Linear Regression Model- Facebook Data

Contents

Setup

```
library(tidyverse)
library(broom)
library(digest)
library(testthat)
```

Load Data

```
facebook_data <- read_csv("data/facebook_data.csv")
facebook_sampling_data <- read_csv("data/facebook_sampling_data.csv")
```

The Facebook Dataset

This dataset is related to posts' critical information on user engagement during 2014 on a Facebook page of a famous cosmetics brand. The original dataset contains 500 observations relative to different classes of posts, and it can be found in data.world. After some data cleaning, it ends up with 491 observations. The dataset was firstly analyzed by Moro et al. (2016) in their data mining work to predict the performance of different post metrics, which are also based on the type of post. The original dataset has 17 different continuous and discrete variables. Nonetheless, for this lab, we extracted five variables for `facebook_data` as follows:

1. The continuous variable `total_engagement_percentage` is an essential variable for any company owning a Facebook page. It gives a sense of how engaged the overall social network's users are with the company's posts, **regardless of whether they previously liked their Facebook page or not**. *The larger the percentage, the better the total engagement.* It is computed as follows:

$$\texttt{total\_engagement\_percentage} = \frac{\text{Lifetime Engaged Users}}{\text{Lifetime Post Total Reach}} \times 100\%$$

- Lifetime Post Total Reach: The number of overall *Facebook unique users* who *saw* the post. **- Lifetime Engaged Users:** The number of overall *Facebook unique users* who *saw and clicked* on the post. This count is a subset of **Lifetime Post Total Reach**.

2. The continuous variable `page_engagement_percentage` is analogous to `total_engagement_percentage`, but only with users who engaged with the post **given they have liked the page**. This variable provides a sense to the company to what extent these subscribed users are reacting to its posts. *The larger the percentage, the better the page engagement.* It is computed as follows:

$$\texttt{page\_engagement\_percentage} = \frac{\text{Lifetime Users Who Have Liked the Page and Engaged with the Post}}{\text{Lifetime Post Reach by Users Who Liked the Page}} \times 100\%$$

- Lifetime Post Reach by Users Who Liked the Page: The number of *Facebook unique page subscribers* who *saw* the post. **- Lifetime Users Who Have Liked the Page and Engaged with the Posts:** The number of *Facebook unique page subscribers* who *saw and clicked* on the post. This count is a subset of **Lifetime Post Reach by Users Who Liked the Page**.

3. The continuous `share_percentage` is the percentage that the number of *shares* represents from the sum of *likes*, *comments*, and *shares* in each post. It is computed as follows:

$$\texttt{share\_percentage} = \frac{\text{Number of Shares}}{\text{Total Post Interactions}} \times 100\%$$

- Total Post Interactions: The sum of *likes*, *comments*, and *shares* in a given post. **- Number of Shares:** The number of *shares* in a given post. This count is a subset of *Total Post Interactions.*

4. The continuous `comment_percentage` is the percentage that the number of *comments* represents from the sum of *likes*, *comments*, and *shares* in each post. It is computed as follows:

$$\texttt{comment\_percentage} = \frac{\text{Number of Comments}}{\text{Total Post Interactions}} \times 100\%$$

- Total Post Interactions: The sum of *likes*, *comments*, and *shares* in a given post. **- Number of Comments:** The number of *comments* in a given post. This count is a subset of *Total Post Interactions.*

5. The discrete and nominal variable `post_category` has three different categories depending on the content characterization:

- `Action`: Brand's contests and special offers for the customers.
- `Product`: Regular advertisements for products with explicit brand content.
- `Inspiration`: Non-explicit brand-related content.

Inference in a Simple Linear Regression (SLR) Model

$$Y_i = \beta_0 + \beta_1 \times X_i + \epsilon_i$$

Y_i: Here total_engagement_percentage is the assessed value

X_i: Here page_engagement_percentage is the explanatory variable

β_0: is the population y-intercept which represents the total_engagement_percentage when page_engagement_percentage is 0

β_1: is the population slope which represents the change in the total_engagement_percentage associated with a unit change in page_engagement_percentage

ϵ_i: is the error term which contains all other factors affecting Yi other than Xi.

Now, with the variables from the previous regression equation, we estimate the SLR called `SL_reg` using the function `lm()`:

```
SL_reg <- lm(total_engagement_percentage ~ page_engagement_percentage, facebook_data)
SL_reg
```

```
##
## Call:
## lm(formula = total_engagement_percentage ~ page_engagement_percentage,
##     data = facebook_data)
##
## Coefficients:
##                (Intercept)  page_engagement_percentage
##                    -0.6711                      1.0288
```

Inference and Prediction in a Multiple Linear Regression (MLR) Model

Now, suppose we are interested in building a MLR to explain the variation observed in `total_engagement_percentage`, using the variables `page_engagement_percentage`, `share_percentage`, and `comment_percentage`.

$$Y_i = \beta_0 + \beta_1 \times X_{1i} + \beta_2 \times X_{2i} + \beta_3 \times X_{3i} + \epsilon_i$$

Y_i: Here total_engagement_percentage is the assessed value.

