Programming for Data Science with R

Part - IV

Mohammad Wasiq

13/2/22

DSM - 1005

Programming for Data Science with R - IV
DSM-1005

Table of Contents

1 R programing fo Data Science DSM-1005 1
2 Infer for Regression 2
2.1 Conditions for Inference for Regression 4
2.1.1 Residuals Refreshers 4
3 Conclusion of Book 15
3.1 Explaratoy Data Analysis : Part - I 15
3.2 Explaratoy Data Analysis : Part - II 25

We follow the book named **Statistical Inference via Data Science** *A ModernDive into R and the Tidyvers* by **Chester Ismay** and **Albert Y. Kim**

Teacher : **Prof. Athar Ali Khan Sir**

Writer : **Mohammad Wasiq** , *MS(Data Science)*

1 R programing fo Data Science DSM-1005

In this Script we learn the R programming for Data Science at intermediate level . We learn the following Topics

1. **Tidyverse**
- **Data Visualization** Using **ggplot2**
- **Data Wrangling** Using **dplyr**
- **Data Importing** & **Tidy Data**

2. **Data Modelling** with **moderndive**
- **Simple Regression**
- **Multiple Regression**

3. **Statistical Inference** with **infer**
- **Sampling**
- **Confidence Intervals**
- **Hypothesis Testing**

- **Inference for Regression**

4. **Conclusion**

- **Exploratory Data Analysis**

- **Statistics**

Note Here we are Discussing Only **Inference for Regression** from *Unit- 3* and **EDA** from *Unit-4* . All the above *Topics / Units* , we have already discused in $Part - I^{st}, II^{nd}, III^{rd}$ respctively .

2 Infer for Regression

Note: This is the additional of **Part - II**

Task : Load the require packages go this chapter .

```
library(tidyverse)
library(moderndive)
library(infer)
```

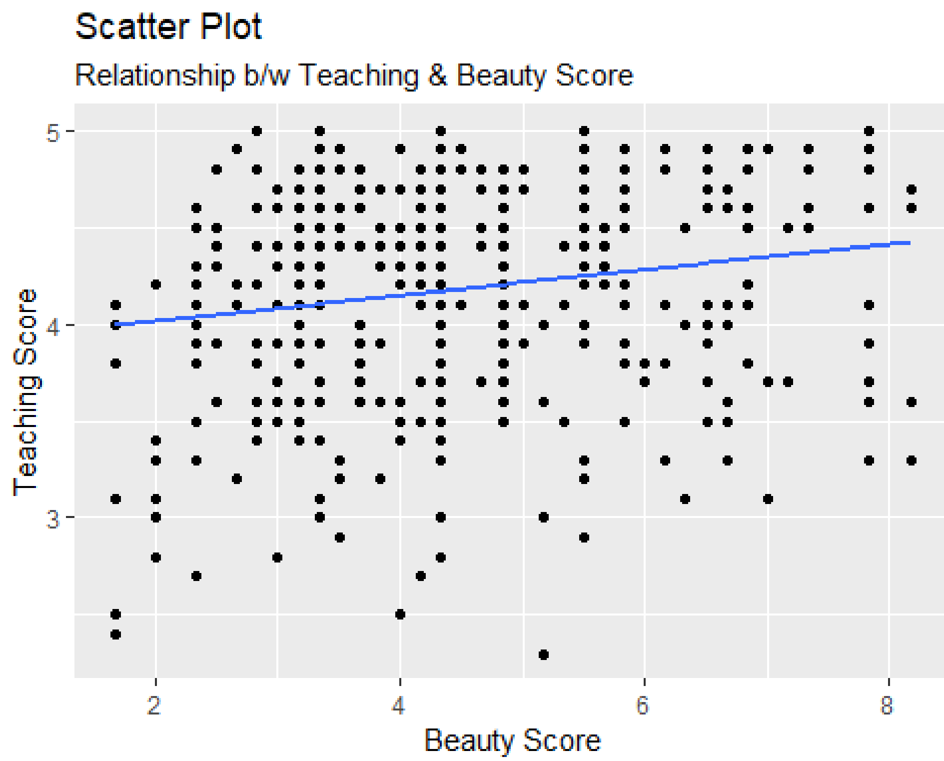
Task : Select the variable *ID , score , bty_avg , age* fom *evals* data .

```
evals_ch5 <- evals %>%
  select(ID , score , bty_avg , age)

evals_ch5 %>% glimpse()

## Rows: 463
## Columns: 4
## $ ID      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,~
## $ score   <dbl> 4.7, 4.1, 3.9, 4.8, 4.6, 4.3, 2.8, 4.1, 3.4, 4.5, 3.8,
4.5, 4.6, 3.9, 3.9, 4.3, 4.5, 4.8, 4.6, 4.6, 4.~
## $ bty_avg <dbl> 5.000, 5.000, 5.000, 5.000, 3.000, 3.000, 3.000, 3.333,
3.333, 3.167, 3.167, 3.167, 3.167, 3.167, 3.16~
## $ age     <int> 36, 36, 36, 36, 59, 59, 59, 51, 51, 40, 40, 40, 40, 40,
40, 40, 40, 31, 31, 31, 31, 31, 31, 62, 62, 62~
```

Task : Make a Scatter Plot between bty_avg , score .

```
ggplot(evals_ch5 , aes(bty_avg , score)) +
  geom_point() +
  geom_smooth(method = "lm" , se = F) +
  labs(
    x = "Beauty Score" ,
    y = "Teaching Score" ,
    title = "Scatter Plot" ,
    subtitle = "Relationship b/w Teaching & Beauty Score"
    )
```

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



Task : Fit the Linear Model

Model : $Score = \beta_0 + \beta_{bty\_avg}(bty\_avg) + \epsilon$

```
# Fit Regression Model :
score_model <- lm(score ~ bty_avg , data = evals_ch5)

# Regression Model
get_regression_table(score_model)

## # A tibble: 2 x 7
##   term      estimate std_error statistic p_value lower_ci upper_ci
##   <chr>        <dbl>     <dbl>     <dbl>   <dbl>    <dbl>    <dbl>
## 1 intercept    3.88      0.076     51.0        0    3.73     4.03
## 2 bty_avg      0.067     0.016      4.09       0    0.035    0.099
```

Fitted Model : $\widehat{Score} = 3.88 + 0.067(bty\_avg)$
Interpretation :
Estimate : If *Beauty Average* is increase by *1* unit than *Score* increases by *0.067* unit .
Standard Error : We can expect about 0.016 units of variation in the bty_avg slope variable
Our Hypotheses : $H_0: \beta_1 = 0 \quad vs \quad H_A: \beta_1 \neq 0$.
P-Value : The *p-value* in this case is 0, for any choice of significance level α we would reject H_0 in favor of H_A . **OR** In other word , we can say that *we reject the hypothesis that there is no relationsip b/w teaching and beauty score*.

Confidence Interval : The resulting *95%* confidence interval for β_1 of **(0.035, 0.099)** can be thought of as a range of possible values for the population slope β_1 of the linear relationship between teaching and "beauty" scores.

2.1 Conditions for Inference for Regression

The first four letters of these conditions are **LINE** .
1. **L**inearity of relationship between variables
2. **I**ndependence of the residuals
3. **N**ormality of the residuals
4. **E**quality of variance of the residuals

Conditions **L, N** , and **E** can be verified through what is known as a residual analysis. Condition I can only be verified through an understanding of how the data was collected .

2.1.1 Residuals Refreshers

The *observed value* minus the *fitted value* denoted by $(y - \hat{y})$.

```
# Fit regression model:
score_model <- lm(score ~ bty_avg, data = evals_ch5)

# Get regression points:
regression_points <- get_regression_points(score_model)

regression_points %>% head()

## # A tibble: 6 x 5
##      ID score bty_avg score_hat residual
##   <int> <dbl>   <dbl>     <dbl>    <dbl>
## 1     1   4.7       5      4.21    0.486
## 2     2   4.1       5      4.21   -0.114
## 3     3   3.9       5      4.21   -0.314
## 4     4   4.8       5      4.21    0.586
## 5     5   4.6       3      4.08    0.52
## 6     6   4.3       3      4.08    0.22
```

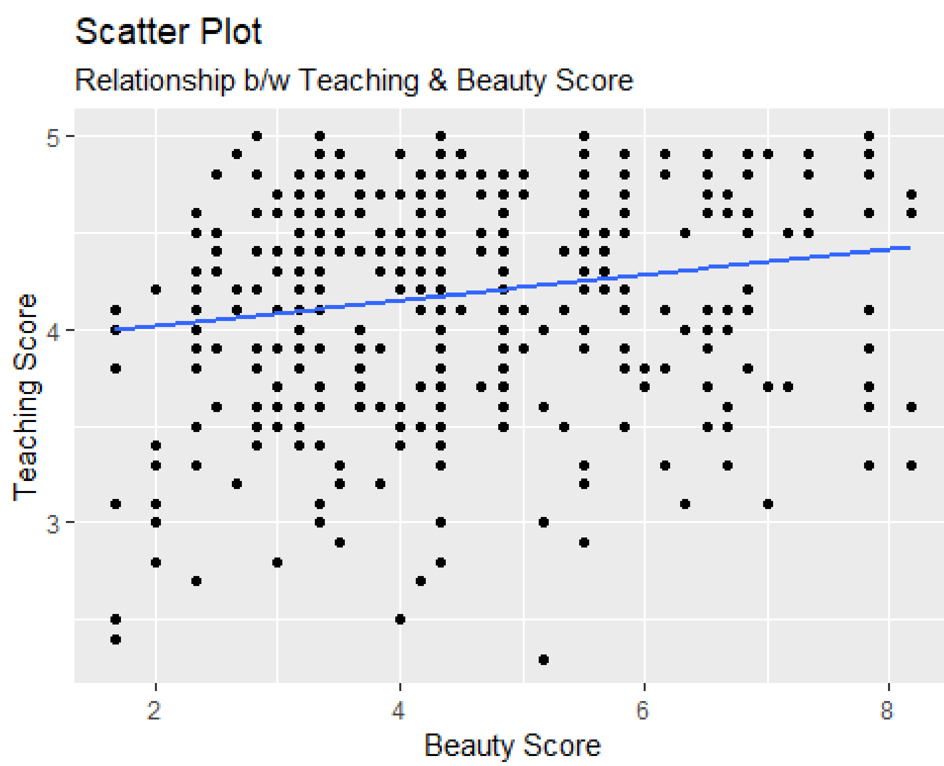
A residual analysis is used to verify conditions **L, N, E** and can be performed using appropriate data visualizations .

