Linear Regression

Dr. Saad

Outline

1. General Concepts

Outline

Outline

Outline

1. General Concepts
2. Simple linear Regression
3. Methodology of analysis
4. lm() function

Outline

1. General Concepts
2. Simple linear Regression
3. Methodology of analysis
4. lm() function
5. Verbs from dplyr package

Outline

Outline

1. General Concepts
2. Simple linear Regression
3. Methodology of analysis
4. lm() function
5. Verbs from dplyr package
6. Fitting Simple Linear Regression: Example
7. Introduction to **broom** package

Outline

1. General Concepts
2. Simple linear Regression
3. Methodology of analysis
4. lm() function
5. Verbs from dplyr package
6. Fitting Simple Linear Regression: Example
7. Introduction to **broom** package
8. Wrap up Examples

Outline

1. General Concepts
2. Simple linear Regression
3. Methodology of analysis
4. lm() function
5. Verbs from dplyr package
6. Fitting Simple Linear Regression: Example
7. Introduction to **broom** package
8. Wrap up Examples
9. Final Project

Section 1

Theoritical Part

Necessary Tools for This Lecture

```
library(dplyr)
library(ggplot2)
library(readr)
library(broom)
```

Terminology

- **Dependent variable** (a.k.a. Response, or Target variable): is the variable we want to predict. In **linear Regression** it must be **continuous** (numeric)

Terminology

- **Dependent variable** (a.k.a. Response, or Target variable): is the variable we want to predict. In **linear Regression** it must be **continuous** (numeric)
- **Independent Variable(s)** (a.k.a **Explanatory**, **Input** variable): The variables that explain how the dependent variable will change.

Types of variable

1. **Numeric** or **Quantitative** Variable: which can be
   1. Continuous: eg. prices, income . . . etc
   2. Discrete: count variable, eg. number of wins
2. **Qualitative** or **Categorical** variable: which can be
   1. Ordinal: eg. Level of education,Regions . . . etc.

Scenarios to be in Mind when Modelling

Determining the type of model depends on the type of variables

Case 01: **Dependent is continuous**

Independent variable(s)

1. Continous:

Scenarios to be in Mind when Modelling

Determining the type of model depends on the type of variables

Case 01: **Dependent is continuous**

Independent variable(s)

1. Continous:

- Linear Regression (OLS)

Scenarios to be in Mind when Modelling

Determining the type of model depends on the type of variables

Case 01: **Dependent is continuous**

Independent variable(s)

1. Continous:
   - Linear Regression (OLS)
2. Categorical

Scenarios to be in Mind when Modelling

Determining the type of model depends on the type of variables

Case 01: **Dependent is continuous**

Independent variable(s)

1. Continous:
   - Linear Regression (OLS)
2. Categorical
   - ANOVA (Analysis of variance)

Scenarios to be in Mind when Modelling

Determining the type of model depends on the type of variables

Case 01: **Dependent is continuous**

Independent variable(s)

1. Continous:
   - Linear Regression (OLS)
2. Categorical
   - ANOVA (Analysis of variance)
3. Both (Continuous and Categorical)

Scenarios to be in Mind when Modelling

Determining the type of model depends on the type of variables

Case 01: **Dependent is continuous**

Independent variable(s)

1. Continous:
   - Linear Regression (OLS)
2. Categorical
   - ANOVA (Analysis of variance)
3. Both (Continuous and Categorical)
   - ANCOVA (Analysis of covariance)

Case 02: **Dependent is Categorical**

Independent variable(s)

1. Continous:

Case 02:**Dependent is Categorical**

Independent variable(s)

1. Continous:

- Logistic Regression or (classification)

Case 02:**Dependent is Categorical**

Independent variable(s)

1. Continous:
   - Logistic Regression or (classification)
2. Categorical

Case 02:**Dependent is Categorical**

Independent variable(s)

1. Continous:
   - Logistic Regression or (classification)
2. Categorical
   - Logistic Regression or (classification)

Case 02:**Dependent is Categorical**

Independent variable(s)

1. Continous:
   - Logistic Regression or (classification)
2. Categorical
   - Logistic Regression or (classification)
3. Both (Continuous and Categorical)

