345  -0.04369761
## unemploy_m39         -0.24484339  0.07169289    -0.21568155   0.18509304
## gdp                  -0.67005506 -0.63694543     0.73599704   0.78722528
## inequality            0.63921138  0.73718106    -0.76865789  -0.63050025
```

```
## prob_prison           0.36111641  0.53086199    -0.38992286  -0.47324704
## time_prison           0.11451072  0.06681283    -0.25397355   0.10335774
## crime_rate           -0.08947240 -0.09063696     0.32283487   0.68760446
##                     police_exp59 labour_participation  m_per1000f   state_pop
## percent_m            -0.51317336          -0.1609488 -0.02867993 -0.28063762
## is_south             -0.37616753          -0.5054695 -0.31473291 -0.04991832
## mean_education        0.49940958           0.5611780  0.43691492 -0.01722740
## police_exp60          0.99358648           0.1214932  0.03376027  0.52628358
## police_exp59          1.00000000           0.1063496  0.02284250  0.51378940
## labour_participation  0.10634960           1.0000000  0.51355879 -0.12367222
## m_per1000f            0.02284250           0.5135588  1.00000000 -0.41062750
## state_pop             0.51378940          -0.1236722 -0.41062750  1.00000000
## nonwhites_per1000    -0.21876821          -0.3412144 -0.32730454  0.09515301
## unemploy_m24         -0.05171199          -0.2293997  0.35189190 -0.03811995
## unemploy_m39          0.16922422          -0.4207625 -0.01869169  0.27042159
## gdp                   0.79426205           0.2946323  0.17960864  0.30826271
## inequality           -0.64815183          -0.2698865 -0.16708869 -0.12629357
## prob_prison          -0.47302729          -0.2500861 -0.05085826 -0.34728906
## time_prison           0.07562665          -0.1236404 -0.42769738  0.46421046
## crime_rate            0.66671414           0.1888663  0.21391426  0.33747406
##                     nonwhites_per1000 unemploy_m24 unemploy_m39          gdp
## percent_m                  0.59319826 -0.224380599  -0.24484339 -0.6700550558
## is_south                   0.76710262 -0.172419305   0.07169289 -0.6369454328
## mean_education            -0.66488190  0.018103454  -0.21568155  0.7359970363
## police_exp60              -0.21370878 -0.043697608   0.18509304  0.7872252807
## police_exp59              -0.21876821 -0.051711989   0.16922422  0.7942620503
## labour_participation      -0.34121444 -0.229399684  -0.42076249  0.2946323090
## m_per1000f                -0.32730454  0.351891900  -0.01869169  0.1796086363
## state_pop                  0.09515301 -0.038119948   0.27042159  0.3082627091
## nonwhites_per1000          1.00000000 -0.156450020   0.08090829 -0.5901070652
## unemploy_m24              -0.15645002  1.000000000   0.74592482  0.0448572017
## unemploy_m39               0.08090829  0.745924815   1.00000000  0.0920716601
## gdp                       -0.59010707  0.044857202   0.09207166  1.0000000000
## inequality                 0.67731286 -0.063832178   0.01567818 -0.8839972758
## prob_prison                0.42805915 -0.007469032  -0.06159247 -0.5553347075
## time_prison                0.23039841 -0.169852838   0.10135833  0.0006485587
## crime_rate                 0.03259884 -0.050477918   0.17732065  0.4413199490
##                      inequality  prob_prison   time_prison  crime_rate
## percent_m            0.63921138  0.361116408  0.1145107190 -0.08947240
## is_south             0.73718106  0.530861993  0.0668128312 -0.09063696
## mean_education      -0.76865789 -0.389922862 -0.2539735471  0.32283487
## police_exp60        -0.63050025 -0.473247036  0.1033577449  0.68760446
## police_exp59        -0.64815183 -0.473027293  0.0756266536  0.66671414
## labour_participation -0.26988646 -0.250086098 -0.1236404364  0.18886635
## m_per1000f          -0.16708869 -0.050858258 -0.4276973791  0.21391426
## state_pop           -0.12629357 -0.347289063  0.4642104596  0.33747406
## nonwhites_per1000    0.67731286  0.428059153  0.2303984071  0.03259884
## unemploy_m24        -0.06383218 -0.007469032 -0.1698528383 -0.05047792
## unemploy_m39         0.01567818 -0.061592474  0.1013583270  0.17732065
## gdp                 -0.88399728 -0.555334708  0.0006485587  0.44131995
## inequality           1.00000000  0.465321920  0.1018228182 -0.17902373
## prob_prison          0.46532192  1.000000000 -0.4362462614 -0.42742219
## time_prison          0.10182282 -0.436246261  1.0000000000  0.14986606
## crime_rate          -0.17902373 -0.427422188  0.1498660617  1.00000000
```

Regression Models II

Multiple Regression

Let's build another regression model to solidify our understanding of regression models. Earlier on, we predict a future value of y given one input, x. Recall the y in our first example is the Profit and our x is the number of Sales. In the graded assignment example, our y and x are `inequality` and `gdp` respectively.

Also recall that because the number of profit *depend* on the number of sales, this `y` we're working with is often referred to as **dependent variable** while the x are referred to as, you guessed it, **independent variables**. Can a regression model contain more than one dependent variable? Absolutely!

Imagine you're hired as an intern to a city council that is planning for a few awareness-based hill races around the different hills and forests in West Java. The first question they're interested in: Given the **distance** and **elevation** of each future race, are you able to predict the race time (in minutes) for each of these hill races?

Turns out regression models are perfect for these kind of problems. Applying what we've learnt above to this new question, the first thing obviously is to collect some data and run a linear model function, `lm()` on it. While we don't have a record of past hill races in West Java, we will use a similar dataset called `hills` which is just a record of 35 Scottish hill races. The exercise while theoretical in nature, demonstrates yet another common application of regression models.

The `hills` dataset is shipped with the `MASS` library so let's load the library into our workspace and inspect the first 6 rows of the data:

```
library(MASS)
data(hills)
head(hills)
```

```
##              dist climb   time
## Greenmantle   2.5   650 16.083
## Carnethy      6.0  2500 48.350
## Craig Dunain  6.0   900 33.650
## Ben Rha       7.5   800 45.600
## Ben Lomond    8.0  3070 62.267
## Goatfell      8.0  2866 73.217
```

Observe that for each race, the dataset has collected the following attributes:
- `dist`: Distance in miles
- `climb`: Total height gained during the route, in feet
- `time`: Record time in minutes

Supposed we like to predict the required time to complete a race given the specified route distance and elevation in a race, we can regress `time` on `dist` and `climb`. Our formula can be specified the following way: `formula = time ~ dist + climb` or `formula = time ~ .`

When we use a ".", R will take all remaining variables (except the dependent variable) so I hope you can see that the two formulas are equivalent:

```
race.lm <- lm(formula = time ~ ., hills)
summary(race.lm)
```

```
##
## Call:
## lm(formula = time ~ ., data = hills)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.215  -7.129  -1.186   2.371  65.121
##
## Coefficients:
##              Estimate Std. Error t value         Pr(>|t|)
## (Intercept) -8.992039   4.302734  -2.090           0.0447 *
## dist         6.217956   0.601148  10.343 0.00000000000986 ***
## climb        0.011048   0.002051   5.387 0.00000644518298 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.68 on 32 degrees of freedom
## Multiple R-squared:  0.9191, Adjusted R-squared:  0.914
## F-statistic: 181.7 on 2 and 32 DF,  p-value: < 0.00000000000000022
```