2.1.1.1 Linearity of Relationship

The first condition is that the relationship b/w the outcome variable y and the explanatory variable x must be **Linear** .

```
ggplot(evals_ch5 , aes(bty_avg , score)) +
  geom_point() +
  geom_smooth(method = "lm" , se = F) +
  labs(
    x = "Beauty Score" ,
    y = "Teaching Score" ,
```

```
    title = "Scatter Plot" ,
    subtitle = "Relationship b/w Teaching & Beauty Score"
    )

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



In the figure we can easily see that there is a non-linear relationship .

```
get_correlation(evals_ch5 , formula = score ~ bty_avg)

## # A tibble: 1 x 1
##     cor
##   <dbl>
## 1 0.187
```

There is only **0.187** correlation .

2.1.1.2 Independence of Residuals

The second condition is that the residuals must be **I**ndependent. In other words, the different observations in our data must be independent of one another.

```
evals %>%
  select(ID , prof_ID , score , bty_avg) %>%
  head()

## # A tibble: 6 x 4
##      ID prof_ID score bty_avg
```

```
##    <int>    <int> <dbl>    <dbl>
## 1      1        1   4.7        5
## 2      2        1   4.1        5
## 3      3        1   3.9        5
## 4      4        1   4.8        5
## 5      5        2   4.6        3
## 6      6        2   4.3        3
```

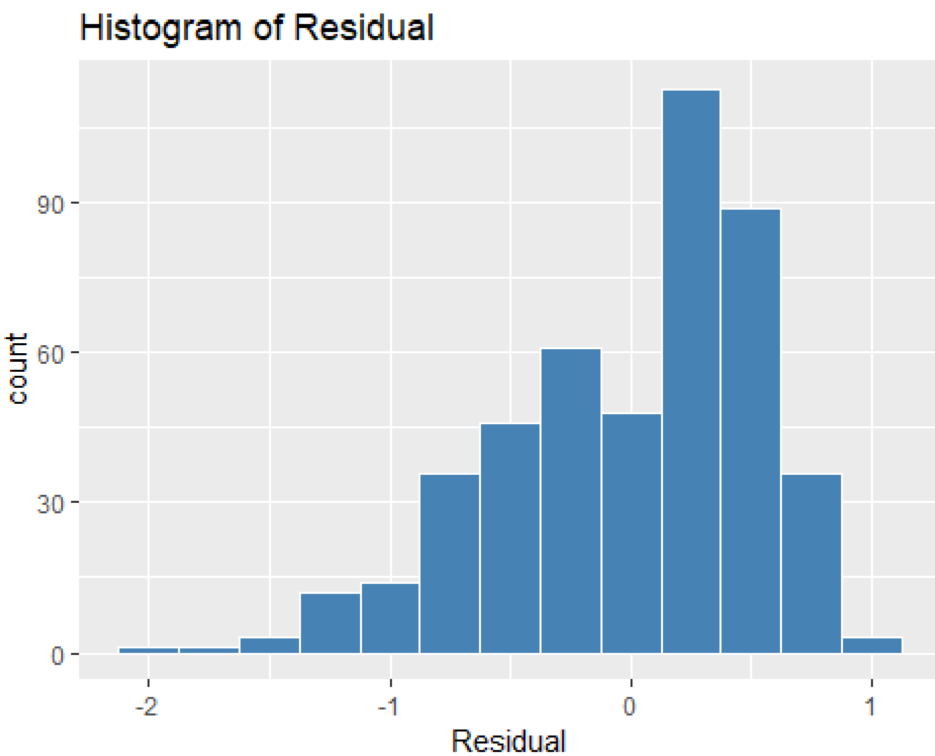
The professor with prof_ID equal to 1 taught the first 4 courses in the data, the professor with prof_ID equal to 2 taught the next 3, and so on. Given that the same professor taught these first four courses, it is reasonable to expect that these four teaching scores are related to each other. If a professor gets a high score in one class, chances are fairly good they'll get a high score in another. The first four courses taught by professor 1 are dependent, the next 3 courses taught by professor 2 are related, and so on. Any proper analysis of this data needs to take into account that we have repeated measures for the same profs .

2.1.1.3 Normality of Residuals

The third condition is that the residuals should follow a **N**ormal distribution. The center of this distribution should be 0. In other words, sometimes the regression model will make positive errors: $(y - \hat{y}) > 0$. Other times, the regression model will make equally negative errors: $(y - \hat{y}) < 0$

The simplest way to check the normality of the residuals is to look at a **Histogram** .

```
ggplot(regression_points, aes(x = residual)) +
  geom_histogram(binwidth = 0.25 , col = "white" , fill = "steelblue") +
  labs(x = "Residual" ,
    title = "Histogram of Residual")
```

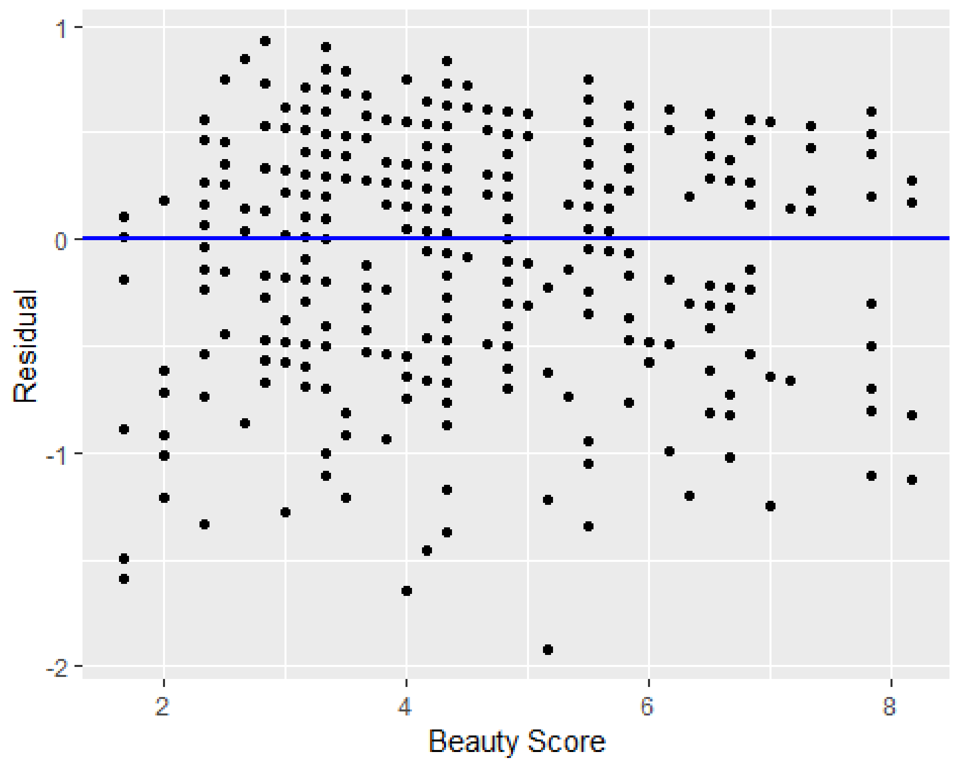


This *Histogram* shows that we have more positive residuals than negative. Since the residual $(y - \hat{y})$ is positive when $y > \hat{y}$, it seems our regression model's fitted teaching scores $\hat{y}$ tend to underestimate the true teaching scores y. Furthermore, this *histogram* has a slight left-skew in that there is a tail on the left. This is another way to say the residuals exhibit a negative skew.

2.1.1.4 Equality of Variance

The fourth and final condition is that the residuals should exhibit **E**qual variance across all values of the explanatory variable x. In other words, the value and spread of the residuals should not depend on the value of the explanatory variable x .