Case 02:**Dependent is Categorical**

Independent variable(s)

1. Continous:
   - Logistic Regression or (classification)
2. Categorical
   - Logistic Regression or (classification)
3. Both (Continuous and Categorical)
   - Logistic Regression or (classification)

General Formula of a simple linear regression

The general formula of a simple linear regression

$$\mathbf{Dep} = \beta_\mathbf{0} + \beta_\mathbf{1}\mathbf{indep} + \varepsilon$$

or in mathematical notations

$$\mathbf{Y} = \beta_\mathbf{0} + \beta_\mathbf{1}\mathbf{X} + \varepsilon$$

Or literally:

$$\mathbf{Y} = \mathbf{intercept} + \mathbf{Slope} * \mathbf{X} + \mathbf{noise}$$

Methodology of Analysis

The first step is called EDA (**Exploratory Data Analysis**).

1. Data Exploration: data structure, variables names, number of rows and coloumns.

Methodology of Analysis

The first step is called EDA (**Exploratory Data Analysis**).

1. Data Exploration: data structure, variables names, number of rows and coloumns.
2. Draw graphs

Methodology of Analysis

The first step is called EDA (**Exploratory Data Analysis**).

1. Data Exploration: data structure, variables names, number of rows and coloumns.
2. Draw graphs
3. Summary statistics

Methodology of Analysis

The first step is called EDA (**Exploratory Data Analysis**).

1. Data Exploration: data structure, variables names, number of rows and coloumns.
2. Draw graphs
3. Summary statistics
4. Preprocess the data if necessary: scaling, centering, normalizing, dealing with missing data . . . etc

Methodology of Analysis

The first step is called EDA (**Exploratory Data Analysis**).

1. Data Exploration: data structure, variables names, number of rows and coloumns.
2. Draw graphs
3. Summary statistics
4. Preprocess the data if necessary: scaling, centering, normalizing, dealing with missing data ...etc
5. Building the model (Fitting linear regression)

Methodology of Analysis

The first step is called EDA (**Exploratory Data Analysis**).

1. Data Exploration: data structure, variables names, number of rows and coloumns.
2. Draw graphs
3. Summary statistics
4. Preprocess the data if necessary: scaling, centering, normalizing, dealing with missing data . . . etc
5. Building the model (Fitting linear regression)
6. Report the results

Start Point of a Project

Before we start doing any analysis, we must start with a question in mind, or a problem to solve.

The second step is to collect some data to answer this question.

Based on the question(s), we will determine the response variable, and the dependent variables.

Estimation of simple linear Regression

the function used to estimate the linear regression is the `lm()` function.

as always, use the `help(lm)` if you are not familiar with the function.

```
help(lm)
```

get the arguments

```
args(lm)
```

```
## function (formula, data, subset, weights, na.action, method = "qr",
##     model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRU
##     contrasts = NULL, offset, ...)
## NULL
```

The formula function

The first argument of `lm` function is another function called formula. We must get familiar with this function

`formula()` is used to represent a mathematical equation in R, in our case the model.

To represent the equal sign $=$ in R, we will use a sign called **tilde** ~. This sign separates the **response variable** from the **explanatory variables**

for example

$$y = a + b$$

can be represented in R by using `formula()` function as.

```
formula(y~a + b)
```

```
## y ~ a + b
```

Checking the formula object

```
fml <- formula(y~a+b)
fml
```

```
## y ~ a + b
```

```
class(fml)
```

```
## [1] "formula"
```

```
typeof(fml)
```

```
## [1] "language"
```

```
mode(fml)
```

```
## [1] "call"
```

Simple Linear Regression Formula in R

As for a simple linear regression model, the formula simply can be written as

```
model_formula <- as.formula(y~x)
model_formula
```

```
## y ~ x
```

Note: The automatically includes an intercept in the model. So there is no need to add one.