X_{1i}: Here page_engagement_percentage is the first explanatory variable

X_{2i}: Here share_percentage is the second explanatory variable

X_{3i}: Here comment_percentage is the third explanatory variable

β_0:is the sample intercept i.e. the total_engagement_percentage when the page_engagement_percentage, share_percentage and comment_percentage are 0

β_1: is the slope coefficient corresponding to X_1 which represents the change in the total_engagement_percentage with a unit change in page_engagement_percentage holding other variables constant.

β_2: is the slope coefficient corresponding to X_2 which represents the change in the total_engagement_percentage with a unit change in share_percentage holding other variables constant.

β_3: is the slope coefficient corresponding to X_3 which represents the change in the total_engagement_percentage with a unit change in comment_percentage holding other variables constant.

ϵ_i: is the error term which contains all other factors affecting Y_i other than X_1, X_2 and X_3.

We fit the MLR model using `lm()` and assign it to the object `ML_reg`.

```
ML_reg <- lm(total_engagement_percentage~page_engagement_percentage+share_percentage+comment_percentage
ML_reg
```

```
##
## Call:
## lm(formula = total_engagement_percentage ~ page_engagement_percentage +
##     share_percentage + comment_percentage, data = facebook_data)
##
## Coefficients:
##                (Intercept)  page_engagement_percentage
##                   -1.29650                     1.01558
##           share_percentage          comment_percentage
##                    0.05497                    -0.01087
```

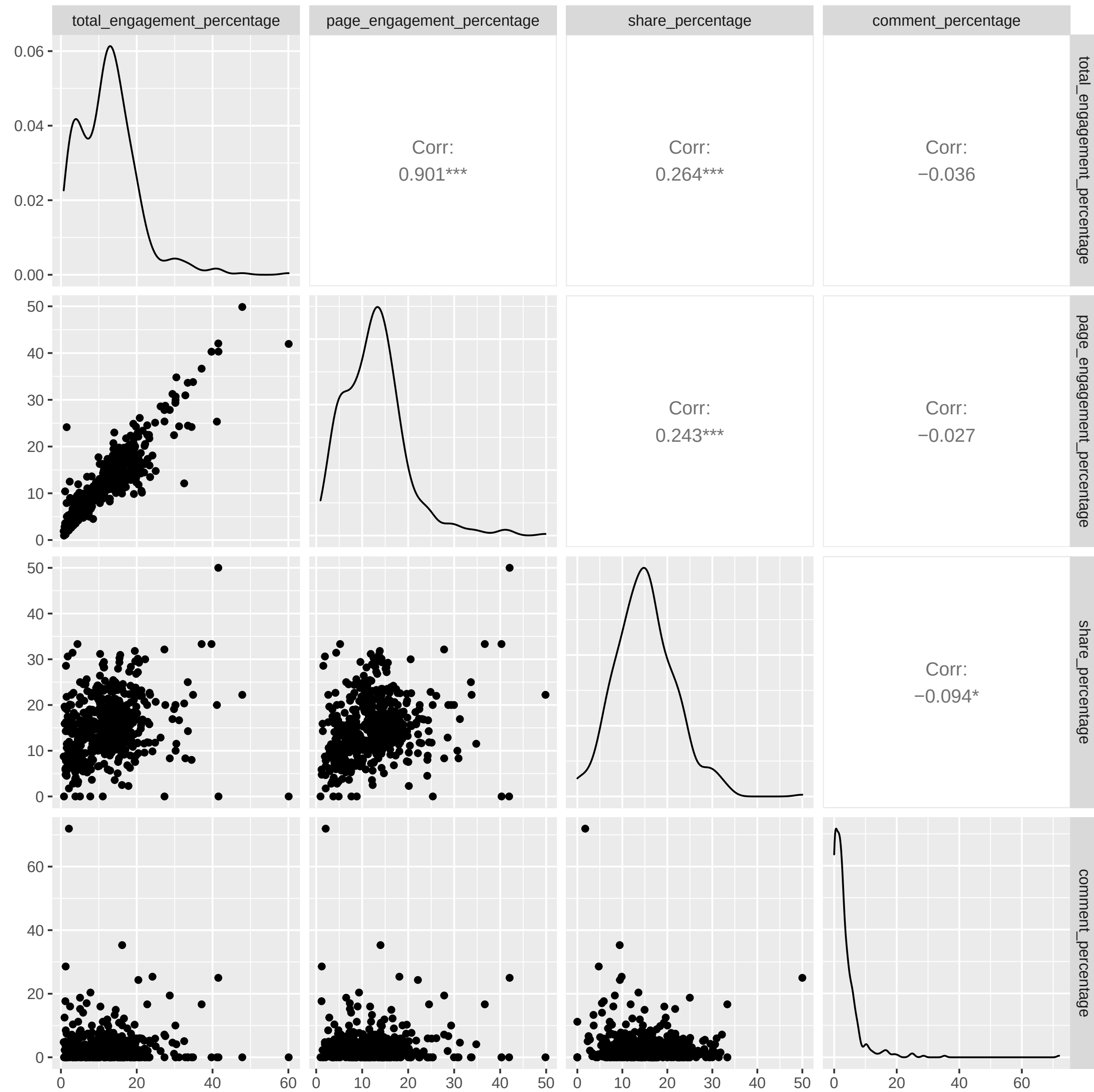
We use `ggpairs()` from `GGally` to generate a pair plot **of the variables used in `ML_reg`**. Observe the relationship between the response and explanatory variables, as well as the relationships between the explanatory variables themselves.