```
ggplot(regression_points, aes(x = bty_avg, y = residual)) +
  geom_point() +
  labs(x = "Beauty Score", y = "Residual") +
  geom_hline(yintercept = 0, col = "blue", size = 1)
```

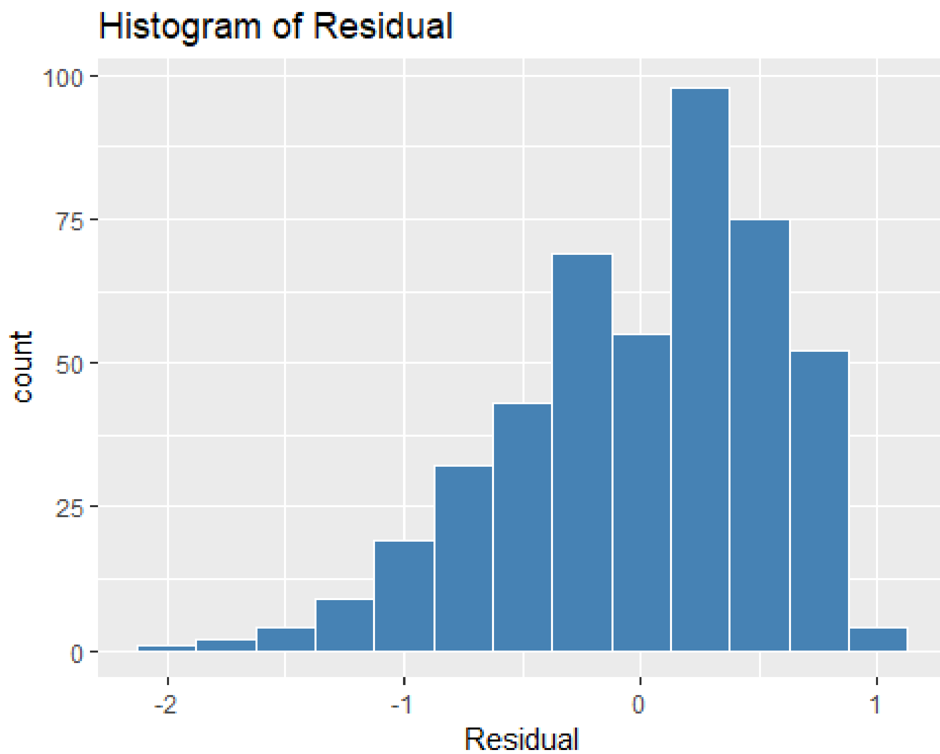


How the spread of the residuals increases as the value of x increases. This is a situation known as heteroskedasticity .

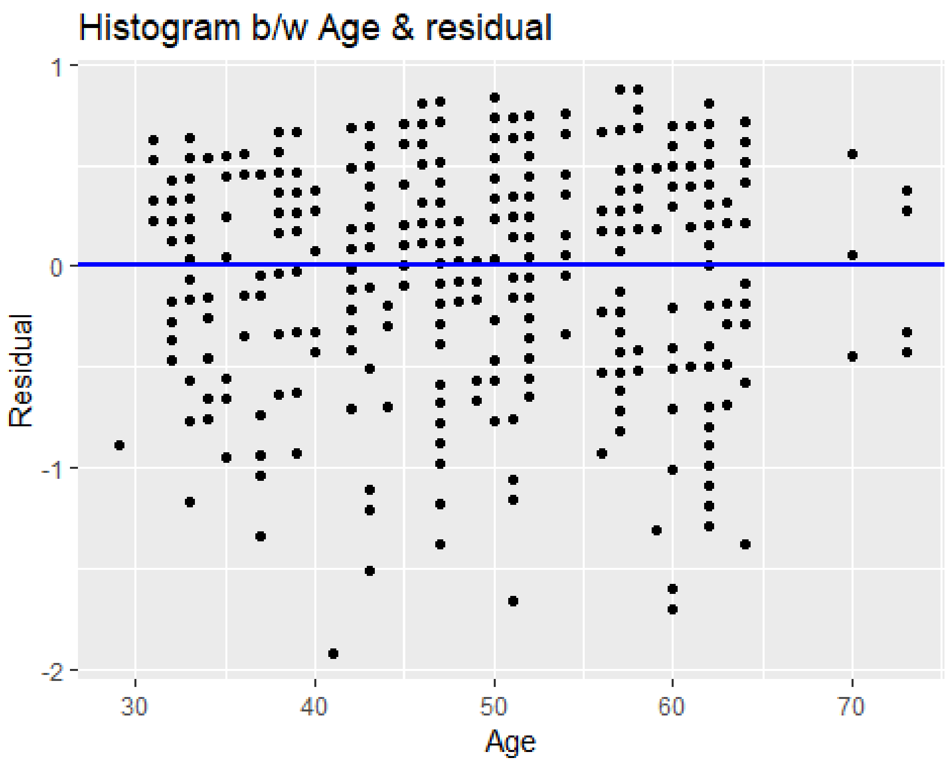
(LC 10.1) Continuing with our regression using age as the explanatory variable and teaching score as the outcome variable. Use the get_regression_points() function to get the observed values, fitted values, and residuals for all 463 instructors.

```
evals_ch5 <- evals %>%
  select(ID, score, bty_avg, age)

# Fit regression model:
score_age_model <- lm(score ~ age, data = evals_ch5)

# Get regression points:
regression_points <- get_regression_points(score_age_model)

regression_points %>% head()

## # A tibble: 6 x 5
##       ID score   age score_hat residual
##    <int> <dbl> <int>     <dbl>    <dbl>
## 1      1   4.7    36      4.25    0.452
## 2      2   4.1    36      4.25   -0.148
## 3      3   3.9    36      4.25   -0.348
## 4      4   4.8    36      4.25    0.552
## 5      5   4.6    59      4.11    0.488
## 6      6   4.3    59      4.11    0.188
```

```r
ggplot(regression_points, aes(x = residual)) +
  geom_histogram(binwidth = 0.25 , col = "white" , fill = "steelblue") +
  labs(x = "Residual" ,
    title = "Histogram of Residual")
```



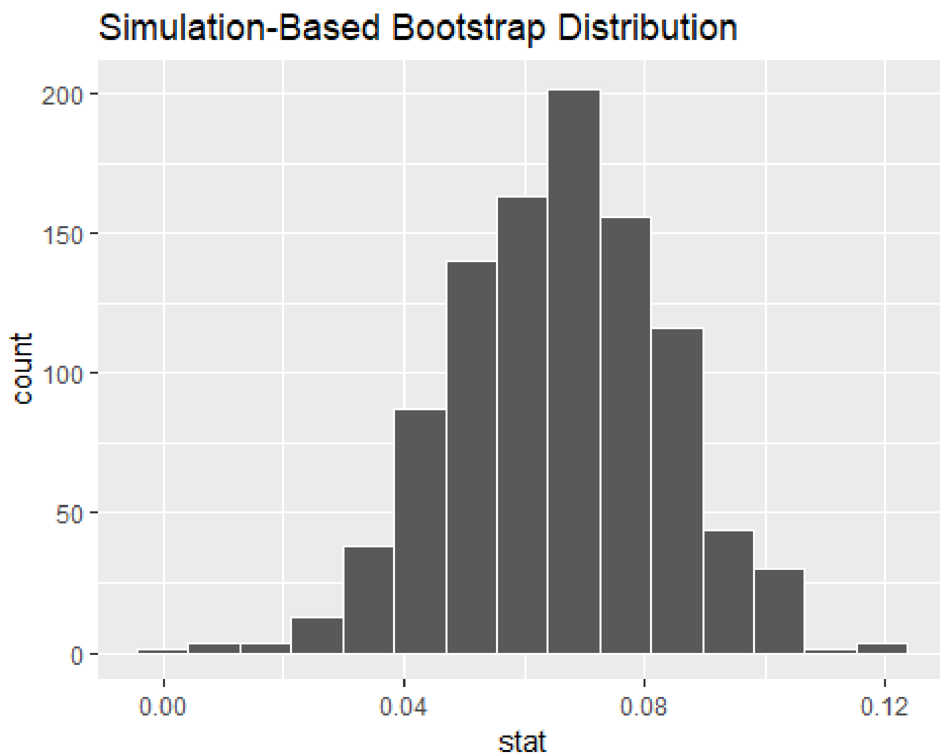

```r
ggplot(regression_points, aes(x = age, y = residual)) +
  geom_point() +
  labs(x = "Age", y = "Residual" , title = "Histogram b/w Age & residual") +
  geom_hline(yintercept = 0, col = "blue", size = 1)
```



2.1.1.5 Simulation Based Inference for Regression

Confidence Interval for Slope

```
bootstrap_distn_slope <- evals_ch5 %>%
  specify(formula = score ~ bty_avg) %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "slope")

bootstrap_distn_slope %>% glimpse()

## Rows: 1,000
## Columns: 2
## $ replicate <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 2~
## $ stat      <dbl> 0.04005456, 0.06209225, 0.07607721, 0.06215655,
0.04306178, 0.05387582, 0.06472703, 0.08562614, 0.06~

# Visualize
visualize(bootstrap_distn_slope)
```



C.I. Percentile Method

```
percentile_ci <- bootstrap_distn_slope %>%
  get_confidence_interval(type = "percentile", level = 0.95)

percentile_ci

## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1   0.0320   0.0992
```

The resulting percentile-based 95% confidence interval for β_1 of *(0.032, 0.099)* is similar to the confidence interval in the regression .

CI Standard Error Method :

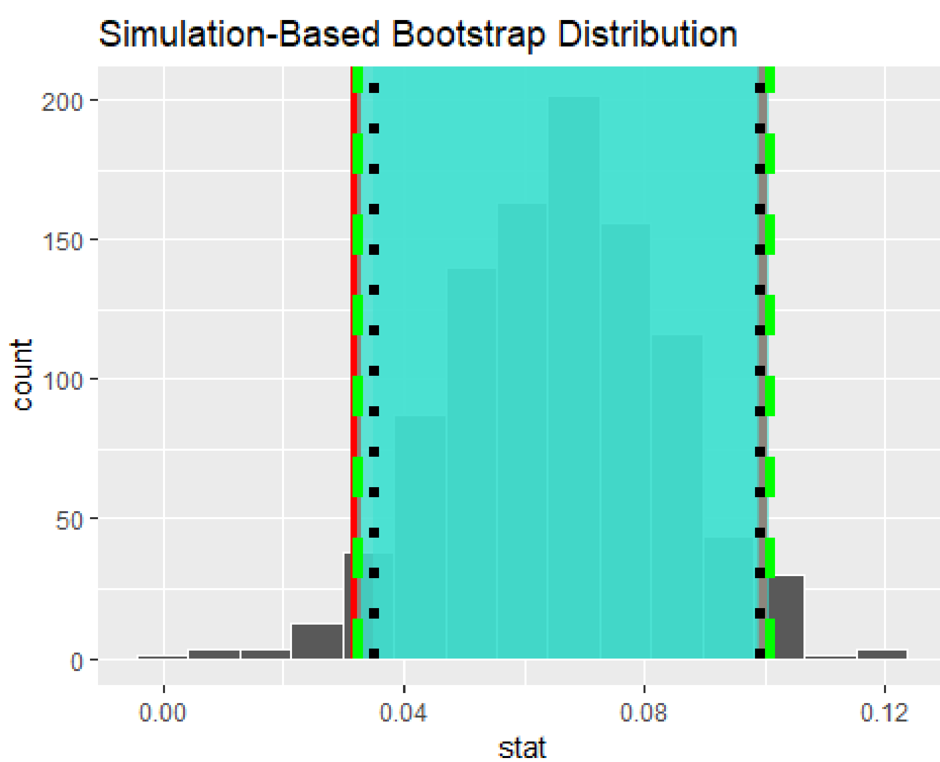
```
observed_slope <- evals %>%
  specify(score ~ bty_avg) %>%
  calculate(stat = "slope")

observed_slope

## Response: score (numeric)
## Explanatory: bty_avg (numeric)
## # A tibble: 1 x 1
##     stat
```

```
##    <dbl>
## 1 0.0666

# CI by SE Method
se_ci <- bootstrap_distn_slope %>%
  get_ci(level = 0.95, type = "se", point_estimate = observed_slope)