General Formula

the simple linear regression formula can extend to be a multiple formula by using the plus + sign between predictors. It is done in R like this:

```
as.formula(y ~ x_1 + x_2 + ... + x_k)
```

```
## y ~ x_1 + x_2 + ... + x_k
```

The Data Argument

The second argument to be passed to `lm()` function is **data**, which is the data set that contains the variables to be estimated in the model. data can be a `data.frame`.

the general formula for a model will be

```
#model <- lm(dep ~ indep, data='dataset-name')
```

```
-- See the help for other arguments ‘help(lm)‘.
```

A visit to dplyr package

`dplyr` package is called the **The Grammar of Data manipulation**. In this short visit we are going to call few functions

1. `glimpse` function: It works just like `str` but it gives a nice printing of the data structure
2. `select` select the variables of interest from a data set.

Another important and widely used operator is the **pipe operator** %>% from `magrittr` package -imported by dplyr-. This operator simplifies the code a great deal.

Understanding The Pipe %>%

To write the pipe use the shortcut `ctrl + Shift + m`

Simply, the operator passes the last result to the first argument of the next function

Examples

```
16 %>% log() # the same as
```

```
## [1] 2.773
```

```
log(16)
```

```
## [1] 2.773
```

```
81 %>% sqrt() %>% log() # The same as
```

```
## [1] 2.197
```

```
log(sqrt(81))
```

```
## [1] 2.197
```

The pipe is used extensively in data manipulation. Here an example with `select` function

```
# data(mtcars)
# names(mtcars)
mtcars %>% select(mpg, wt) %>% head(2)
```

```
##                 mpg    wt
## Mazda RX4        21 2.620
## Mazda RX4 Wag    21 2.875
```

```
# the same as
select(mtcars, mpg, wt) %>% head(2)
```

```
##                 mpg    wt
## Mazda RX4        21 2.620
## Mazda RX4 Wag    21 2.875
```

Section 2

Practice Part

Practice Part

1. Loading a dataset

```
df <- MASS::Cars93
colnames(df)
```

```
##  [1] "Manufacturer"       "Model"                "Type"
##  [4] "Min.Price"          "Price"                "Max.Price"
##  [7] "MPG.city"           "MPG.highway"          "AirBags"
## [10] "DriveTrain"         "Cylinders"            "EngineSize"
## [13] "Horsepower"         "RPM"                  "Rev.per.mile"
## [16] "Man.trans.avail"    "Fuel.tank.capacity" "Passengers"
## [19] "Length"             "Wheelbase"            "Width"
## [22] "Turn.circle"        "Rear.seat.room"       "Luggage.room"
## [25] "Weight"             "Origin"               "Make"
```

2. Data Exploration

The most comonly used functions in data exploration are listed below

```
#str() or glimpse()
#head(df)
#tail(df)
#names() or colnames()
#nrow()
#ncol()
#summary()
```

3. Selecting the variables of interest

To make things simple, we start by fitting simple linear regression. The we will tackle more complicated models.

We will use the `select` function to choose only two variables. The reason we use this function for its simplicity.

```
my_df <- df %>% select(Weight, MPG.highway)
my_df %>% head(3)
```

```
##   Weight MPG.highway
## 1   2705          31
## 2   3560          25
## 3   3375          26
```

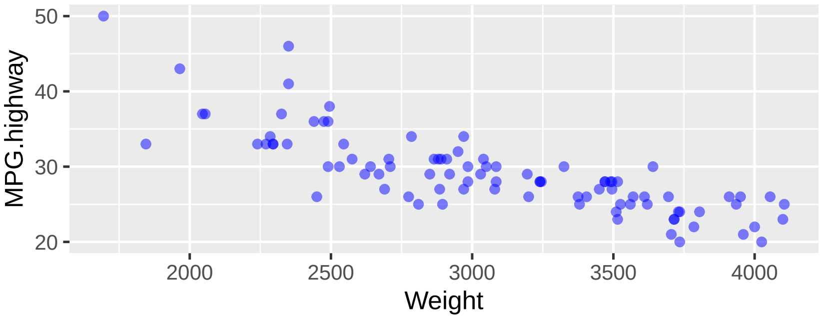
4. Summary Statistics

```
summary(my_df)
```

```
##       Weight        MPG.highway
##  Min.   :1695   Min.   :20.0
##  1st Qu.:2620   1st Qu.:26.0
##  Median :3040   Median :28.0
##  Mean   :3073   Mean   :29.1
##  3rd Qu.:3525   3rd Qu.:31.0
##  Max.   :4105   Max.   :50.0
```