```
# equiv:
# summary(lm(time ~ dist + climb, hills))
```

Looking at the multivariate model above, consider a regression where `Distance` and `Climb` enter linearly, our estimation could be written in the following:
Time = b1 + b2 x Distance + b3 x Climb

Supposed the race organizers inform us the next race is 9.65km (6 miles) in distance and has a total elevation of 701 meters (2300 foot), you'd substitute the values and have them multiplied by the coefficients:

Time = -8.99 + 6.218 x Distance + 0.01105 x Climb
Time = -8.99 + 6.218 x 6 + 0.01105 x 2300
Time = 53.733

```
-8.99 + 6.218 * 6 + 0.01105 * 2300
```

```
## [1] 53.733
```

```
predict(race.lm,
        data.frame(dist=6, climb=2300))
```

```
##        1
## 53.72589
```

We interpret b3 = 0.01105 minutes per foot as the slope of record time against Climb, allowing for the contribution of Distance. In this example 1 unit is not a useful amount of change in Climb. We would do better to restate b3 as 11.05 minutes per 1000 feet.

Supposed the next race is Sentul Hill Color Run and with the given information from our appointed race organizers: 9.65km (6 miles) and total height gained of 701 meters (2300 foot) during the route we can use a least squares regression to predict the record race time.

With the linear model, we can use the `predict()` function and obtain a prediction. We'll supply our `predict` function with the two parameters:
1. The model to predict with (`race.lm`)
2. The data on which the prediction is made

```
predict(race.lm, data.frame(cbind(dist=6, climb=2300)))
```

```
##        1
## 53.72589
```

...and you will still predict 53.73 minutes being the record time. This checks out with the manual calculation we performed by hand earlier.

```
summary(race.lm)
```

```
##
## Call:
## lm(formula = time ~ ., data = hills)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.215  -7.129  -1.186   2.371  65.121
##
## Coefficients:
##              Estimate Std. Error t value         Pr(>|t|)
## (Intercept) -8.992039   4.302734  -2.090           0.0447 *
## dist         6.217956   0.601148  10.343 0.00000000000986 ***
## climb        0.011048   0.002051   5.387 0.00000644518298 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.68 on 32 degrees of freedom
## Multiple R-squared:  0.9191, Adjusted R-squared:  0.914
## F-statistic: 181.7 on 2 and 32 DF,  p-value: < 0.00000000000000022
```

Adjusted R-Squared

Using R-squared itself can be misleading in our assessment of the model fit and this is due to the one of the key limitation of this metric. R-squared, it turns out increases with every new addition of a predictor variable, even if it turns out that the variable is just completely random number - the R-squared does not decrease. As a result, a model with more independent variables may appear to have a better fit just on the merit of having more terms alone.

A model that has too many predictors also tend to overfit and worse, the regression model would "model" the random noise in our data as if they were "features", hence producing misleading R-squared values.

The adjusted R-squared compares the explanatory power of regression models built with different number of predictors, allowing us to compare a crime rate regression model with 4 variables to another one with just 2 variables and find out if the one with 4 has achieved a higher R-squared simply because it has more predictors or if they truly lead to a better fit.

Compare the following 3 models and pay attention to it's Adjusted R-squared value. Model 1:

```
model1 <- lm(inequality ~ gdp, crime)
summary(model1)
```

```
##
```

```
## Call:
## lm(formula = inequality ~ gdp, data = crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.029 -11.754   1.714  12.438  30.006
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 386.03058   15.38651   25.09 <0.0000000000000002 ***
## gdp          -0.36551    0.02881  -12.69 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.86 on 45 degrees of freedom
## Multiple R-squared:  0.7815, Adjusted R-squared:  0.7766
## F-statistic: 160.9 on 1 and 45 DF,  p-value: < 0.00000000000000022
```

Model 2:

```
model2 <- lm(inequality ~ gdp + mean_education, crime)
summary(model2)
```

```
##
## Call:
## lm(formula = inequality ~ gdp + mean_education, data = crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -40.824  -7.585   0.672  13.344  30.179
##
## Coefficients:
##                 Estimate Std. Error t value             Pr(>|t|)
## (Intercept)    441.88243   25.44031  17.369 < 0.0000000000000002 ***
## gdp             -0.28713    0.03994  -7.189        0.00000000604 ***
## mean_education  -0.91851    0.34448  -2.666               0.0107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.69 on 44 degrees of freedom
## Multiple R-squared:  0.8119, Adjusted R-squared:  0.8033
## F-statistic: 94.93 on 2 and 44 DF,  p-value: < 0.00000000000000022
```

Model 3:

```
model3 <- lm(inequality ~ gdp + labour_participation + m_per1000f + time_prison, crime)
summary(model3)
```

```
##
## Call:
## lm(formula = inequality ~ gdp + labour_participation + m_per1000f +
##     time_prison, data = crime)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.231  -8.905   1.394  12.797  31.798
##
## Coefficients:
##                       Estimate Std. Error t value            Pr(>|t|)
## (Intercept)          305.05148  110.09696   2.771              0.0083 **
## gdp                   -0.36705    0.03038 -12.081 0.00000000000000297 ***
## labour_participation  -0.02186    0.08350  -0.262              0.7948
## m_per1000f             0.07681    0.12249   0.627              0.5340
## time_prison            0.69764    0.44046   1.584              0.1207
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.96 on 42 degrees of freedom
## Multiple R-squared:  0.7939, Adjusted R-squared:  0.7742
## F-statistic: 40.44 on 4 and 42 DF,  p-value: 0.00000000000006981
```

```
print("Multiple R-Squared")
```

```
## [1] "Multiple R-Squared"
```

```
summary(model1)$r.squared
```

```
## [1] 0.7814512
```

```
summary(model2)$r.squared
```

```
## [1] 0.8118523
```

```
summary(model3)$r.squared
```

```
## [1] 0.7938777
```

```
print("Adjusted R-Squared")
```

```
## [1] "Adjusted R-Squared"
```

```
summary(model1)$adj.r.squared
```

```
## [1] 0.7765945
```

```
summary(model2)$adj.r.squared
```

```
## [1] 0.8033001
```

```r
summary(model3)$adj.r.squared
```

```
## [1] 0.774247
```

`model1` has one predictor variable and has an R-squared of 0.7815 (adjusted R-squared of 0.7766). We see that by adding three additional predictors (`labor_participation`, `m_per1000f`, `time_prison`) the R-squared of our model increased to 0.7939 (`model3`). Now we say that R-squared indicates the quality of model fit, so does this necessarily means that `model3` is a better model than `model1`? Not really. In fact, by adding three additional parameters, our Adjusted R-Squared has decreased and returned a model that has a lower Adjusted R-squared than the two other models (despite built with more predictor variables).

So as a recap, our R-squared value tells us how well our model describes the data. It measures the extent to which the variance in our dependent variable (inequality) can be explained by the independent variables (gdp etc). However, as we increase the number of independent variables our model's R-squared value will also increase as it is incorporate any legitimate information as well as the noise introduced by these extra variables.

Adjusted R-squared on the other hand does not increases the way R-squared does because it is adjusted for the number of predictor variables in our model. It increases only when the new variable actually leads to a better prediction. While the mathematical details of the adjusted R-squared formula is beyond the scope of this workshop, I'll give you a quick proof that it does "penalizes" the r-squared based on the number of predictors the model contains.

The mathematical notation of adjusted R-squared: $R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$

Where n is the number of observations and p is the number of predictors. Notice that as p increases, the second term becomes larger and pushing the overall adjusted R-squared value down. If it isn't obvious, I included the following code so you can play around by changing the number of predictors (we used 4 in the model):

```r
no_of_preds <- 4
adjusted_model3 <- 1 - (1-summary(model3)$r.squared) * (nrow(crime)-1)/(nrow(crime)-1-no_of_preds)

adjusted_model3
```

```
## [1] 0.774247
```

```r
summary(model3)$adj.r.squared
```

```
## [1] 0.774247
```

Discussion

In the following linear model, I deliberately fitted a poor regression model. Can you describe / explain why the following regression model is a poor fit?