se_ci

## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1   0.0324    0.101
```

The resulting standard error-based 95% confidence interval for β_1 of *(0.033, 0.1)* is slightly different than the confidence interval in the regression

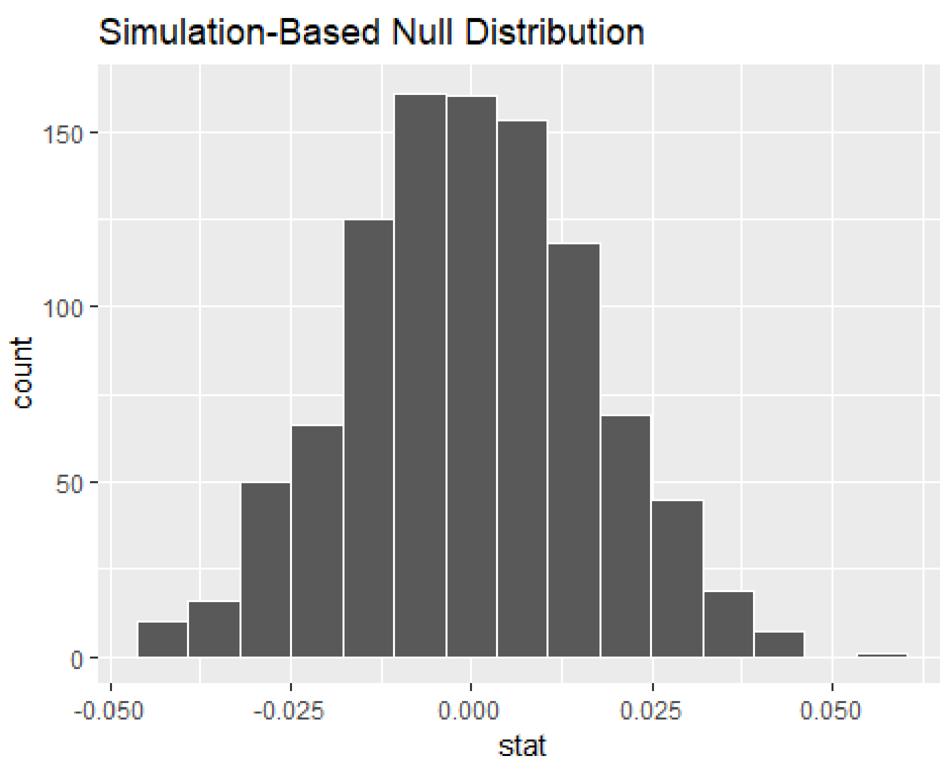
Comparing all Three

```
visualize(bootstrap_distn_slope) +
  shade_confidence_interval(endpoints = percentile_ci, fill = NULL, linetype
= "solid", color = "red") +
  shade_confidence_interval(endpoints = se_ci, fill = NULL,
linetype = "dashed", color = "green") +
  shade_confidence_interval(endpoints = c(0.035, 0.099), fill = NULL,
linetype = "dotted", color = "black")
```



Hypothesis Testing fo Slope $H_0: \beta_1 = 0 \quad vs \quad H_A: \beta_1 \neq 0$ We construct the null distribution of the fitted slope β_1 by performing the steps that follow 1. `specify()` the variables of interest in evals_ch5 with the formula: score ~ bty_avg. 2. `hypothesize()` the null hypothesis of independence. 3. `generate()` replicates by permuting/shuffling values from the original sample of 463 courses. We generate reps = 1000 replicates using type = "permute" here. 4. `calculate()` the test statistic of interest: the fitted slope β_1

```
# Then we calculate the "slope" coefficient for each of these 1000 generated
samples .
null_distn_slope <- evals %>%
  specify(score ~ bty_avg) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 1000 , type = "permute") %>%
  calculate(stat = "slope")

visualize(null_distn_slope)
```




```
# P-Vlaue
null_distn_slope %>%
  get_p_value(obs_stat = observed_slope , direction = "both")

## Warning: Please be cautious in reporting a p-value of 0. This result is an
approximation based on the number of `reps`
## chosen in the `generate()` step. See `?get_p_value()` for more
information.
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1       0
```

P-Value is **0** , So we reject H_0 thus have evidence that suggests there is a significant relationship between teaching and “beauty” scores for all instructors at UT Austin.

(LC 10.3) : Repeat the inference but this time for the correlation coefficient instead of the slope. Note the implementation of stat = “correlation” in the calculate() function of the infer package.

```
bootstrap_distn_slope <- evals_ch5 %>%
  specify(formula = score ~ bty_avg) %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "correlation")

bootstrap_distn_slope %>% head(5)  # dim = 1000 x 2

## Response: score (numeric)
## Explanatory: bty_avg (numeric)
## # A tibble: 5 x 2
##   replicate  stat
##       <int> <dbl>
## 1         1 0.195
## 2         2 0.193
## 3         3 0.106
## 4         4 0.200
## 5         5 0.189

set.seed(76)
bootstrap_distn_slope <- evals %>%
  specify(score ~ bty_avg) %>%
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "slope")
bootstrap_distn_slope %>% head(5)  # dim = 1000 x 2

## Response: score (numeric)
## Explanatory: bty_avg (numeric)
## # A tibble: 5 x 2
##   replicate   stat
##       <int>  <dbl>
## 1         1 0.0651
## 2         2 0.0382
## 3         3 0.108
## 4         4 0.0667
## 5         5 0.0716

observed_slope <- evals %>%
  specify(score ~ bty_avg) %>%
  calculate(stat = "correlation")
```

```
observed_slope

## Response: score (numeric)
## Explanatory: bty_avg (numeric)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1 0.187
```

3 Conclusion of Book

In this Part , we will discuss **Seattle House Prices**

About the Data : "House Sales in King County, USA". It consists of sale prices of homes sold between May 2014 and May 2015 in King County, Washington, USA, which includes the greater Seattle metropolitan area. This dataset is in the house_prices data frame included in the moderndive package. The dataset consists of 21,613 houses and 21 variables describing these houses (for a full list and description of these variables, see the help file by running ? house_prices in the console). In this case study, we'll create a multiple regression model where: The outcome variable y is the sale price of houses. Two explanatory variables: i. A numerical explanatory variable x_1 : house size sqft_living as measured in square feet of living space. Note that 1 square foot is about 0.09 square meters. ii.. A categorical explanatory variable x_2 : house condition, a categorical variable with five levels where 1 indicates "poor" and 5 indicates "excellent."

Load the Require Libraries :

```
library(tidyverse)
library(moderndive)
library(skimr)
library(fivethirtyeight)
```

3.1 Explaratoy Data Analysis : Part - I

Univariate Data In this part we follow the following steps : 1. Looking at the raw values . 2. Computing summary statistics . 3. Creating data visualization .

```
# Load the Data
data("house_prices")
glimpse(house_prices)

## Rows: 21,613
## Columns: 21
## $ id            <chr> "7129300520", "6414100192", "5631500400",
"2487200875", "1954400510", "7237550310", "1321400060"~
## $ date          <date> 2014-10-13, 2014-12-09, 2015-02-25, 2014-12-09,
2015-02-18, 2014-05-12, 2014-06-27, 2015-01-15,~
## $ price         <dbl> 221900, 538000, 180000, 604000, 510000, 1225000,
```

```
257500, 291850, 229500, 323000, 662500, 468000,~
## $ bedrooms      <int> 3, 3, 2, 4, 3, 4, 3, 3, 3, 3, 3, 2, 3, 3, 5, 4, 3,
4, 2, 3, 4, 3, 5, 2, 3, 3, 3, 3, 3, 4, 3, 2, ~
## $ bathrooms     <dbl> 1.00, 2.25, 1.00, 3.00, 2.00, 4.50, 2.25, 1.50,
1.00, 2.50, 2.50, 1.00, 1.00, 1.75, 2.00, 3.00, ~
## $ sqft_living   <int> 1180, 2570, 770, 1960, 1680, 5420, 1715, 1060, 1780,
1890, 3560, 1160, 1430, 1370, 1810, 2950, 1~
## $ sqft_lot      <int> 5650, 7242, 10000, 5000, 8080, 101930, 6819, 9711,
7470, 6560, 9796, 6000, 19901, 9680, 4850, 50~
## $ floors        <dbl> 1.0, 2.0, 1.0, 1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 2.0,
1.0, 1.0, 1.5, 1.0, 1.5, 2.0, 2.0, 1.5, 1.0, 1~
## $ waterfront    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,
FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE~
## $ view          <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0,
0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ condition     <fct> 3, 3, 3, 5, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3,
4, 4, 4, 4, 3, 3, 3, 4, 5, 3, 5, 3, 3, 3, 3, ~
## $ grade         <fct> 7, 7, 6, 7, 8, 11, 7, 7, 7, 7, 8, 7, 7, 7, 7, 9, 7,
7, 7, 7, 7, 9, 8, 7, 8, 6, 8, 8, 7, 8, 8, 7,~
## $ sqft_above    <int> 1180, 2170, 770, 1050, 1680, 3890, 1715, 1060, 1050,
1890, 1860, 860, 1430, 1370, 1810, 1980, 18~
## $ sqft_basement <int> 0, 400, 0, 910, 0, 1530, 0, 0, 730, 0, 1700, 300, 0,
0, 0, 970, 0, 0, 0, 0, 760, 720, 0, 0, 0, 0~
## $ yr_built      <int> 1955, 1951, 1933, 1965, 1987, 2001, 1995, 1963,
1960, 2003, 1965, 1942, 1927, 1977, 1900, 1979, ~
## $ yr_renovated  <int> 0, 1991, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ zipcode       <fct> 98178, 98125, 98028, 98136, 98074, 98053, 98003,
98198, 98146, 98038, 98007, 98115, 98028, 98074~
## $ lat           <dbl> 47.5112, 47.7210, 47.7379, 47.5208, 47.6168,
47.6561, 47.3097, 47.4095, 47.5123, 47.3684, 47.600~
## $ long          <dbl> -122.257, -122.319, -122.233, -122.393, -122.045, -
122.005, -122.327, -122.315, -122.337, -122.0~
## $ sqft_living15 <int> 1340, 1690, 2720, 1360, 1800, 4760, 2238, 1650,
1780, 2390, 2210, 1330, 1780, 1370, 1360, 2140, ~
## $ sqft_lot15    <int> 5650, 7639, 8062, 5000, 7503, 101930, 6819, 9711,
8113, 7570, 8925, 6000, 12697, 10208, 4850, 40~