5. Plotting

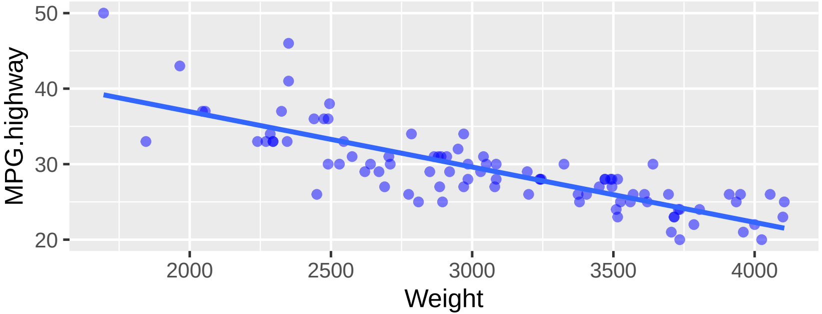
```
my_df %>% ggplot(aes(Weight, MPG.highway)) +
  geom_point(color = "blue", alpha = 0.5)
```



Adding a fitted line to the plot

```r
my_df %>% ggplot(aes(Weight, MPG.highway)) +
  geom_point(color = "blue", alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Fit Simple Linear Regression

```r
model <- lm(MPG.highway~Weight , data = my_df)
model
```

```
##
## Call:
## lm(formula = MPG.highway ~ Weight, data = my_df)
##
## Coefficients:
## (Intercept)       Weight
##    51.60137     -0.00733
```

Ckecking `lm()` Objects

It is extremely important to know how objects are stored in R. This will help you a great deal when you deal with complex models.

```
class(model)
```

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

```
typeof(model)
```

```
## [1] "list"
```

```
length(model)
```

```
## [1] 12
```

```r
names(model)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
# Use str() for more details about lm object
# str(model)
```

Extracting Information From lm object

- `summary()` The very first function would anyone know about, It gives the summary results of the fitted model.
- `coefficients()` Reports the estimated parameters
- `residuals()` gives the residuals or **errors**
- `fitted()` provides the fitted values **y_hat**
- `predict()` used for predictions
- `plot()` For diagnostic plots

Other functions

- `confint()` for confidence interval
- `anova()` for analysis of variance or comparing models.
- `vcov()` for variance covariance matrix.
- `AIC()` For Akaike's Information Criterion.

Summary Function

```
summary(model)
```

```
##
## Call:
## lm(formula = MPG.highway ~ Weight, data = my_df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.650 -1.836 -0.077  1.824 11.617
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 51.601365   1.735550    29.7   <2e-16 ***
## Weight      -0.007327   0.000555   -13.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.14 on 91 degrees of freedom
## Multiple R-squared:  0.657,  Adjusted R-squared:  0.653
## F-statistic:  174 on 1 and 91 DF,  p-value: <2e-16
```

Coefficients Function

```
coefficients(model)
```

```
## (Intercept)      Weight
##   51.601365   -0.007327
```

Fitted function

```
head(fitted(model), 5)
```

```
##     1     2     3     4     5
## 31.78 25.52 26.87 26.65 24.93
```

Residuals Function

```
head(residuals(model), 5)
```

```
##       1       2       3       4       5
## -0.7817 -0.5170 -0.8725 -0.6527  5.0691
```

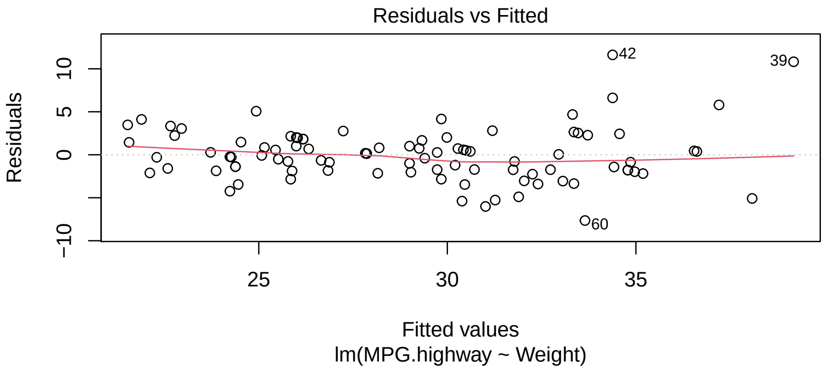
anova function