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
GGally::ggpairs(facebook_data %>% select_if(is.numeric), progres= FALSE)
```

```
## Warning in warn_if_args_exist(list(...)): Extra arguments: 'progres' are being
## ignored. If these are meant to be aesthetics, submit them using the 'mapping'
## variable within ggpairs with ggplot2::aes or ggplot2::aes_string.
```



total__engagement__percentage and page__engagement__percentage have the highest linear correlation as seen from the plot above i.e. 0.901 i.e. high engagement in page interaction leads to high total engagement.

We find the estimated coefficients of `ML_reg` using `tidy()`. Report the estimated coefficients, their standard errors and corresponding p-values and bind our results to the variable `tidy_ML_reg`.

```
tidy_ML_reg <- tidy(ML_reg)

tidy_ML_reg
```

```
## # A tibble: 4 x 5
##   term                          estimate std.error statistic   p.value
##   <chr>                            <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)                     -1.30      0.453     -2.86  4.36e-  3
```

```
## 2 page_engagement_percentage   1.02      0.0230    44.1   4.23e-172
## 3 share_percentage              0.0550    0.0238     2.31  2.13e-  2
## 4 comment_percentage           -0.0109    0.0295    -0.368 7.13e-  1
```

At a significance level of $\alpha = 0.05$

The `page_engagement_percentage` estimated coefficient i.e. $\hat{\beta}_1$ and `share_percentage` estimated coefficient i.e. $\hat{\beta}_2$ are statistically significant because their p-value is lesser than $\alpha = 0.05$ and hence we can reject the null hypothesis, while the p-value for $\hat{\beta}_3$ i.e.`comment_percentage` estimate coefficient is greater than alpha hence we fail to reject the null hypothesis hence the feature is not statistically significant.

Now we use both the `SL_reg` and `ML_reg` to make predictions of their response. Plot the **in-sample** *predicted values* on the y-axis versus the *observed values* on the x-axis of the response (using `geom_point()`) to check the goodness of fit of the two models **visually**. We can assess this by putting a 45° *dashed* line on our plot (`geom_abline()`). A perfect prediction will be exactly located on this 45° degree line.

Firstly, we need to put both sets of in-sample predictions in a single data frame called `predicted_response`, where each row represents a predicted value, with three columns (*from left to right*):

- `total_engagement_percentage` (the observed response in `facebook_data` associated to each prediction).
- `Model` (with label `SLR` or `MLR`),
- `predicted_percentage` (the in-sample prediction), and

Then, we can proceed with the plot using `predicted_response`. Note that we have to colour the points by `Model`. Assign our plot to an object called `prediction_plot`.

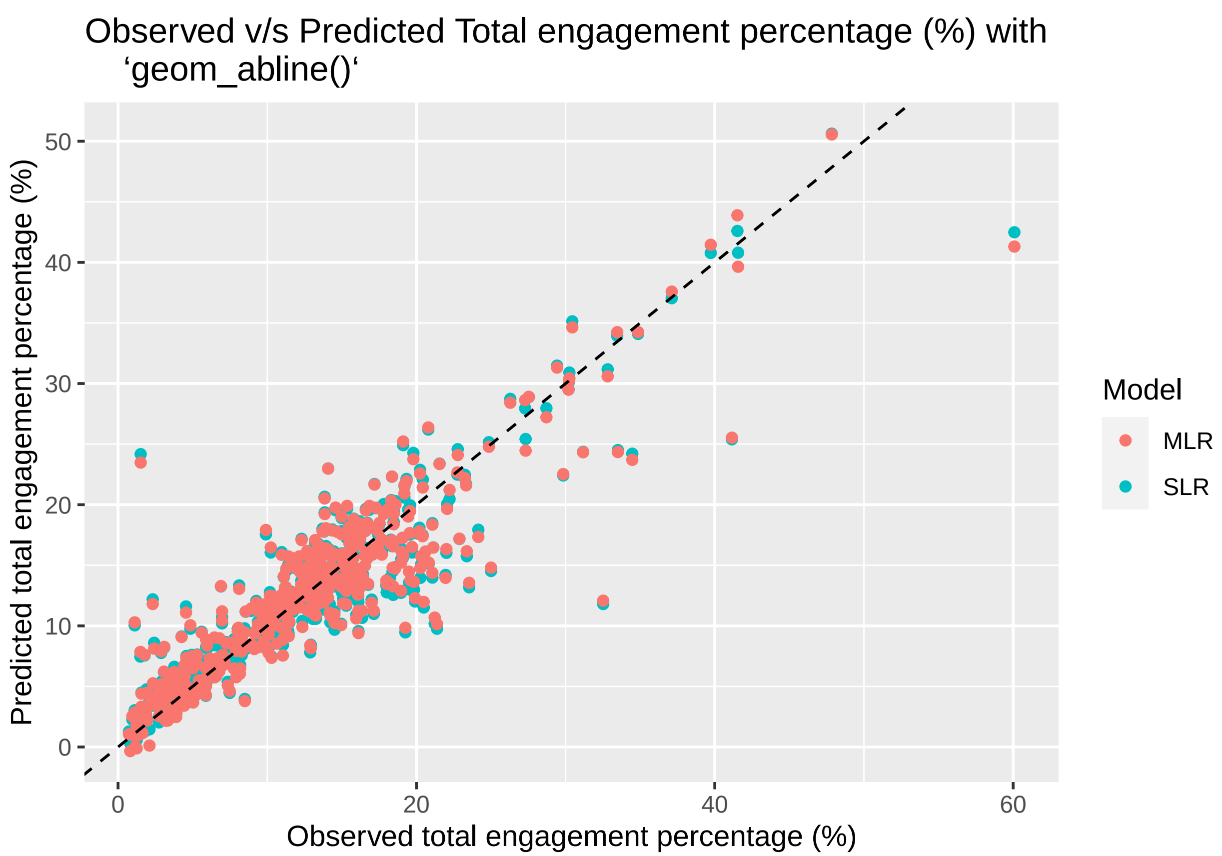
```
predict_SL <- predict(SL_reg, facebook_data[2])
predict_ML <- predict(ML_reg, facebook_data[2:4])
```