# Summary of data
house_prices %>%
select(price, sqft_living, condition) %>%
skim()
```

Data summary

Name	Piped data
Number of rows	21613
Number of columns	3
_______________________	

Column type frequency:	
factor	1
numeric	2
________________________	
Group variables	None

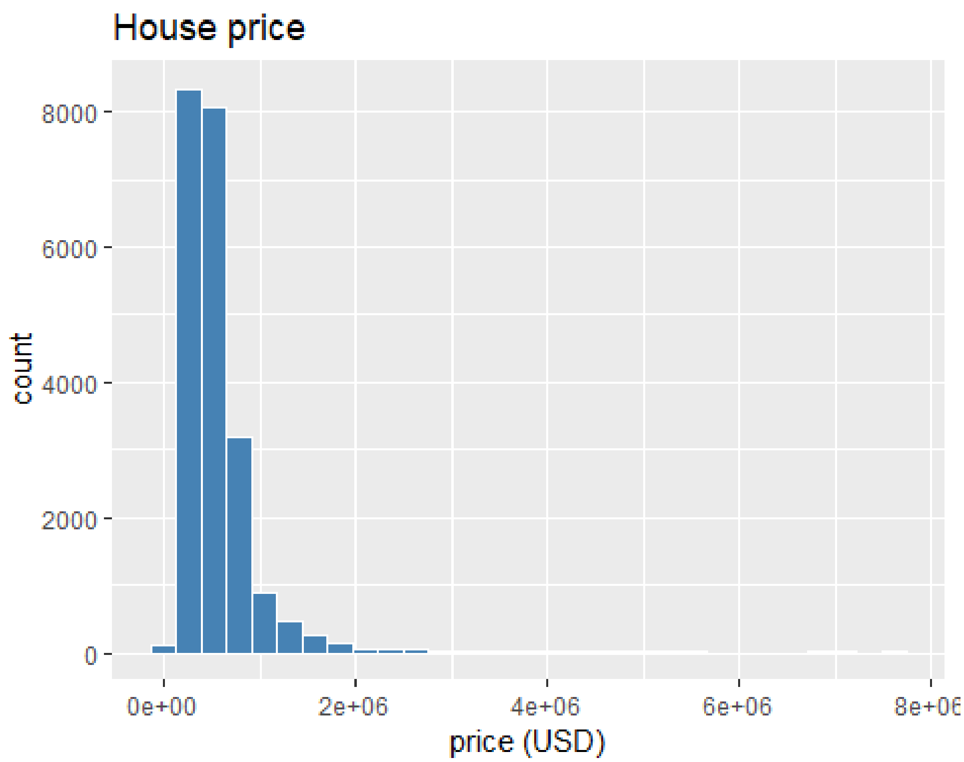
Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
condition	0	1	FALSE	5	3: 14031, 4: 5679, 5: 1701, 2: 172

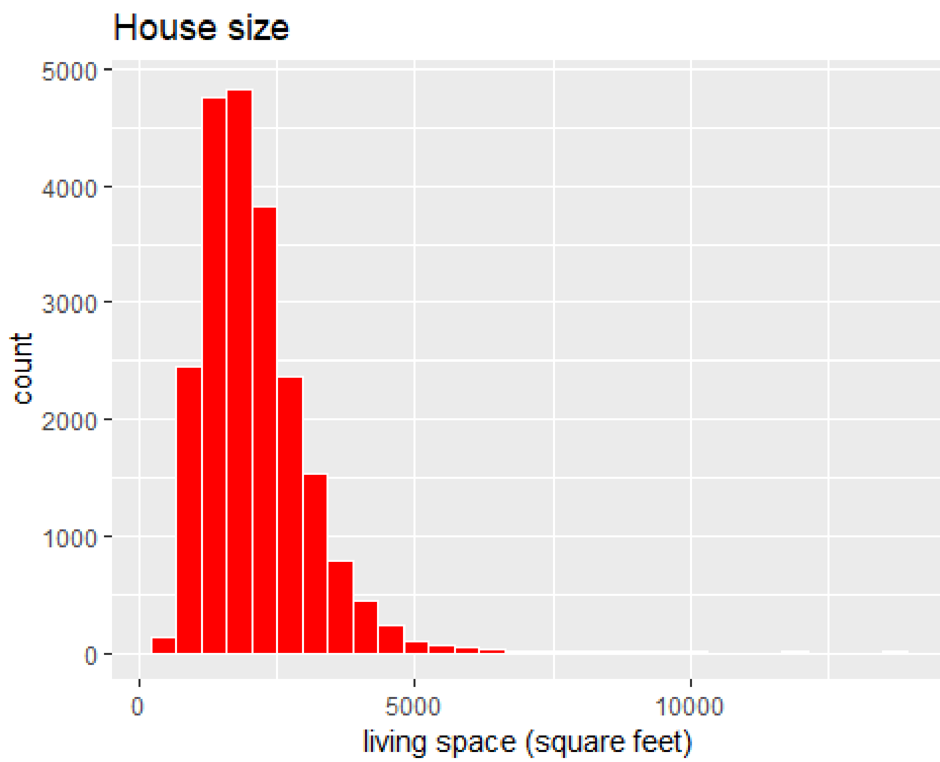
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
price	0	1	540088.1	367127.20	75000	321950	450000	645000	7700000	▇▁▁▁▁
sqft_living	0	1	2079.9	918.44	290	1427	1910	2550	13540	▇▂▁▁▁

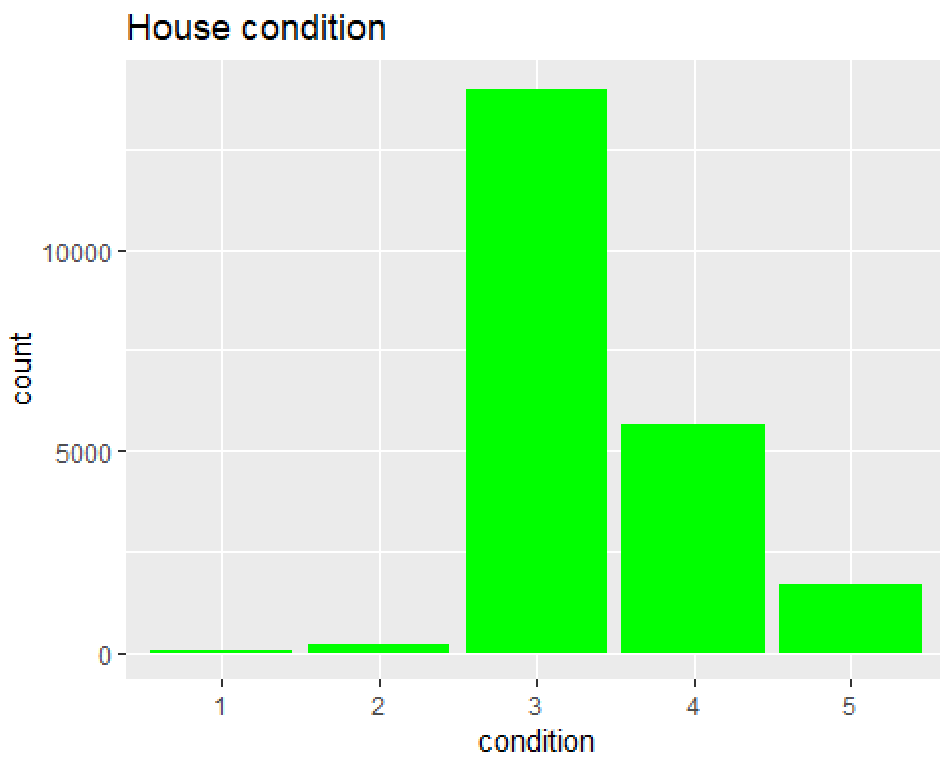
```r
library(cowplot)

# Histogram of house price:
p1 <- ggplot(house_prices, aes(x = price)) +
  geom_histogram(color = "white" , fill = "steelblue") +
  labs(x = "price (USD)", title = "House price")