```
anova(model)
```

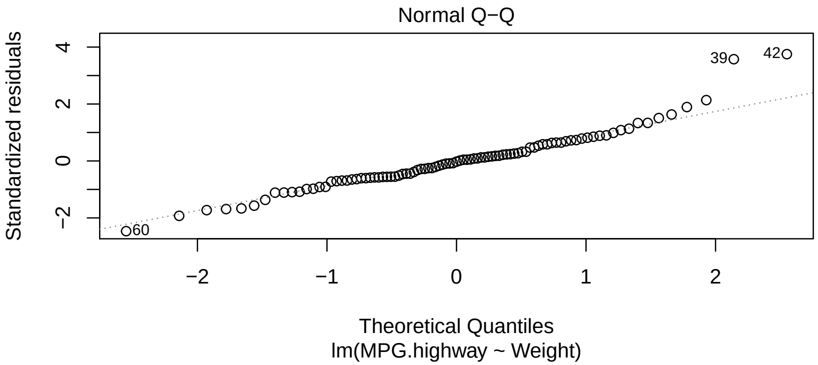
```
## Analysis of Variance Table
##
## Response: MPG.highway
##           Df Sum Sq Mean Sq F value Pr(>F)
## Weight     1   1719    1719     174 <2e-16 ***
## Residuals 91    897      10
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Diagnostic plots