```
predicted_response <- tibble(total_engagement_percentage= rep(facebook_data$total_engagement_percentage


predicted_response
```

```
## # A tibble: 982 x 3
##    total_engagement_percentage Model predicted_percentage
##                          <dbl> <fct>                <dbl>
##  1                        6.47 SLR                   6.79
##  2                       13.9  SLR                  18.0
##  3                        7.34 SLR                   8.36
##  4                        4.41 SLR                   3.78
##  5                        9.26 SLR                  12.1
##  6                       11.4  SLR                  12.6
##  7                        4.11 SLR                   3.51
##  8                        3.91 SLR                   3.26
##  9                       12.9  SLR                  15.6
## 10                        5.97 SLR                   8.14
## # ... with 972 more rows
```

```
prediction_plot <- ggplot(
  predicted_response,
  aes(
```

```
    x= total_engagement_percentage, y=predicted_percentage, color= Model
  )
) +
  geom_point() +
  geom_abline( linetype = "dashed") +
  labs(
    title = "Observed v/s Predicted Total engagement percentage (%) with
    `geom_abline()`",
    x = "Observed total engagement percentage (%)",
    y = "Predicted total engagement percentage (%)"
  )



prediction_plot
```



Looking at the data points above the both the models do not show much difference in the predictions, also there is no much visual difference between the two models, because additional features are not correlated to total_engagement_percentage as much as the single feature page_engagement_percentage.But both SLR and MLR have the feature page_engagement_percentage in common.

Bootstrapping

Until now, we have been using asymptotic theory/Central Limit Theorem (CLT) to approximate the sampling distribution of the estimators of the regression coefficients as a normal distribution centred around the true coefficients. This asymptotic distribution was used to make inference about the true coefficients of our linear model. While this is a good approach for a wide variety of problems, it may be inappropriate if, for example, the underlying distribution is extremely skewed or your sample size is small ($n < 30$). Bootstrapping is an alternative (non-parametric approach) to approximate the sampling distribution needed to assess the uncertainty of our estimated coefficients and make inference. We will work with a sample of $n = 50$ observations from the `facebook_dataset`.

Here we draw a random sample of size $n = 50$ from the original `facebook_data`.

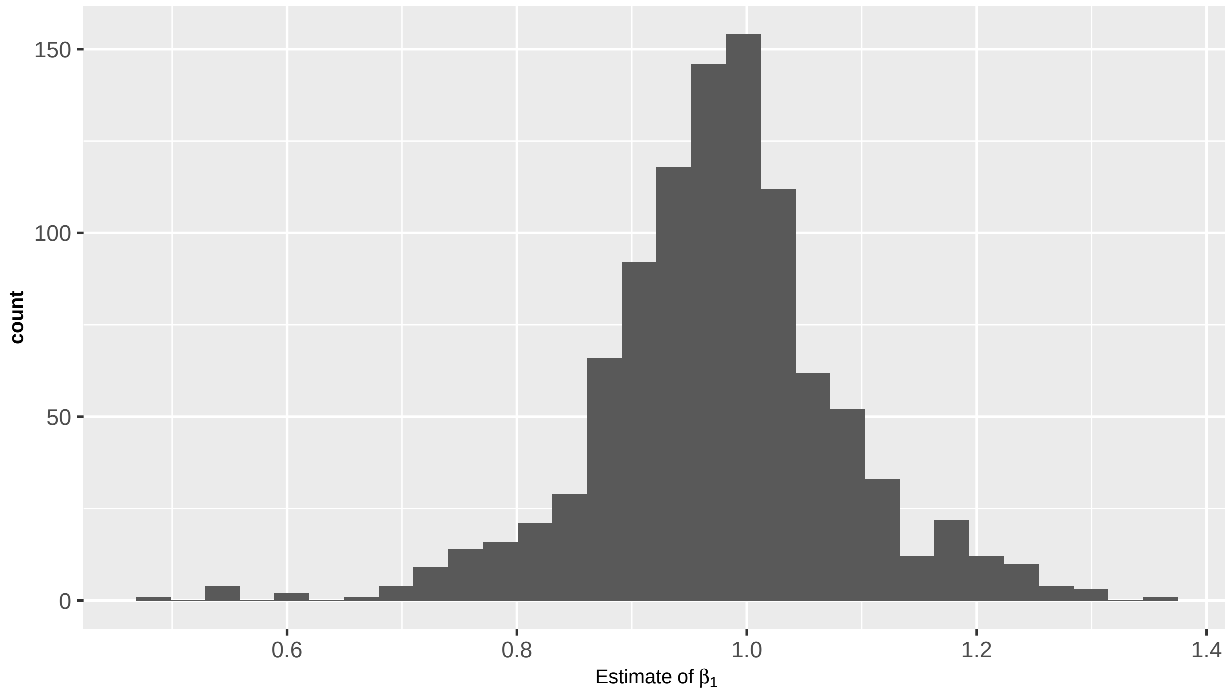
```
set.seed(1234)
facebook_base_sample <- sample_n(facebook_data, 50)
```

```
N <- 1000

set.seed(1234)