```r
summary(lm(inequality ~ labour_participation + m_per1000f + time_prison, crime))
```

```
##
## Call:
## lm(formula = inequality ~ labour_participation + m_per1000f +
##     time_prison, data = crime)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -69.632 -29.102   1.083  32.332  77.622
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          334.29002  230.12013   1.453    0.154
## labour_participation  -0.25552    0.16982  -1.505    0.140
## m_per1000f            -0.00713    0.25567  -0.028    0.978
## time_prison            0.38038    0.91922   0.414    0.681
##
## Residual standard error: 39.63 on 43 degrees of freedom
## Multiple R-squared:  0.07761,    Adjusted R-squared:  0.01326
## F-statistic: 1.206 on 3 and 43 DF,  p-value: 0.319
```

Confidence Interval

Let's go back to the sales predictor we made at the beginning of this class. We named it `ols` and we learned how to use `predict()` to predict on a new value. Turns out when we use `predict()` on a linear model, we can also specify a particular interval like the following:

```
predict(ols, data.frame(Sales=6000), interval="confidence", level=0.9)
```

```
##        fit      lwr      upr
## 1 2283.067 2134.136 2431.998
```

```
predict(ols, data.frame(Sales=6000), interval="confidence", level=0.95)
```

```
##        fit      lwr      upr
## 1 2283.067 2104.749 2461.385
```

When we create a linear model prediction with a confidence interval, we are asking for a range (`lwr` and `upr`) to help us determine what the true profit (an unknown parameter) could be. This interval (the range between `lwr` and `upr`) is derived from the sample and will contain the true unknown parameter in 95% of the possible samples.

Now in practice, it is very costly or even impossible to know a true population's parameters (imagine the work it takes to find the true mean of the "height" of the Indonesian male population - it would take years and even then it is arguably not the true value of the mean because every minute there are newborn babies, deaths etc), statisticians resort to estimating the true population parameters using sampling. In a business context, imagine the daunting if not impossible task of obtaining the true "discount sensitivity" to a certain facial product. This statistic require us to survey every possible household / individual within the studied population, upon which we calculate a certain average.

What's far more achievable and realistic is to survey random set of individuals that visit our stores each day and therefore extrapolate from these samples to obtain a sense of how "sensitive" the population will be.

For each sample mean, we calculate a confidence interval of the mean. If we repeat this process enough times, we'd expect about 95% of these intervals to include the true value of the population mean. Essentially, we're quantifying the accuracy of our estimation.

[Optional] Understanding confidence interval

```
library(MASS)
height <- na.omit(survey$Height)
height.se <- sd(height)/ sqrt(length(height))
height.se
```

```
## [1] 0.6811677
```

```
qt(.975, df=length(height)-1)
```

```
## [1] 1.971435
```

```
height.conf <- mean(height) + c(-1,1) * qt(.975, df=length(height)-1) * height.se
height.conf
```

```
## [1] 171.0380 173.7237
```

Without knowing the true population parameters (such as its mean and sd) we can construct a confidence interval at a 95% confidence level.

Without going through the above exercise, we could easily make use of the R's `t.test` function to obtain the 95% confidence interval too:

```
t.test(height)
```

```
##
##  One Sample t-test
##
## data:  height
## t = 253.07, df = 208, p-value < 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  171.0380 173.7237
## sample estimates:
## mean of x
##  172.3809
```

Notice we get the same 95% confidence interval. Say we're interested in finding the 95% confidence interval for our slope, we can do so using the `confint`:

```
confint(ols)
```

```
##                    2.5 %     97.5 %
## (Intercept) -144.9186102 -22.177626
## Sales          0.3570237   0.431848
```

This gives us the same confidence interval have we calculated it manually:

```r
summary(ols)$coefficients
```

```
##                Estimate Std. Error   t value
## (Intercept) -83.5481183 30.6806773 -2.723151
## Sales         0.3944358  0.0187033 21.089103
##                                        Pr(>|t|)
## (Intercept) 0.008455272733379431623901290038247 68
## Sales       0.000000000000000000000000000001949109
```

As an estimation, we could also use the estimated coefficient (1.462) +/- two standard errors (0.021) and that would approximate to our 95% confidence interval.

```r
0.3944358 + c(-1,1) * 2 * 0.02145512
```

```
## [1] 0.3515256 0.4373460
```

Prediction interval, on the other hand, is an estimate of an interval in which future observations will fall given what has already been observed. Where a confidence interval is concerned with the distribution of estimates of a true population parameter (such as mean height), the prediction interval is concerned with the distribution of individual future points (i.e. the value of the next sample variable $X_n + 1$).

```r
predict(ols, newdata=data.frame(Sales = 6000), interval="prediction")
```

```
##        fit      lwr     upr
## 1 2283.067 1949.074 2617.06
```

Step-wise Regression for Feature Selection

```r
crime$police_exp <- crime$police_exp59 + crime$police_exp60
crime_s <- subset(crime, select=-c(police_exp59, police_exp60))
```

Stepwise regression is a set of methods to fit regression models where the choice of predictive variables is carried out by an automatic procedure. On each step, one variable is considered for addition to or subtraction from the set of variables in the current model - and this process repeats itself until the model cannot be further improved upon.

A common estimator of the relative quality of statistical models is the AIC (Akaike information criterion). The formula of the AIC is beyond the scope of this literature, but simply put the formula rewards goodness of fit while penalizing for number of estimated parameters. Akaike's work demonstrated how we can estimate the amount of "information loss" from one model to another.

Backward elimination

Backward elimination works by starting with all candidate variables, testing the deletion of each variable using a chosen model fit criterion (AIC or residual sum of squares) and then removing the variable whose absence gives the highest reduction in AIC (biggest improvement in model fit) and repeating this process until no further variables can be removed without a loss of fit.

```r
lm.all <- lm(inequality ~., crime_s)
step(lm.all, direction="backward")
```

```
## Start:  AIC=261.43
## inequality ~ percent_m + is_south + mean_education + labour_participation +
##     m_per1000f + state_pop + nonwhites_per1000 + unemploy_m24 +
##     unemploy_m39 + gdp + prob_prison + time_prison + crime_rate +
##     police_exp
##
##                         Df Sum of Sq    RSS    AIC
## - time_prison            1       0.6 6466.3 259.44
## - nonwhites_per1000      1       3.5 6469.3 259.46
## - prob_prison            1      12.6 6478.3 259.52
## - unemploy_m24           1      20.1 6485.9 259.58
## - unemploy_m39           1      51.4 6517.1 259.81
## - m_per1000f             1      64.7 6530.4 259.90
## - labour_participation   1     266.0 6731.7 261.33
## <none>                               6465.7 261.43
## - percent_m              1     336.5 6802.2 261.82
## - state_pop              1     491.0 6956.7 262.87
## - is_south               1     776.4 7242.2 264.76
## - police_exp             1     830.2 7295.9 265.11
## - mean_education         1    1205.0 7670.7 267.47
## - crime_rate             1    2069.9 8535.6 272.49
## - gdp                    1    3483.6 9949.3 279.69
##
## Step:  AIC=259.44
## inequality ~ percent_m + is_south + mean_education + labour_participation +
##     m_per1000f + state_pop + nonwhites_per1000 + unemploy_m24 +
##     unemploy_m39 + gdp + prob_prison + crime_rate + police_exp
##
##                         Df Sum of Sq     RSS    AIC
## - nonwhites_per1000      1       4.5  6470.9 257.47
## - prob_prison            1      13.3  6479.6 257.54
## - unemploy_m24           1      19.6  6485.9 257.58
## - unemploy_m39           1      50.8  6517.1 257.81
## - m_per1000f             1      64.3  6530.6 257.90
## - labour_participation   1     266.2  6732.5 259.33
## <none>                               6466.3 259.44
## - percent_m              1     342.0  6808.3 259.86
## - state_pop              1     538.3  7004.7 261.20
## - is_south               1     783.3  7249.7 262.81
## - police_exp             1     876.2  7342.5 263.41
## - mean_education         1    1222.5  7688.9 265.58
## - crime_rate             1    2069.7  8536.1 270.49
## - gdp                    1    3537.5 10003.8 277.95
##
## Step:  AIC=257.47
## inequality ~ percent_m + is_south + mean_education + labour_participation +
##     m_per1000f + state_pop + unemploy_m24 + unemploy_m39 + gdp +
##     prob_prison + crime_rate + police_exp
##
##                         Df Sum of Sq     RSS    AIC
```