p1

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



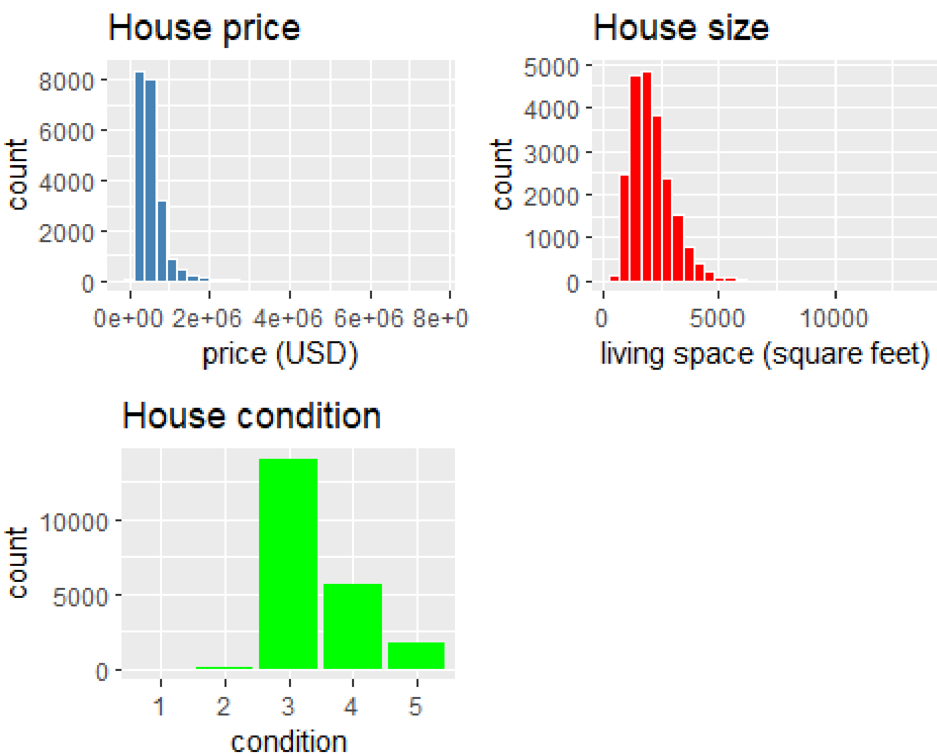

```
# Histogram of sqft_living:
p2 <- ggplot(house_prices, aes(x = sqft_living)) +
  geom_histogram(color = "white" , fill = "red") +
  labs(x = "living space (square feet)", title = "House size")

p2

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
# Barplot of condition:
p3 <- ggplot(house_prices, aes(x = condition)) +
  geom_bar(fill = "green") +
  labs(x = "condition", title = "House condition")

p3
```




```
plot_grid(p1 , p2 , p3)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



First, observe in the bottom plot that most houses are of condition "3", with a few more of conditions "4" and "5", and almost none that are "1" or "2".

Next, observe in the histogram for price in the top-left plot that a majority of houses are less than two million dollars. Observe also that the x-axis stretches out to 8 million dollars, even though there does not appear to be any houses close to that price. This is because there are a very small number of houses with prices closer to 8 million. These are the outlier house prices we mentioned earlier. We say that the variable price is right-skewed as exhibited by the long right tail.

Further, observe in the histogram of sqft_living in the middle plot as well that most houses appear to have less than 5000 square feet of living space. For comparison, a football field in the US is about 57,600 square feet, whereas a standard soccer/association football field is about 64,000 square feet. Observe also that this variable is also right-skewed, although not as drastically as the price variable.

For both the price and sqft_living variables, the right-skew makes distinguishing houses at the lower end of the x-axis hard. This is because the scale of the x-axis is compressed by the small number of quite expensive and immensely-sized houses.

So what can we do about this skew? Let's apply a log10 transformation to these variables.

Task : Let's create new log10 transformed versions of the right-skewed variable price and sqft_living using the mutate() function from Section 3.5, but we'll give the latter the name log10_size, which is shorter and easier to understand than the name log10_sqft_living .

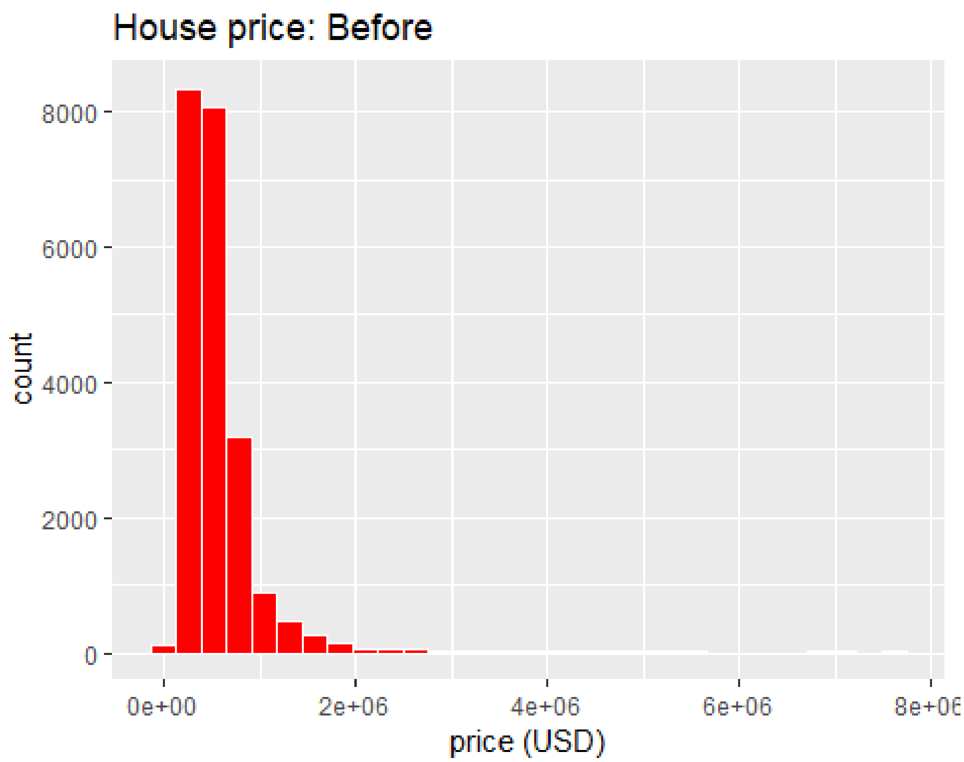
```
house_prices_l <- house_prices %>%
  mutate(
    log10_price = log10(price),
    log10_size = log10(sqft_living)
    )


house_prices_l %>%
  select(price, log10_price, sqft_living, log10_size) %>%
  head(10)
```

```
## # A tibble: 10 x 4
##      price log10_price sqft_living log10_size
##      <dbl>       <dbl>       <int>      <dbl>
##  1  221900        5.35        1180       3.07
##  2  538000        5.73        2570       3.41
##  3  180000        5.26         770       2.89
##  4  604000        5.78        1960       3.29
##  5  510000        5.71        1680       3.23
##  6 1225000        6.09        5420       3.73
##  7  257500        5.41        1715       3.23
##  8  291850        5.47        1060       3.03
##  9  229500        5.36        1780       3.25
## 10  323000        5.51        1890       3.28
```

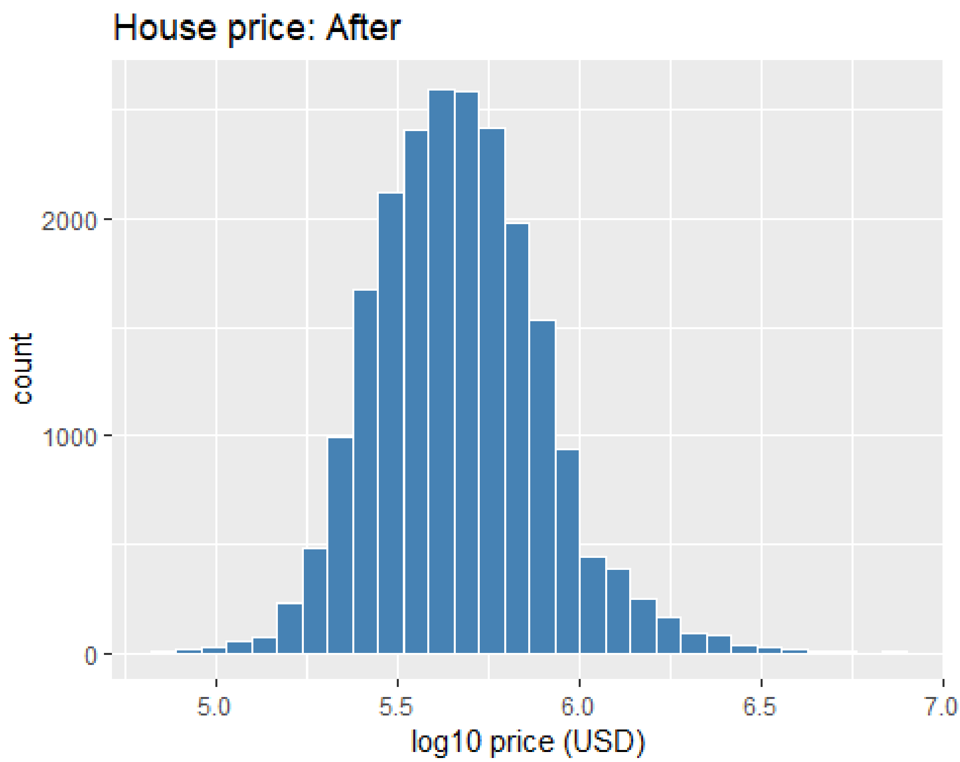
Task : Let's now visualize the before and after effects of this transformation for price.