boot_fits <- facebook_base_sample %>%
  rsample::bootstraps(times = N) %>%
  mutate(
    lm = map(splits, ~
    lm(total_engagement_percentage ~ page_engagement_percentage,
      data = .x
    )),
    tidy = map(lm, broom::tidy)
  ) %>%
  select(-splits, -lm) %>%
  unnest(tidy) %>%
  filter(term == "page_engagement_percentage") %>%
  select(-term)
boot_fits
```

```
## # A tibble: 1,000 x 5
##    id            estimate std.error statistic  p.value
##    <chr>            <dbl>     <dbl>     <dbl>    <dbl>
##  1 Bootstrap0001    1.00      0.0397     25.2  2.56e-29
##  2 Bootstrap0002    1.06      0.0517     20.6  1.77e-25
##  3 Bootstrap0003    0.999     0.0742     13.5  6.36e-18
##  4 Bootstrap0004    1.13      0.0576     19.5  1.74e-24
##  5 Bootstrap0005    1.01      0.0361     27.9  2.38e-31
##  6 Bootstrap0006    0.926     0.100       9.27 2.88e-12
##  7 Bootstrap0007    0.916     0.0728     12.6  8.31e-17
##  8 Bootstrap0008    1.04      0.0463     22.4  4.61e-27
##  9 Bootstrap0009    1.03      0.0490     21.0  8.11e-26
## 10 Bootstrap0010    0.919     0.0798     11.5  2.00e-15
## # ... with 990 more rows
```

```
ggplot(boot_fits, aes(estimate)) +
  geom_histogram(bins = 30) +
  xlab(expression(Estimate ~ of ~ beta[1])) +
```

```
  theme(
    plot.title = element_text(face = "bold", size = 8, hjust = 0.5),
    axis.title = element_text(face = "bold", size = 8)
  )
```



The code calculates the SLR regression coefficients, std.error, t-statistic and the p-value for each bootstrap sample i.e. 1000 bootstrap samples.The rsample function nests all the bootstrap samples after creating them , the lm function maps to each bootstrap sample and applies lm i.e. linear regression between `total_engagement_percentage` and `page_engagement_percentage` and calculates the intercept and the slope coefficients and stores it as a tibble. The tidy function calculates the test statistic and the p- value etc per sample, we then unnest it to get the values per sample, we use the filter to remove the intercepts and store only the slope coefficients in the estimate column.

After that we plot the 1000 slope coefficients using ggplot.

Since the sample size is large enough i.e. 50 samples ($n > 30$),so the bootstrap sampling distribution's is bell shaped i.e. a normal distribution.

Estimate the mean of the slope's estimator, $\hat{\beta}_1$, based on your bootstrap **coefficient** estimates in `boot_fits` and call it `boot_mean`.

Then, estimate the SLR called `SL_reg_n50` using the function `lm()` using the sample of 50 observations from the dataset `facebook_base_sample` with `total_engagement_percentage` and `page_engagement_percentage` as response and explanatory variable, respectively. Assign you model's `tidy()` output to the object `tidy_SL_reg_n50`.