```
## - prob_prison           1      16.8  6487.7 255.59
## - unemploy_m24          1      23.6  6494.5 255.64
## - unemploy_m39          1      51.7  6522.6 255.84
## - m_per1000f            1      59.7  6530.6 255.90
## <none>                               6470.9 257.47
## - labour_participation  1     321.1  6792.0 257.75
## - percent_m             1     350.8  6821.6 257.95
## - state_pop             1     543.0  7013.8 259.26
## - police_exp            1     942.9  7413.8 261.87
## - is_south              1    1037.8  7508.6 262.46
## - mean_education        1    1294.9  7765.8 264.05
## - crime_rate            1    2099.0  8569.8 268.68
## - gdp                   1    3856.1 10327.0 277.44
##
## Step:  AIC=255.59
## inequality ~ percent_m + is_south + mean_education + labour_participation +
##     m_per1000f + state_pop + unemploy_m24 + unemploy_m39 + gdp +
##     crime_rate + police_exp
##
##                         Df Sum of Sq     RSS    AIC
## - unemploy_m24           1      23.7  6511.4 253.76
## - unemploy_m39           1      49.1  6536.8 253.95
## - m_per1000f             1      65.0  6552.7 254.06
## <none>                               6487.7 255.59
## - labour_participation   1     324.5  6812.2 255.89
## - percent_m              1     351.8  6839.5 256.07
## - state_pop              1     526.2  7013.8 257.26
## - police_exp             1     932.1  7419.8 259.90
## - mean_education         1    1282.2  7769.9 262.07
## - is_south               1    1387.6  7875.2 262.70
## - crime_rate             1    2277.4  8765.1 267.73
## - gdp                    1    4263.4 10751.1 277.33
##
## Step:  AIC=253.76
## inequality ~ percent_m + is_south + mean_education + labour_participation +
##     m_per1000f + state_pop + unemploy_m39 + gdp + crime_rate +
##     police_exp
##
##                         Df Sum of Sq     RSS    AIC
## - unemploy_m39           1      26.7  6538.1 251.96
## - m_per1000f             1     175.7  6687.0 253.02
## <none>                               6511.4 253.76
## - labour_participation   1     307.6  6819.0 253.93
## - percent_m              1     347.3  6858.6 254.21
## - state_pop              1     576.5  7087.8 255.75
## - police_exp             1    1014.3  7525.7 258.57
## - mean_education         1    1288.6  7800.0 260.25
## - is_south               1    1442.0  7953.4 261.17
## - crime_rate             1    2294.6  8806.0 265.95
## - gdp                    1    4514.5 11025.8 276.52
##
## Step:  AIC=251.96
## inequality ~ percent_m + is_south + mean_education + labour_participation +
##     m_per1000f + state_pop + gdp + crime_rate + police_exp
```

```
##
##                         Df Sum of Sq     RSS    AIC
## - m_per1000f             1     149.0  6687.1 251.02
## <none>                                 6538.1 251.96
## - percent_m              1     327.2  6865.3 252.25
## - labour_participation   1     526.4  7064.5 253.60
## - state_pop              1     549.8  7087.9 253.75
## - police_exp             1     987.7  7525.8 256.57
## - mean_education         1    1341.2  7879.3 258.73
## - is_south               1    1495.0  8033.1 259.64
## - crime_rate             1    2354.2  8892.3 264.41
## - gdp                    1    4664.6 11202.7 275.27
##
## Step:  AIC=251.02
## inequality ~ percent_m + is_south + mean_education + labour_participation +
##     state_pop + gdp + crime_rate + police_exp
##
##                         Df Sum of Sq     RSS    AIC
## <none>                                 6687.1 251.02
## - percent_m              1     320.6  7007.6 251.22
## - state_pop              1     407.6  7094.7 251.80
## - labour_participation   1     744.7  7431.7 253.98
## - police_exp             1    1041.7  7728.8 255.82
## - mean_education         1    1257.2  7944.2 257.11
## - is_south               1    1422.4  8109.5 258.08
## - crime_rate             1    2900.8  9587.9 265.95
## - gdp                    1    4663.8 11350.9 273.88

##
## Call:
## lm(formula = inequality ~ percent_m + is_south + mean_education +
##     labour_participation + state_pop + gdp + crime_rate + police_exp,
##     data = crime_s)
##
## Coefficients:
##          (Intercept)             percent_m               is_south
##            383.13849              -0.31896              19.04721
##       mean_education  labour_participation             state_pop
##             -0.90375               0.13285               0.10035
##                  gdp            crime_rate            police_exp
##             -0.25013               0.03238              -0.19414
```

```
summary(lm(formula = inequality ~ percent_m + is_south + mean_education +
    labour_participation + state_pop + gdp + crime_rate + police_exp,
    data = crime_s))
```

```
##
## Call:
## lm(formula = inequality ~ percent_m + is_south + mean_education +
##     labour_participation + state_pop + gdp + crime_rate + police_exp,
##     data = crime_s)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -24.549  -7.471  -0.636   9.342  30.317
##
## Coefficients:
##                         Estimate Std. Error t value     Pr(>|t|)
## (Intercept)          383.138492  55.426156   6.913 0.0000000323 ***
## percent_m             -0.318964   0.236329  -1.350     0.185111
## is_south              19.047208   6.699475   2.843     0.007152 **
## mean_education        -0.903748   0.338127  -2.673     0.011021 *
## labour_participation   0.132854   0.064583   2.057     0.046586 *
## state_pop              0.100351   0.065938   1.522     0.136311
## gdp                   -0.250129   0.048587  -5.148 0.0000083525 ***
## crime_rate             0.032382   0.007976   4.060     0.000236 ***
## police_exp            -0.194138   0.079792  -2.433     0.019787 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.27 on 38 degrees of freedom
## Multiple R-squared:  0.9087, Adjusted R-squared:  0.8894
## F-statistic: 47.26 on 8 and 38 DF,  p-value: < 0.00000000000000022
```

```
lm.none <- lm(inequality ~ 1, crime_s)
# Foward step-wise regression
step(lm.none, scope=list(lower=lm.none, upper=lm.all), direction="forward")
# Step-wise regression by default chooses both-direction
step(lm.none, scope = list(upper=lm.all), data=crime, direction="both")
```

```
summary(lm(formula = inequality ~ gdp + crime_rate + mean_education +
    police_exp + is_south + labour_participation + state_pop +
    percent_m, data = crime_s)
)
```