```
# Before log10 transformation:
p1 <- ggplot(house_prices, aes(x = price)) +
  geom_histogram(color = "white" , fill = "red") +
  labs(x = "price (USD)", title = "House price: Before") ;p1
```

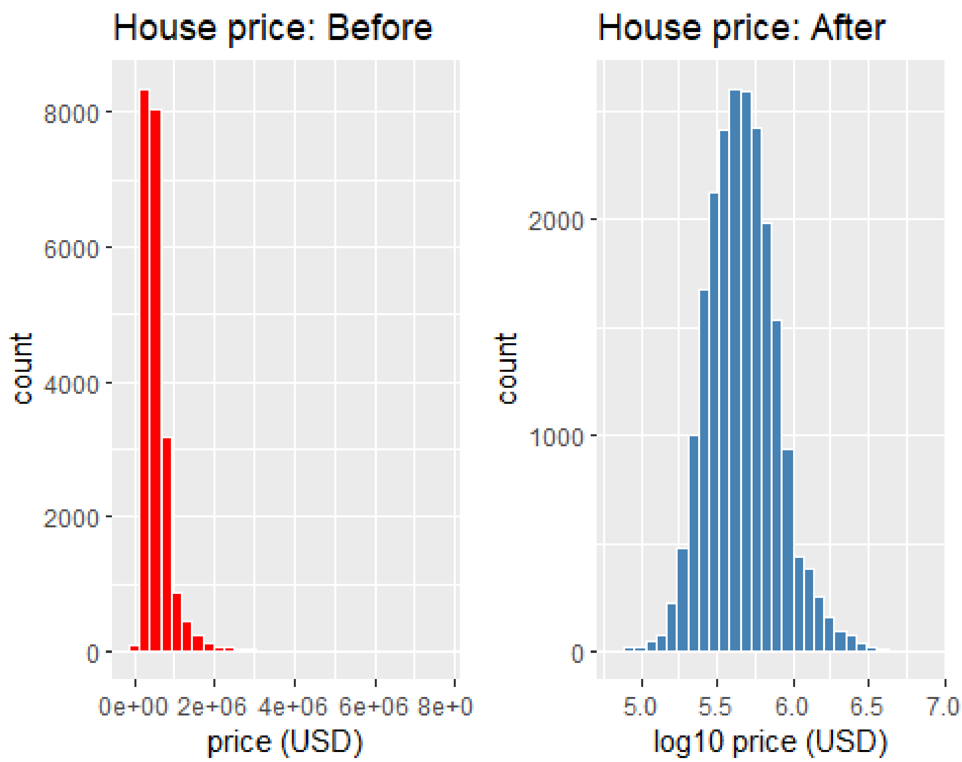
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
# After log10 transformation:
p2 <- ggplot(house_prices_l, aes(x = log10_price)) +
  geom_histogram(color = "white" , fill = "steelblue") +
  labs(x = "log10 price (USD)", title = "House price: After") ; p2

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
plot_grid(p1 , p2)

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



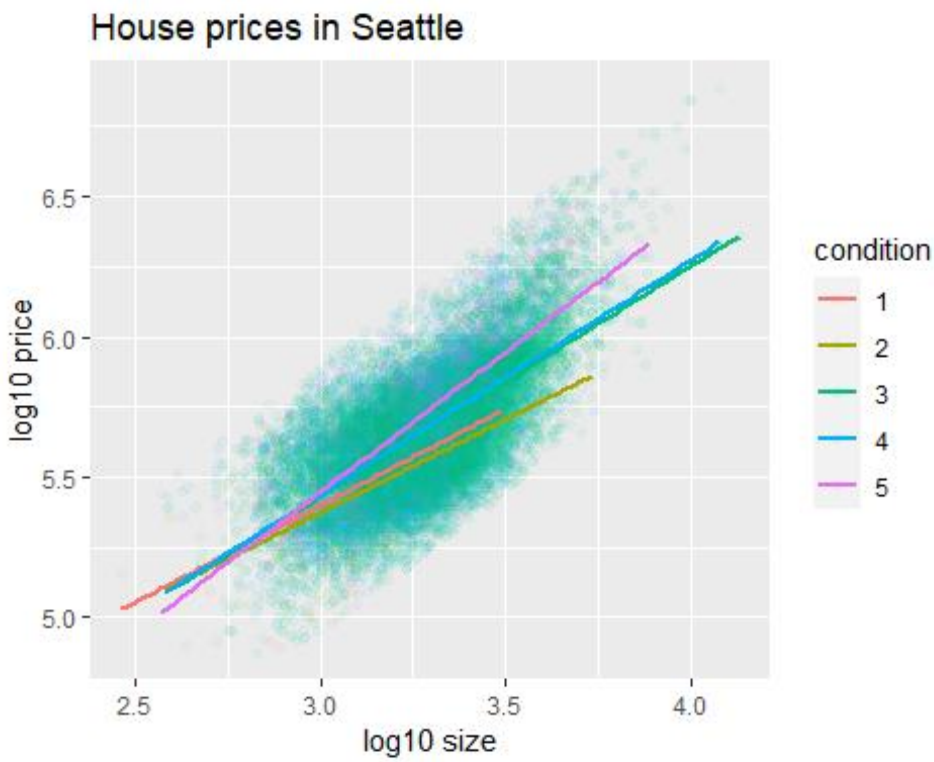
Observe that after the transformation, the distribution is much less skewed, and in this case, more symmetric and more bell-shaped.

3.2 Explaratoy Data Analysis : Part - II

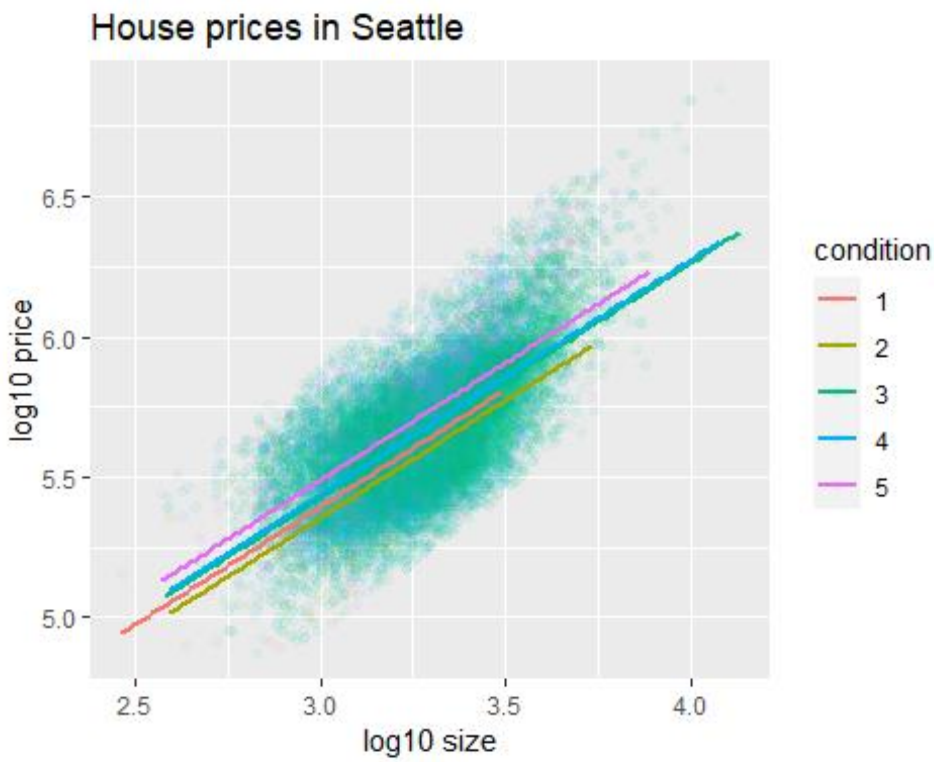
Multivariate Data

Visualization :

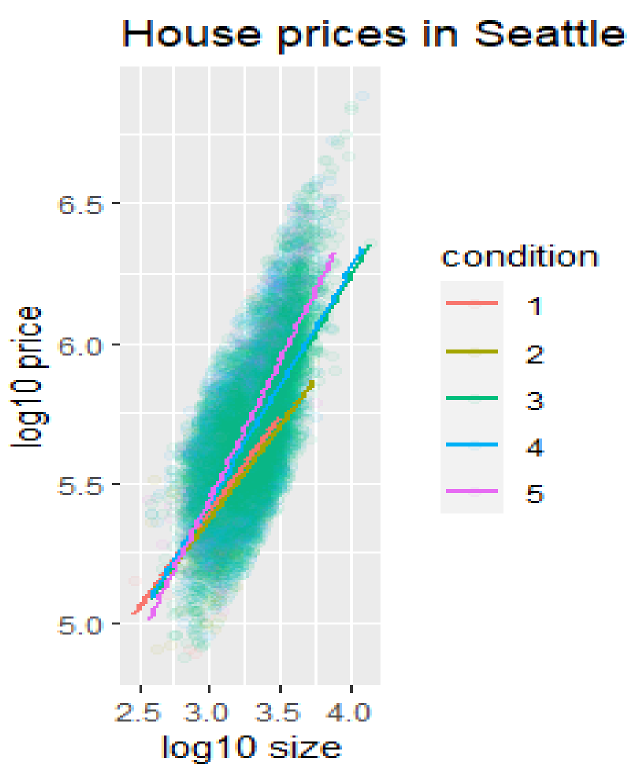
```
# Plot interaction model
p1 <- ggplot(house_prices_l,
       aes(x = log10_size, y = log10_price, col = condition)) +
  geom_point(alpha = 0.05) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(y = "log10 price",
       x = "log10 size",
       title = "House prices in Seattle")  ; p1

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

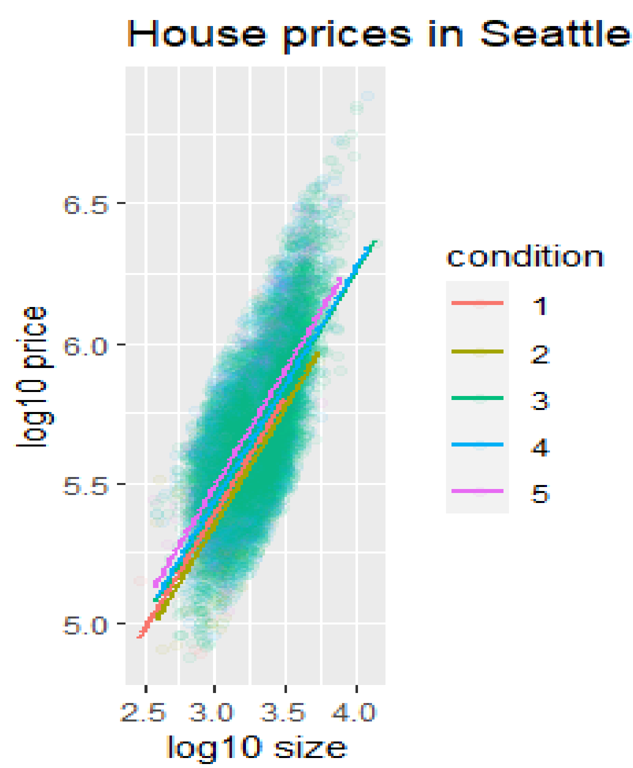



```r
# Plot parallel slopes model
p2 <- ggplot(house_prices_l,
       aes(x = log10_size, y = log10_price, col = condition)) +
  geom_point(alpha = 0.05) +
  geom_parallel_slopes(se = FALSE) +
  labs(y = "log10 price",
       x = "log10 size",
       title = "House prices in Seattle") ; p2
```

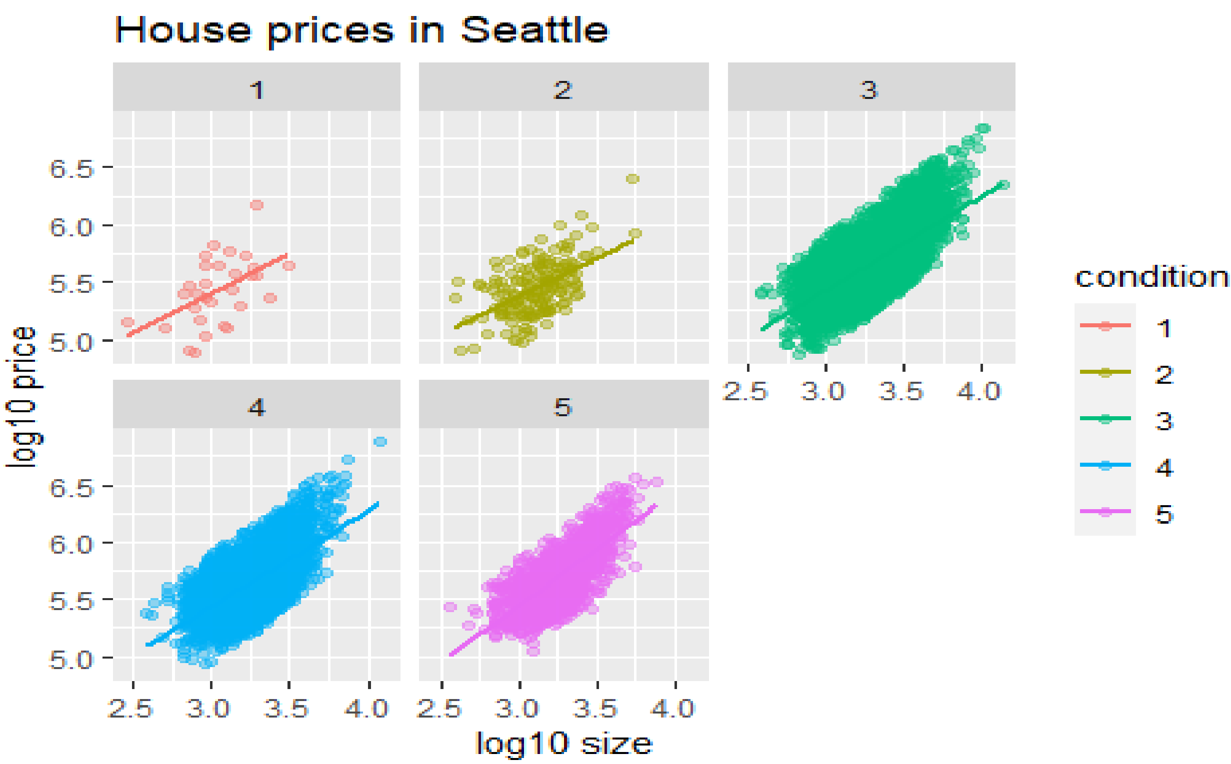



```
plot_grid(p1 , p2)

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