```
boot_mean <- mean(boot_fits$estimate)
SL_reg_n50 <- lm(total_engagement_percentage~page_engagement_percentage, facebook_base_sample)
tidy_SL_reg_n50 <- tidy(SL_reg_n50)

boot_mean
```

```
## [1] 0.97647
```

```r
tidy_SL_reg_n50
```

```
## # A tibble: 2 x 5
##   term                        estimate std.error statistic  p.value
##   <chr>                          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)                  -0.0555    1.24     -0.0448 9.64e- 1
## 2 page_engagement_percentage   0.976     0.0764    12.8    4.65e-17
```

The boot_mean is very similar to the estimated **slope** in SL_reg_n50 i.e. the one calculated from the original sample. Since we have taken 1000 random samples with repetitions by bootstrapping on the original sample the the bootstrap point estimate is very close to the sample point estimate.

Now we use the `quantile()` function to calculate the 95% CI from our bootstrap **coefficient** estimates in dataset `boot_fits` and then bind our CI bounds to the vector `boot_ci`.

Then, use the `conf.int = TRUE` argument in the `tidy()` function to find the 95% confidence interval calculated by `lm()` using the sample `facebook_base_sample` of 50 observations (without bootstrapping) for the estimated **slope** from object `SL_reg_n50`. Reassign the output to `tidy_SL_reg_n50`.

```r
boot_ci <- quantile(boot_fits$estimate,probs = c(0.025,0.975))
tidy_SL_reg_n50 <- tidy(SL_reg_n50, conf.int = TRUE)


boot_ci
```

```
##      2.5%     97.5%
## 0.7545946 1.2058666
```

```r
tidy_SL_reg_n50
```

```
## # A tibble: 2 x 7
##   term                   estimate std.error statistic  p.value conf.low conf.high
##   <chr>                     <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 (Intercept)             -0.0555    1.24     -0.0448 9.64e- 1   -2.54       2.43
## 2 page_engagement_perc~    0.976     0.0764    12.8    4.65e-17    0.822      1.13
```

Then we create a two-row data frame called `ci_data` containing the coefficient CIs with the following columns (*from left to right*):

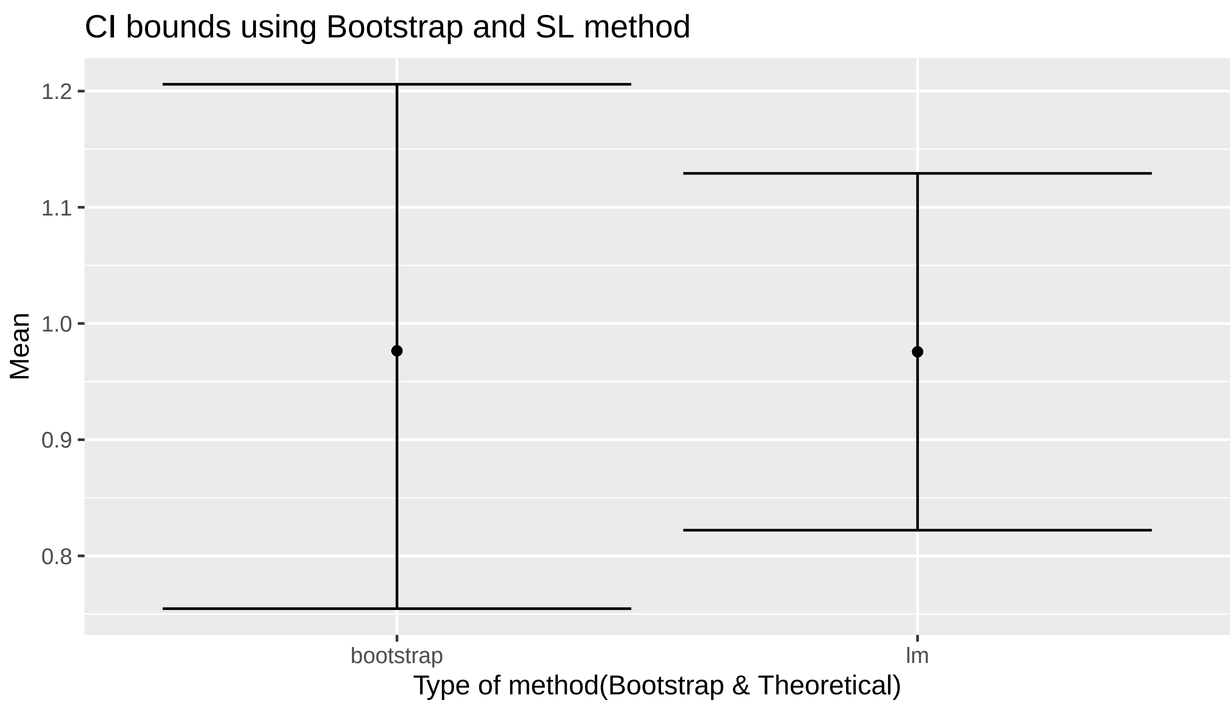
- `type`, the label `"bootstrap"` for the CI in `boot_ci` and `"lm"` for the one in `tidy_SL_reg_n50`.
- `mean`, the center of each CI (`boot_mean` for the bootrapping CI and the `estimate` found `tidy_SL_reg_n50`).
- `conf.low`, the respective lower bound in `boot_ci` and `tidy_SL_reg_n50`.
- `conf.high`, the respective upper bound in `boot_ci` and `tidy_SL_reg_n50`.