```
##
## Call:
## lm(formula = inequality ~ gdp + crime_rate + mean_education +
##     police_exp + is_south + labour_participation + state_pop +
##     percent_m, data = crime_s)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.549  -7.471  -0.636   9.342  30.317
##
## Coefficients:
##                         Estimate Std. Error t value     Pr(>|t|)
## (Intercept)          383.138492  55.426156   6.913 0.0000000323 ***
## gdp                   -0.250129   0.048587  -5.148 0.0000083525 ***
## crime_rate             0.032382   0.007976   4.060     0.000236 ***
## mean_education        -0.903748   0.338127  -2.673     0.011021 *
## police_exp            -0.194138   0.079792  -2.433     0.019787 *
## is_south              19.047208   6.699475   2.843     0.007152 **
## labour_participation   0.132854   0.064583   2.057     0.046586 *
## state_pop              0.100351   0.065938   1.522     0.136311
## percent_m             -0.318964   0.236329  -1.350     0.185111
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.27 on 38 degrees of freedom
## Multiple R-squared:  0.9087, Adjusted R-squared:  0.8894
## F-statistic: 47.26 on 8 and 38 DF,  p-value: < 0.00000000000000022
```

In the forward step-wise regression, we're starting from the `lm.none` model (`inequality ~ 1`) and searching through models lying in the range between that and the `lm.all` model (using all candidate variables).

When we specify `direction='both'`, we're getting the algorithm to start from `lm.none` while progressively adding or removing each candidate variable before recording the AIC improvements and residual sum of squares. Because it compares the improvement it gets from dropping / adding each candidate variable from the current model, it can identify the variable that leads to the best AIC improvement. Printing `summary()` on the step-wise regression model will illustrate this algorithm's process[4].

Notice that in this case, the forward selection and stepwise regression strategy leads us to a model that is equivalent to the one we obtained from backward elimination. Let's inspect the model's R-squared value and see if it was able to explain the variance in inequality better than the ones we created without automatic feature selection methods:

```
summary(lm(formula = inequality ~ is_south + mean_education +
    labour_participation + gdp + crime_rate + police_exp,
    data = crime_s))
```

```
##
## Call:
## lm(formula = inequality ~ is_south + mean_education + labour_participation +
##     gdp + crime_rate + police_exp, data = crime_s)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.969  -8.182   0.143   9.970  24.564
##
## Coefficients:
##                       Estimate Std. Error t value       Pr(>|t|)
## (Intercept)         344.737004  40.834421   8.442 0.000000000201 ***
## is_south             16.381935   6.761602   2.423       0.020022 *
## mean_education       -1.019212   0.333988  -3.052       0.004031 **
## labour_participation  0.121705   0.066519   1.830       0.074767 .
## gdp                  -0.231985   0.048926  -4.742 0.000026876562 ***
## crime_rate            0.029137   0.007895   3.691       0.000666 ***
## police_exp           -0.129733   0.076562  -1.694       0.097948 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.75 on 40 degrees of freedom
## Multiple R-squared:  0.8967, Adjusted R-squared:  0.8812
## F-statistic: 57.85 on 6 and 40 DF,  p-value: < 0.00000000000000022
```

I'd like to introduce you to another variable selection method: "all-possible-regressions". With this method we're considering all possible subsets of the pool explanatory variables and finds the model that best fits the

[4] J.M. Chambers, T.J.Hastie, Statistical Models in S

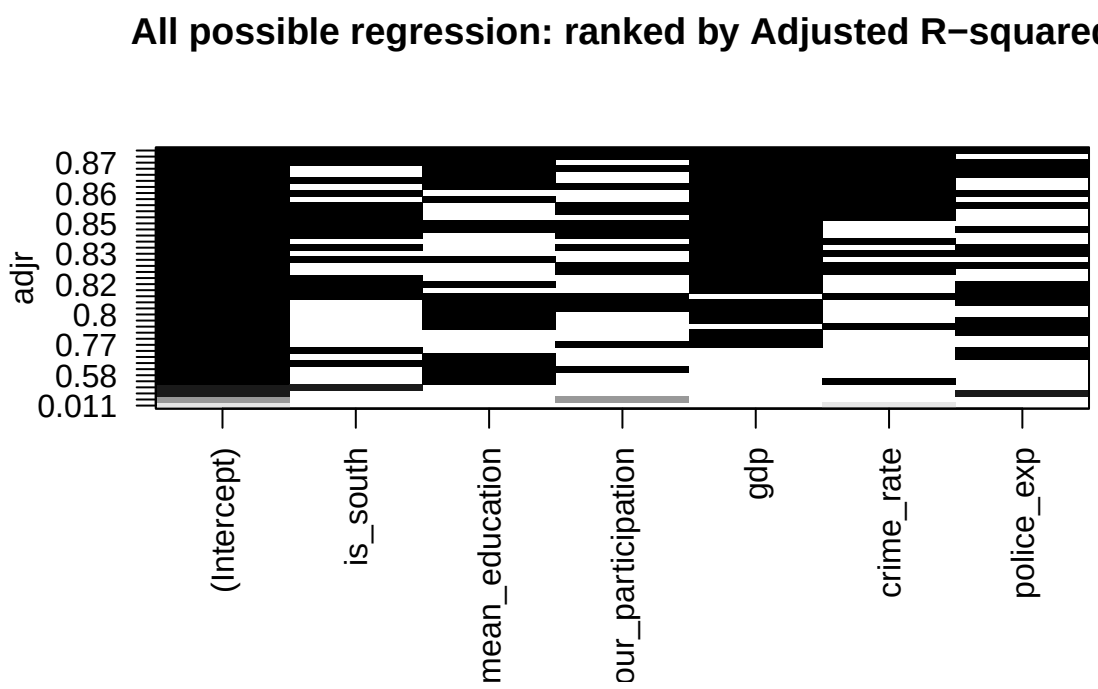
data according to a specified criteria, which can be adjusted r-squared or AIC/BIC. The `nbest` parameter also allow us to specify the number of subsets of each size to record.

```
library(leaps)
regs <- regsubsets(inequality ~ is_south + mean_education +
    labour_participation + gdp + crime_rate + police_exp, crime_s, nbest=10)
```

```
library(dplyr)
```

We can use `summary(regs)` to see the exhaustive algorithm at work, including all the different combination of subsets it has considered. However, I am going to just visualize the all-possible-regression and observe the adjusted r-squared value for each possible subsets:

```
plot(regs, scale="adjr", main="All possible regression: ranked by Adjusted R-squared")
```



Seeing how the `police_exp` doesn't significantly reduce our adjusted R-squared, we're going to drop that variable and save this final model as `model4`:

```
model4 <- lm(formula = inequality ~ is_south + mean_education +
    labour_participation + gdp + crime_rate,
    data = crime_s)
summary(model4)
```

```
##
## Call:
## lm(formula = inequality ~ is_south + mean_education + labour_participation +
##     gdp + crime_rate, data = crime_s)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.220  -8.532   1.294   9.020  28.844
```

```
##
## Coefficients:
##                          Estimate Std. Error t value       Pr(>|t|)
## (Intercept)           338.118733  41.564425   8.135 0.000000000433 ***
## is_south               16.876729   6.907731   2.443        0.01895 *
## mean_education         -0.938677   0.338049  -2.777        0.00824 **
## labour_participation    0.144489   0.066616   2.169        0.03594 *
## gdp                    -0.286901   0.037480  -7.655 0.000000001992 ***
## crime_rate              0.020930   0.006375   3.283        0.00210 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.06 on 41 degrees of freedom
## Multiple R-squared:  0.8893, Adjusted R-squared:  0.8757
## F-statistic: 65.84 on 5 and 41 DF,  p-value: < 0.00000000000000022
```

When dealing with regression analysis of a different type: one where the dependent variable is continuous but the independent variables are categorical, we would use a different test such as the ANOVA and t.test. You have learned about the t-test in your Practical Statistics course, and I will expose you to the idea of ANOVA as well as other type of tests in future workshops of the Academy (Machine Learning Specialization).

Evaluating a Regression Model

In some cases, you might end up with more than one model for you to choose. It is often important for us to evaluate the performance of each model to find out which of the models result in the least total error. On the next chapters of machine learning specialization, we will dwell into an in-depth discussion on various evaluation metrics.

Mean Absolute Error (MAE)

Mean Absolute Error (MAE) measures the average of the absolute differences between observed data points and predicted values. In mathematical notation:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |Predicted_i - Actual_i|$$

As this metric computes the absolute difference, MAE has the same unit as the actual data. Therefore, to understand the result of MAE, it is necessary to compare its value with the range of the actual data.