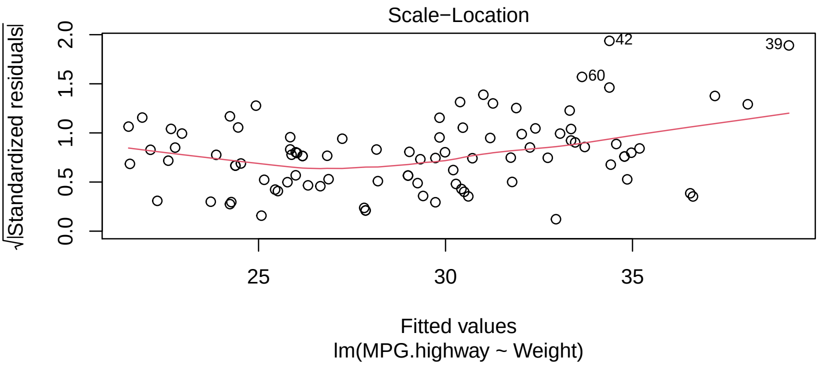
```r
plot(model, which = 1)
```



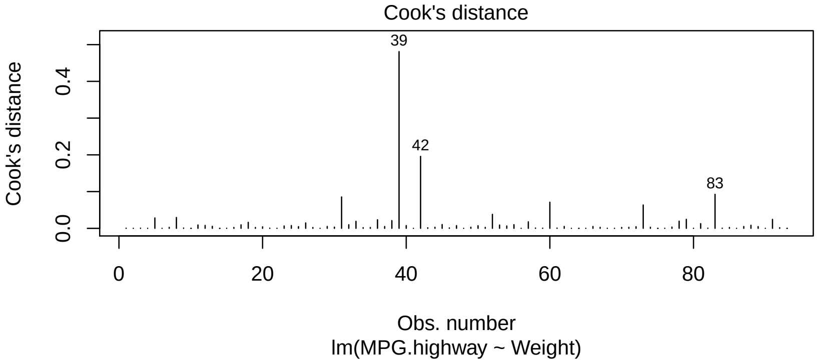
```
plot(model, which = 2)
```



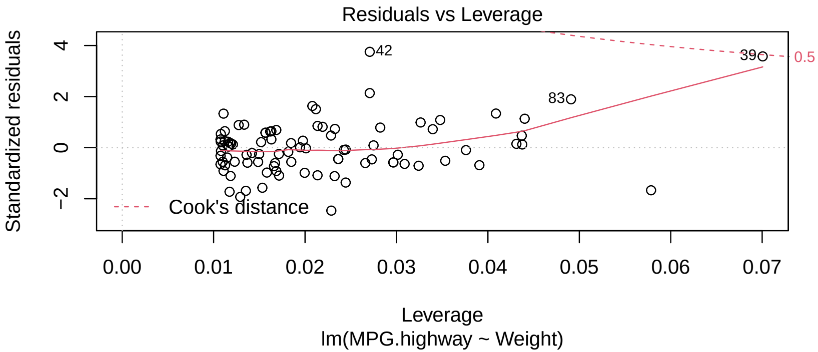
```
plot(model, which = 3)
```



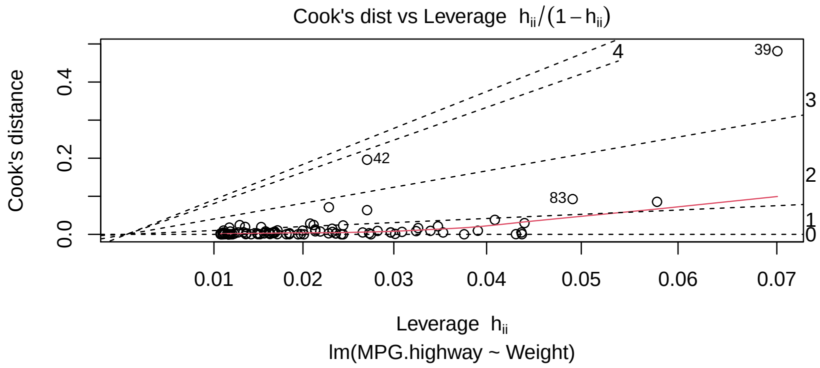
```
plot(model, which = 4)
```



```
plot(model, which = 5)
```



```r
plot(model, which = 6)
```



predict function

```r
head(predict(model, my_df), 10)
```

```
##     1     2     3     4     5     6     7     8     9    10
## 31.78 25.52 26.87 26.65 24.93 30.50 26.18 21.52 25.99 25.08
```

Note:

predict function Will be discussed later in the course in a great detail

Section 3

Introduction to **broom** Package

Overview of Broom functions

We have seen that the summary function returns lots of information, which is designed to be read not to be manipulated with code. However, we certainly need that information to be used in our code, such as plots. Only functions that return data in some data type like vectors or data frames can be used inside the `R` code.

Broom Package provides three convenient function: `tidy()`, `augment()` and `glance()`. These functions return information in form of `data.frame` which makes it easier to include in `tidyverse` package function. **We will focus on this package along our course**

Each of these function will be discussed individually in the next slides.

Tidy() function

Generally, `tidy()` function returns the estimated coefficients and their details in a data.frame. Consider that `tidy()` deals with **The first part of summary() function output**

```
head(tidy(model))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) 51.6      1.74          29.7 1.20e-48
## 2 Weight      -0.00733  0.000555     -13.2 7.18e-23
```

Augment() Function

This function deals with observation specifications that are used in the model estimation and more useful information.

```
names(augment(model))
```

```
## [1] "MPG.highway" "Weight"      ".fitted"     ".resid"      ".hat"
## [6] ".sigma"      ".cooksd"     ".std.resid"
```

```
head(augment(model), 3)
```

```
## # A tibble: 3 x 8
##   MPG.highway Weight .fitted .resid    .hat .sigma   .cooksd .std.resi
##         <int>  <int>   <dbl>  <dbl>   <dbl>  <dbl>     <dbl>      <dbl
## 1          31   2705    31.8 -0.782 0.0150    3.16 0.000479      -0.25
## 2          25   3560    25.5 -0.517 0.0182    3.16 0.000256      -0.16
## 3          26   3375    26.9 -0.873 0.0136    3.15 0.000540      -0.28
```

Glance() Function

It returns model-level results, the model specifications. You can think of it as it returns the last part of `summary` function output.

```
names(glance(model))
```

```
##  [1] "r.squared"     "adj.r.squared" "sigma"         "statistic"
##  [5] "p.value"       "df"            "logLik"        "AIC"
##  [9] "BIC"           "deviance"      "df.residual"   "nobs"
```

```
head(glance(model))
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AI
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl
## 1     0.657         0.653  3.14      174. 7.18e-23     1  -237.  481
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs
```