```r
ggplot(house_prices_l,
       aes(x = log10_size, y = log10_price, col = condition)) +
  geom_point(alpha = 0.4) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(y = "log10 price",
       x = "log10 size",
       title = "House prices in Seattle") +
  facet_wrap(~ condition)

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



Regression Analysis :

Model : $log10\_price = \beta_0 + \beta_1(log10\_size * condition) + \epsilon$ $log10\_price = \beta_0 + \beta_{size} \times log10(size) + \epsilon$

```r
# Fit regression model:
price_interaction <- lm(log10_price ~ log10_size * condition,
                        data = house_prices_l)

# Get regression table:
get_regression_table(price_interaction)

## # A tibble: 10 x 7
##    term                  estimate std_error statistic p_value lower_ci
upper_ci
##    <chr>                    <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
```

```
<dbl>
##  1 intercept                3.33      0.451     7.38    0         2.45
4.22
##  2 log10_size               0.69      0.148     4.65    0         0.399
0.98
##  3 condition: 2             0.047     0.498     0.094   0.925    -0.93
1.02
##  4 condition: 3            -0.367     0.452    -0.812   0.417    -1.25
0.519
##  5 condition: 4            -0.398     0.453    -0.879   0.38     -1.29
0.49
##  6 condition: 5            -0.883     0.457    -1.93    0.053    -1.78
0.013
##  7 log10_size:condition2   -0.024     0.163    -0.148   0.882    -0.344
0.295
##  8 log10_size:condition3    0.133     0.148     0.893   0.372    -0.158
0.424
##  9 log10_size:condition4    0.146     0.149     0.979   0.328    -0.146
0.437
## 10 log10_size:condition5    0.31      0.15      2.07    0.039     0.016
0.604
```

Interpretation :

$$\widehat{log10\_price} = \hat{\beta_0} + \hat\beta_{size}\times log10(size)$$

1. Condition 1 : $\widehat{log10\_price} = 3.33 + 0.69 \times log10(size)$
2. Condition 2 : $\widehat{log10\_price} = 3.33 + (0.69 - 0.024) \times log10(size) = \ 0.666 \times log10(size)$
3. Condition 3 : $\widehat{log10\_price} = 3.33 + (0.69 + 0.133) \times log10(size) = 0.823 \times log10(size)$
4. Condition 4 : $\widehat{log10\_price} = 3.33 + (0.69 + 0.146) \times log10(size) = 0.836 \times log10(size)$
5. Condition 5 : $\widehat{log10\_price} = 3.33 + (0.69 + 0.31) \times log10(size) = 1.0 \times log10(size)$

For homes of all five condition types, as the size of the house increases, the price increases. This is what most would expect. However, the rate of increase of price with size is fastest for the homes with conditions 3, 4, and 5 of 0.823, 0.836, and 1, respectively. These are the three largest slopes out of the five.

Making Predictions :

Say we're a retailor and someone calls you asking you how much their home will sell for. They tell you that it's in condition = 5 and is sized 1900 square feet. What do you tell them ? Let's use the interaction model we fit to make predictions!

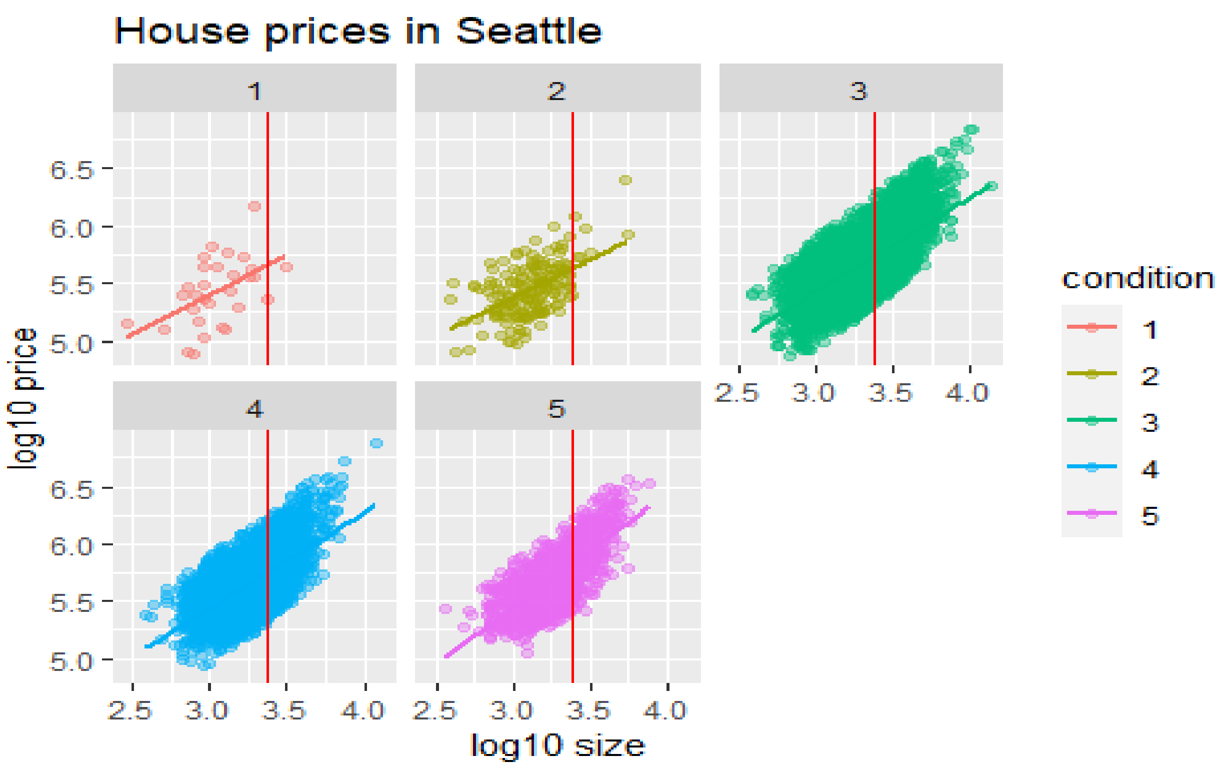
We first make this prediction visually in Figure 11.8. The predicted log10_price of this house is marked with a black dot. This is where the following two lines intersect:

The regression line for the condition = 5 homes and The vertical dashed black line at log10_size equals 3.28, since our predictor variable is the log10 transformed square feet of living space of $log_{10}(1900) = 3.28$

```
# Without Facet
ggplot(house_prices_l, aes(x = log10_size, y = log10_price, col = condition))
+
  geom_point(alpha = 0.4) +
  geom_smooth(method = "lm", se = FALSE) + geom_vline(xintercept = 3.38 , col
= "red" , tly = 2)  +
  labs(y = "log10 price" ,
       x = "log10 size",
       title = "House prices in Seattle")

## Warning: Ignoring unknown parameters: tly

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


```
# With facet
ggplot(house_prices_l, aes(x = log10_size, y = log10_price, col = condition))
+
  geom_point(alpha = 0.4) +
  geom_smooth(method = "lm", se = FALSE) + geom_vline(xintercept = 3.38 , col
= "red" , tly = 2)  +
```

```
  labs(y = "log10 price" ,
       x = "log10 size",
       title = "House prices in Seattle") +
facet_wrap(~ condition)

## Warning: Ignoring unknown parameters: tly

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



Eyeballing it, it seems the predicted log10_price seems to be around 5.75. Let's now obtain the exact numerical value for the prediction using the equation of the regression line for the condition = 5 houses, being sure to log10() the square footage first.

```
2.45 + 1 * log10(1900)

## [1] 5.728754
```

This value is very close to our earlier visually made prediction of 5.75. But wait! Is our prediction for the price of this house $5.75 ? No! Remember that we are using log10_price as our outcome variable! So, if we want a prediction in dollar units of price, we need to unlog this by taking a power of 10 .

```
10^(2.45 + 1 * log10(1900))

## [1] 535492.8
```

(LC 11.1) Prediction making we just did on the house of condition 5 and size 1900 square feet. Show that it's $524,807!

```
house_prices_l <- house_prices %>%
  mutate(
    log10_price = log10(price),
    log10_size = log10(sqft_living)
  )
# Fit regression model:
price_interaction <- lm(log10_price ~ log10_size + condition,
  data = house_prices_l)

# Get regression table:
get_regression_table(price_interaction)

## # A tibble: 6 x 7
##   term         estimate std_error statistic p_value lower_ci upper_ci
##   <chr>           <dbl>     <dbl>     <dbl>   <dbl>    <dbl>    <dbl>
## 1 intercept       2.88      0.036     80.0    0         2.81     2.95
## 2 log10_size      0.837     0.006    134.     0         0.825    0.85
## 3 condition: 2   -0.039     0.033     -1.16   0.246    -0.104    0.027
## 4 condition: 3    0.032     0.031      1.04   0.3      -0.028    0.092
## 5 condition: 4    0.044     0.031      1.42   0.155    -0.017    0.104
## 6 condition: 5    0.096     0.031      3.09   0.002     0.035    0.156

10^(2.88 + 0.096 + 0.837 * log10(1900))

## [1] 525190.4
```

Complete