```
mean(abs(model3$residuals))
```

```
## [1] 14.39466
```

```
mean(abs(model4$residuals))
```

```
## [1] 10.40482
```

```
# actual data range
range(crime$inequality)
```

```
## [1] 126 276
```

```r
range(crime_s$inequality)
```

```
## [1] 126 276
```

`model4` demonstrated the best performance in predicting inequality given the predictors, compared to `model3`. `model4` yielded the lower MAE value. Interpreting this in conjunction with the actual data range, it can be concluded that the error rate is relatively small.

Alternatively, MAE can be compute with `MAE()` function from `MLmetrics`.

```r
MAE(y_true = crime$inequality, y_pred = predict(model3, crime))
```

```
## [1] 14.39466
```

```r
MAE(y_true = crime_s$inequality, y_pred = predict(model4, crime_s))
```

```
## [1] 10.40482
```

Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) evaluates model performance by comparing the actual error to the actual value in terms of percentage, and then computes its average across the data. In mathematical formulation:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|Predicted_i - Actual_i|}{Actual_i}$$

As MAPE is expressed in terms of percentage, it's easier for us to understand and interpret the MAPE result. However, MAPE cannot be utilized if the actual data contains zeros, as it will cause an error during the calculation. We can compute MAPE using the `MAPE()` function from `MLmetrics`.

```r
MAPE(y_true = crime$inequality, y_pred = predict(model3, crime))
```

```
## [1] 0.07797005
```

```r
MAPE(y_true = crime_s$inequality, y_pred = predict(model4, crime_s))
```

```
## [1] 0.05469714
```

Root Mean Squared Error (RMSE)

A rather common measure we've introduced earlier in this course is the Root Mean Squared Error (RMSE). It measures the root mean square of observed data points to its predicted values. In other terms, RMSE is actually the square root of Mean Square Error (MSE). However, since the result of MSE is in quadratic terms, it's harder for us to interpret the result. Thus, RMSE is preferable and easier to use compared to MSE. The mathematical notation of RMSE is:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_i^n (Predicted_i - Actual_i)^2}{n}}$$

Similar to MAE, in order to interpret the RMSE value, we need to check the range of the actual data.

```r
sqrt(mean((model3$residuals^2)))
```

```
## [1] 17.91936
```

```r
sqrt(mean((model4$residuals^2)))
```

```
## [1] 13.13497
```

```r
# actual data range
range(crime$inequality)
```

```
## [1] 126 276
```

```r
range(crime_s$inequality)
```

```
## [1] 126 276
```

As seen in RMSE for `model3` and `model4`, we observe that `model4` did a better job in summarizing `inequality` given its predictors as it has a lower RMSE value. The RMSE value is relatively low compared to the range of the inequality data.

Alternatively, we can compute RMSE with `RMSE()` from `MLmetrics`.

```r
RMSE(y_true = crime$inequality, y_pred = predict(model3, crime))
```

```
## [1] 17.91936
```

```r
RMSE(y_true = crime_s$inequality, y_pred = predict(model4, crime_s))
```

```
## [1] 13.13497
```

[Additional] Best Practices for Model Evaluation

> Detailed explanation of the workflow and technical steps will be discussed in the material for Classification 1.

From the examples provided above, the MAE, MAPE, MSE, and RMSE methods only evaluate the data used to train the linear regression model. This, of course, raises a question: `can the evaluation results really indicate whether the model is ready to be used or not?` Considering that it has only been tested with the exact same data used for training.

To properly evaluate any machine learning model, ideally, the trained model should be evaluated by making predictions on data that was not used during the training process, commonly referred to as test data.

Typically, the entire dataset with labeled targets is not used directly for training the model. Instead, it undergoes a data splitting process into **training data (used to train the model)** and **test data (used to assess the model's performance)**. This process is often referred as `Cross Validation`.

This time, we will provide an example of how to evaluate the performance of a linear regression model using dummy test data.

```r
# Data Test
set.seed(123)
data_test <- data.frame(
  is_south = sample(0:1, 20, replace = TRUE),
  mean_education = sample(80:125, 10),
  labour_participation = sample(450:650, 10),
  gdp = sample(270:690, 10),
  crime_rate = sample(340:2000, 10),
  inequality = sample(125:280, 10)
)
head(data_test)
```

```
##   is_south mean_education labour_participation gdp crime_rate inequality
## 1        0            107                   527 662        556        162
## 2        0             88                   530 599       1653        145
## 3        0            108                   492 292        920        165
## 4        1            114                   552 680        411        214
## 5        0             87                   566 578        927        184
## 6        1            105                   525 404       1938        140
```

```r
# Evaluation Data Train
MAE(y_true = crime_s$inequality, y_pred = predict(model4, crime_s))
```

```
## [1] 10.40482
```

```r
# Evaluation Data Test
MAE(y_true = crime_s$inequality, y_pred = predict(model4, data_test))
```