```r
ci_data <- tibble(type = c("bootstrap", "lm"), mean= c(boot_mean,tidy_SL_reg_n50$estimate[2]),
                  conf.low = c(boot_ci[1], tidy_SL_reg_n50$conf.low[2]),conf.high = c(boot_ci[2], tidy_S


ci_data
```

```
## # A tibble: 2 x 4
##   type       mean conf.low conf.high
##   <chr>     <dbl>    <dbl>     <dbl>
## 1 bootstrap 0.976    0.755      1.21
## 2 lm        0.976    0.822      1.13
```

To compare graphically both CIs from `ci_data` we plot `type` on the x-axis versus `mean` on the y-axis **as points**. Then, plot the CI bounds **using the corresponding ggplot2 function** and assign our plot to an object called `ci_plot`.

```
ci_plot <- ggplot(ci_data) +
    aes(x = type,
        y = mean) +geom_point()+
    geom_errorbar(aes(ymin=conf.low, ymax=conf.high))+
  labs(x= "Type of method(Bootstrap & Theoretical)", y ="Mean", title = "CI bounds using Bootstrap and S


ci_plot
```



LR with a Categorical Variable with More than Two Levels

Here we will be using `facebook_data`, we will fit a LR model with `total_engagement_percentage` as a response and `post_category` as a categorical explanatory variable.

Three parameters are needed to be estimated in this LR model relating `total_engagement_percentage` with post_category i.e. $\hat{\beta_{action}}$ i.e. the intercept ,$\hat{\beta_{inspiration}}$,$\hat{\beta_{product}}$, because there are three post categories in the given data and the action category will be considered as baseline. $\hat{\beta_{action}}$ indicates the `total_engagement_percentage` when the category is action. $\hat{\beta_{inspiration}}$ indicates the change in `total_engagement_percentage` when the category changes from action to inspiration. $\hat{\beta_{product}}$ indicates the change in `total_engagement_percentage` when the category changes from action to product.

Now we fit a LR model, called `LR_post_category`, that relates `total_engagement_percentage` to `post_category`.

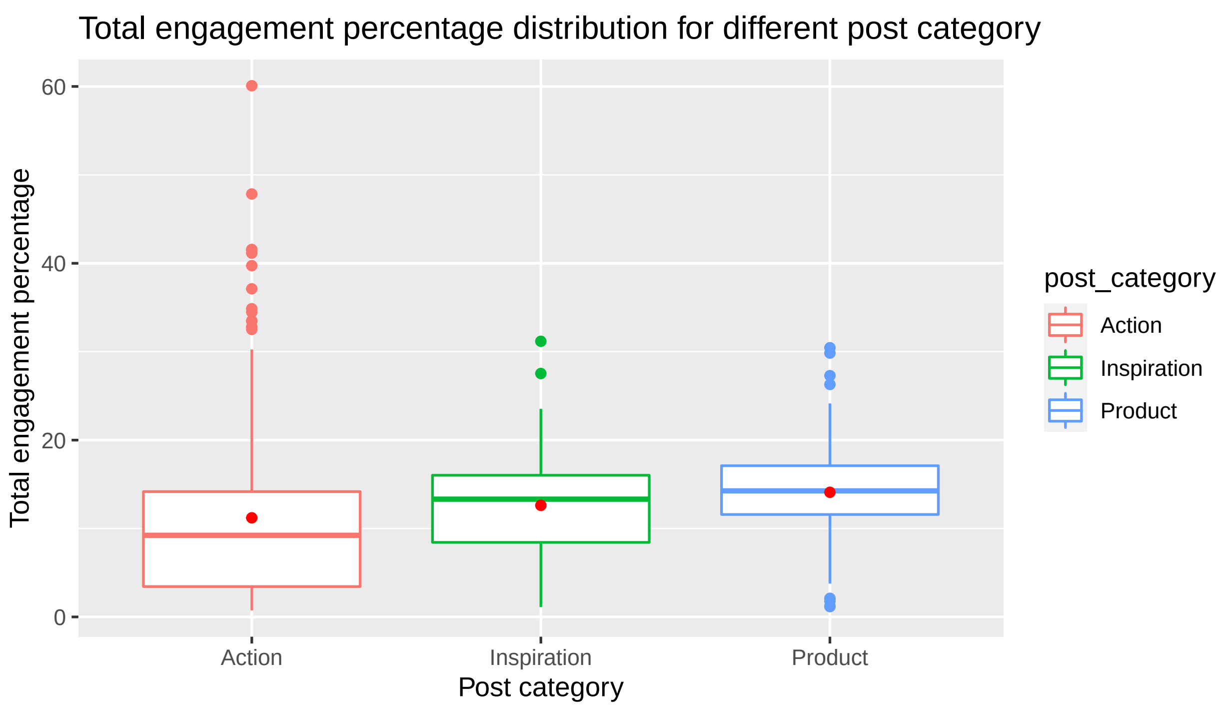
```
LR_post_category <- lm(total_engagement_percentage~post_category, facebook_data)
LR_post_category
```