```
## [1] 45.6402
```

```r
# actual data range
range(crime_s$inequality)
```

```
## [1] 126 276
```

```r
range(data_test$inequality)
```

```
## [1] 130 240
```

From the observations above, it can be seen that the trained model has a MAE value of 10 when using the data it was trained on, but has a larger error when used to predict data it has never seen before. **The more reliable evaluation results are those obtained when using test data or the data that the model has never seen before, as this better reflects the model's performance when it is eventually implemented**.

Once again, a detailed explanation of the workflow and technical steps will be discussed in the material for Classification 1.

Limitations of Regression Models

Linear regression models, even when considered to be the powerhouse of statistics came with its limitations and assumptions.

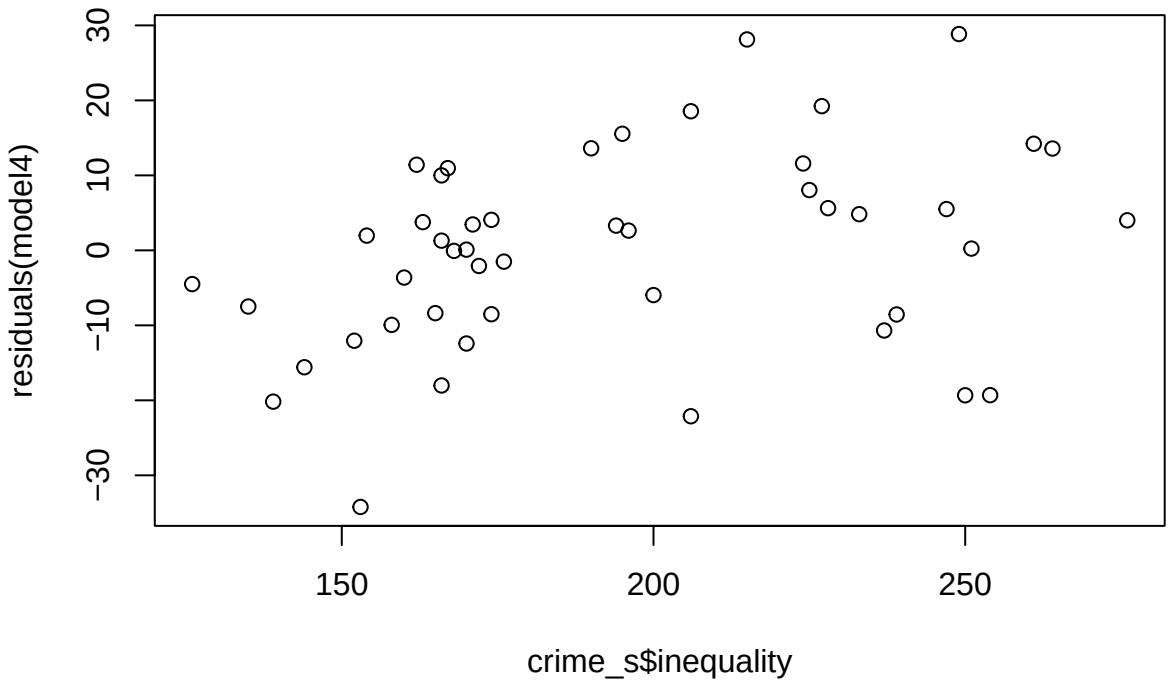
- Linear regressions are best fitted on data where a linear relationship between the predictor variables and target exist
- Simple / Multiple regression models can be sensitive to outliers (recall the chapter regarding leverage and power)
- Simple / Multiple regression models assumes that the independent variables are not highly correlated with each other (hence using the `police_exp` to capture the information from both `police_exp60` and `police_exp59`)

Residual Plot

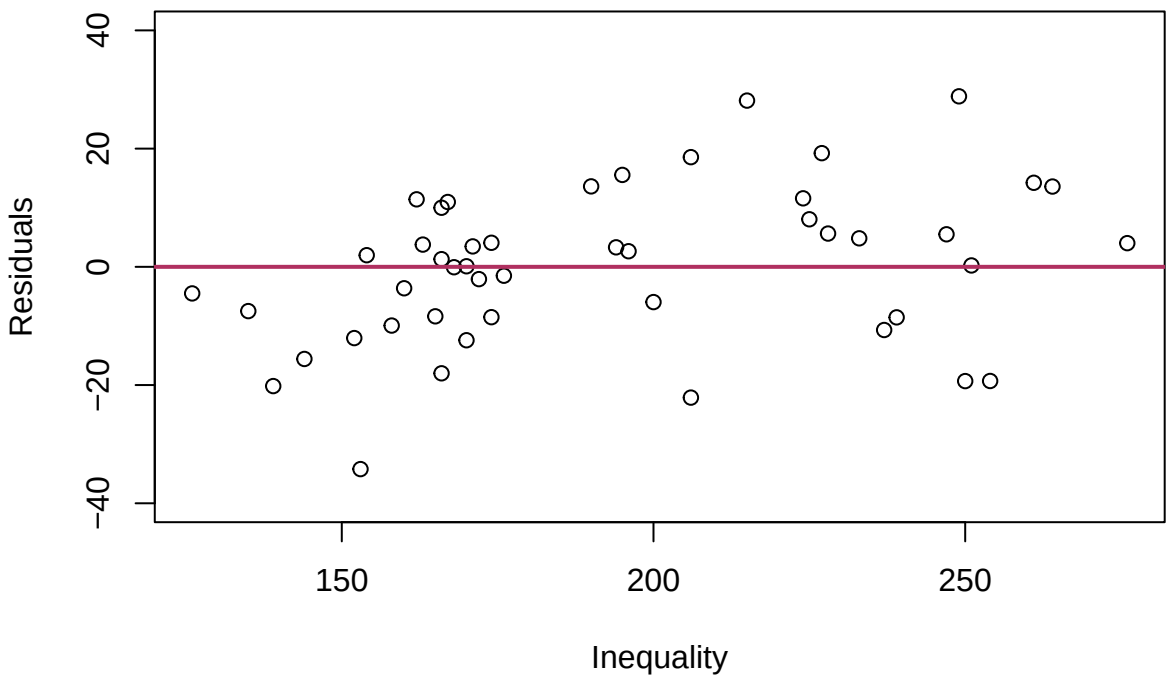
Before we conclude on regression models, I want to touch on yet another important aspect of model evaluation: the **residual plot**. Recall that residuals is the difference between the observed data and the fitted values. A residual plot is a plot that plot the residual values on the y-axis and the fitted values on the x-axis. Once we've done that, we look for any unwanted patterns in the plot that may indicate a violation in model assumptions particularly relating to incorrect specifications.

A residual plot that has points randomly scattered around the x-axis is the one we want. It doesn't mean that the model is perfect, but it does mean that the regression model you fit appropriately describes the variability in our dependent variable. If there is a pattern in the residual plot, it means that the model can be further improved upon or that it does not meet the linearity assumption.

```
plot(crime_s$inequality,residuals(model4))
```



```r
plot(crime$inequality,
  residuals(model4), ylim=c(-40,40), ylab="Residuals", xlab="Inequality")
abline(h = 0, col="maroon", lwd=2)
```



This residual plot do not show any non-random pattern, and at this point suffice to say we can put the model specifications into production use knowing that it can't be improved upon any further.

Assumption of homoscedasticity

For a more objective approach, a common statistical test is to check for heteroscedasticity using the Breusch-Pagan test. Heteroscedasticity is a condition where the variability of a variable is unequal across its range of value. In a linear regression model, if the variance of its error is showing unequal variation across the target variable range, it shows that heteroscedasticity is present and the implication to that is related to the previous statement of a non-random pattern in residual. Now let's use Breusch-Pagan test to our last model:

```r
library(lmtest)
bptest(model4)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model4
## BP = 4.5566, df = 5, p-value = 0.4723
```

Breusch-Pagan hypothesis test:

- H0: Variation of residual is constant (Homoscedasticity)
- H1: Variation of residual is not constant (Heteroscedasticity)

The test has a p-value above the significance level of 0.05, therefore we fail to reject the null hypothesis. We can conclude that the residuals has a constant variance and the assumption is passed.

Assumption of Normality

The normality assumption means that the residuals from the linear regression model should be normally distributed because we expect to get residuals near the zero value. To test for this condition, we can use the formal Shapiro-Wilk test to our residual:

```
shapiro.test(model4$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model4$residuals
## W = 0.9894, p-value = 0.9433
```

Shapiro-Wilk hypothesis test:

- H0: Residuals are normally distributed
- H1: Residuals are not normally distributed

The test has a p-value above the significance level of 0.05, therefore we fail to reject the null hypothesis. We can conclude that the residuals are normally distributed.

In the following experiment, we will take a look at a residual plot that display a non-random pattern, indicating to us that the model has not fully captured all "explainable information". Notice that the `summary` itself points to a R-squared, Adjusted R-squared even residuals summary that look pretty reasonable.

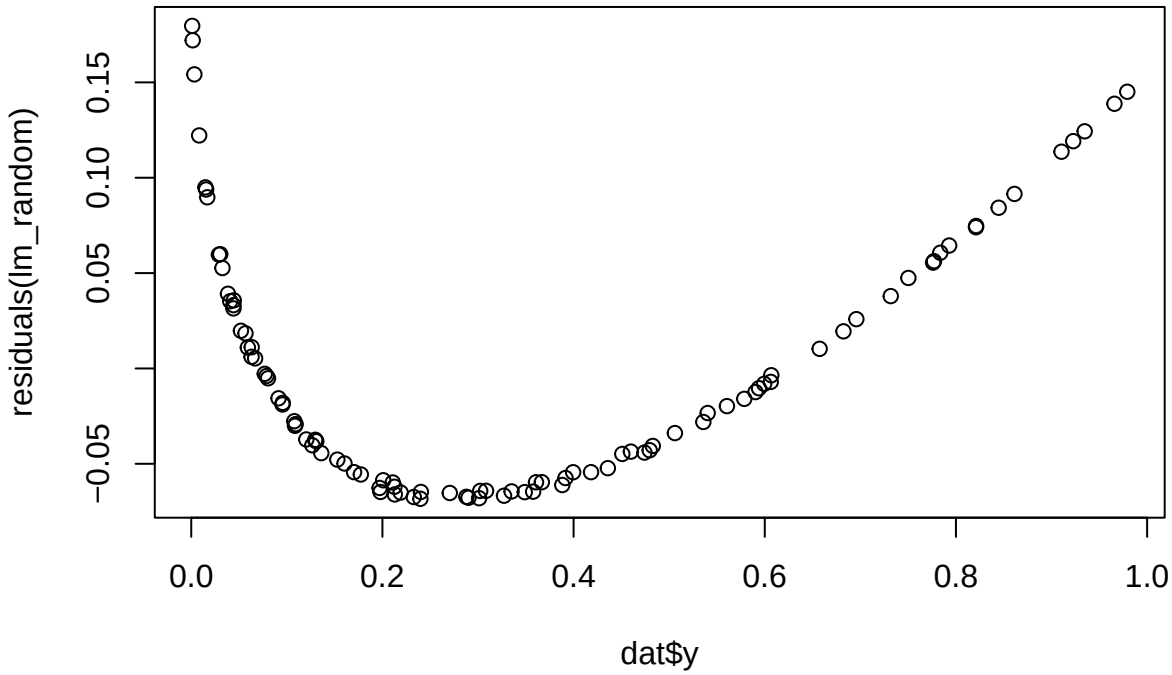
```
set.seed(100)
x <- runif(100)
y <- x ^ (runif(100,1.98,2.02))
dat <- data.frame(x, y)
lm_random <- lm(y ~ x, dat)
summary(lm_random)
```

```
##
## Call:
## lm(formula = y ~ x, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.06834 -0.05472 -0.01847  0.03824  0.17963
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) -0.21050    0.01452  -14.49 <0.0000000000000002 ***
## x            1.05566    0.02497   42.27 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.06511 on 98 degrees of freedom
## Multiple R-squared:  0.948,  Adjusted R-squared:  0.9475
## F-statistic:  1787 on 1 and 98 DF,  p-value: < 0.00000000000000022
```

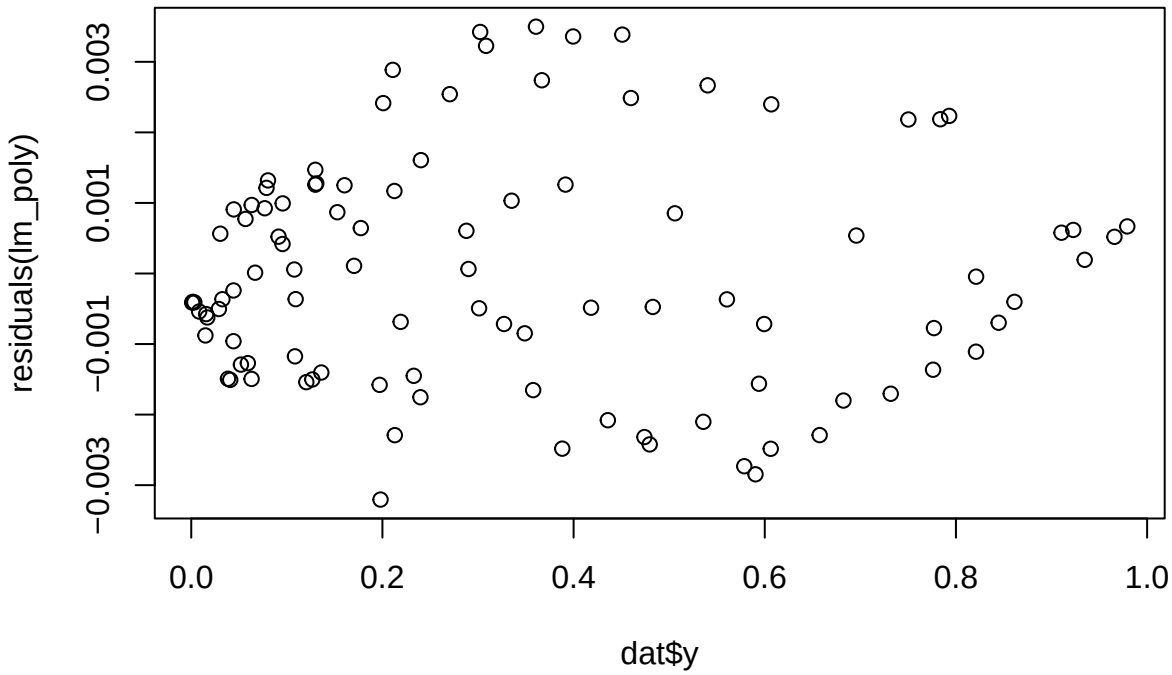
However, the non-random pattern becomes apparent when we plot the residuals plot:

```
plot(dat$y, residuals(lm_random))
```



If we have specified the model differently, by capturing all possible information in the variability of y, our residual plot would have display something like the following:

```
lm_poly<- lm(y ~ I(x^2), dat)
plot(dat$y, residuals(lm_poly))
```



I hope through the examples above and from the earlier experiment, it's clear why we want to see randomness in our residuals: we must have explained everything that is possible with our predictors so that only random errors is leftover as residuals. When there's a non-random pattern, it means our model can be further improved upon.

Multicollinearity

One of the statistical tool you have at your disposal when assessing multicollinearity is the **Variance Inflation Factor** (VIF). Put simply, VIF is a way to measure the effect of multicollinearity among the predictors in our model; One library to help us compute the VIF is the `car` package (car in this case is short for "Companion to Applied Regression"), and you simply call `vif()` on the model:

```
library(car)
vif(model4)
```

```
##            is_south       mean_education labour_participation
##            2.546140             3.326385             1.685616
##                 gdp          crime_rate
##            3.041970             1.413859
```

Manual Calculation of VIFs

VIF measures how much the variance of an estimated regression coefficient increases if your predictors are correlated. Because we want our prediction to be precise, low variance is the ideal.

Mathematically, the VIF works out to be the score we get when we take the prediction of interest and regress in against the rest of the predictors in our model, before applying it to the following formula:
$VIF = \frac{1}{1-R^2(x)}$

```
vif_man_crime <-
summary(lm(crime_rate ~ is_south + mean_education + labour_participation + gdp, crime_s))$r.squared
1/(1-vif_man_crime)
```

```
## [1] 1.413859
```

```
vif_man_edu <-
summary(lm(mean_education ~ is_south + crime_rate + labour_participation + gdp, crime_s))$r.squared
1/(1-vif_man_edu)
```

```
## [1] 3.326385
```

A common rule of thumbs is that a VIF number greater than 10 may indicate high collinearity and worth further inspection.

Dive Deeper: I've created `divedeeper_ols` in the following code chunk. Before you complete the code to inspect the VIF, what are your suspicions? Do you expect any VIF higher than 10?

Now go ahead and calculate the VIF values for the model's predictors? Were you correct?

```
divedeeper_ols <- lm(crime_rate ~ police_exp60 + mean_education + m_per1000f + prob_prison + police_exp5
```

The limitations of the different machine learning models are something we will revisit soon and again as we progress in the machine learning specialization. Not only will we learn how to identify them early - we'll also learn various techniques to treat them, preventing overfitting / underfitting and making model diagnostic a critical part of your machine learning toolset.

Annotations