```
##
## Call:
## lm(formula = total_engagement_percentage ~ post_category, data = facebook_data)
##
## Coefficients:
##               (Intercept)  post_categoryInspiration      post_categoryProduct
##                    11.211                     1.409                     2.887
```

```
# YOUR CODE HERE
```

Now we use **a box plot** to visualize our data for each `post_category` on the x-axis versus `total_engagement_percentage` on the y-axis. Moreover, add the corresponding response mean value by `post_category` using the adequate `ggplot2` function. Assign our plot to an object called `post_cat_plot`.

```
post_cat_plot <- ggplot(facebook_data, aes(y=total_engagement_percentage, x= post_category, group= post_
  labs(x="Post category", y = "Total engagement percentage", title= "Total engagement percentage distril


post_cat_plot
```



When dealing with a categorical variable,there is a baseline level. `R` puts by default all levels in alphabetical order. We can check what level is the baseline by using the function `levels()` as follows:

```
levels(as.factor(facebook_data$post_category))
```

```
## [1] "Action"      "Inspiration" "Product"
```

The level on the left-hand side is the baseline. For `post_category`, the baseline level is `Action`. Hence, our subsequent hypothesis testings will compare this level versus the other two: `Inspiration` and `Product`.

This is the summary of `LR_post_category`:

```
tidy(LR_post_category)
```

```
## # A tibble: 3 x 5
##   term                      estimate std.error statistic  p.value
##   <chr>                        <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)                  11.2      0.542     20.7  9.55e-69
## 2 post_categoryInspiration      1.41     0.827      1.70 8.91e- 2
## 3 post_categoryProduct          2.89     0.871      3.31 9.85e- 4
```

Let $\beta_{\texttt{Inspiration}}$ be the comparison between the level `Inspiration` in `post_category` and the baseline `Action` on the response `total_engagement_percentage`. Is the mean of the group `Inspiration` significantly different from that of `Action` at the $\alpha = 0.05$ significance level?

H0 <- The mean Total engagement percentage of the group Product is same as the mean of the group Action i.e. $\beta_{Inspiration} = 0$

Ha <- The mean Total engagement percentage of the group Product is not same as the mean of the group Action i.e. $\beta_{Inspiration} \neq 0$

Here the p-value here is 8.906192e-02 i.e. greater than the alpha i.e. the significance level so we fail to reject the null hypothesis

Let $\beta_{\texttt{Product}}$ be the comparison between the level `Product` in `post_category` and the baseline `Action` on the response `total_engagement_percentage`. Is the mean of the group `Product` significantly different from that of `Action` at the $\alpha = 0.05$ significance level?

H_0 <- The mean Total engagement percentage of the group Product is same as the mean of the group Action i.e. $\beta_{Product} = 0$

H_a<- The mean Total engagement percentage of the group Product is not same as the mean of the group Action i.e. $\beta_{Product} \neq 0$

Here the p-value here is 9.854578e-04 i.e. lesser than the alpha i.e. the significance level so we reject the null hypothesis

Additive and Interaction Multiple Linear Regression (MLR) Models

Here we use a subset of of 100 observations from the Facebook dataset (`facebook_sampling_data`). We will use this data to explore the difference between additive MLR models and MLR models with interaction terms. We will consider `total_engagement_percentage` as a response along with `page_engagement_percentage` and `post_category` as explanatory variables.

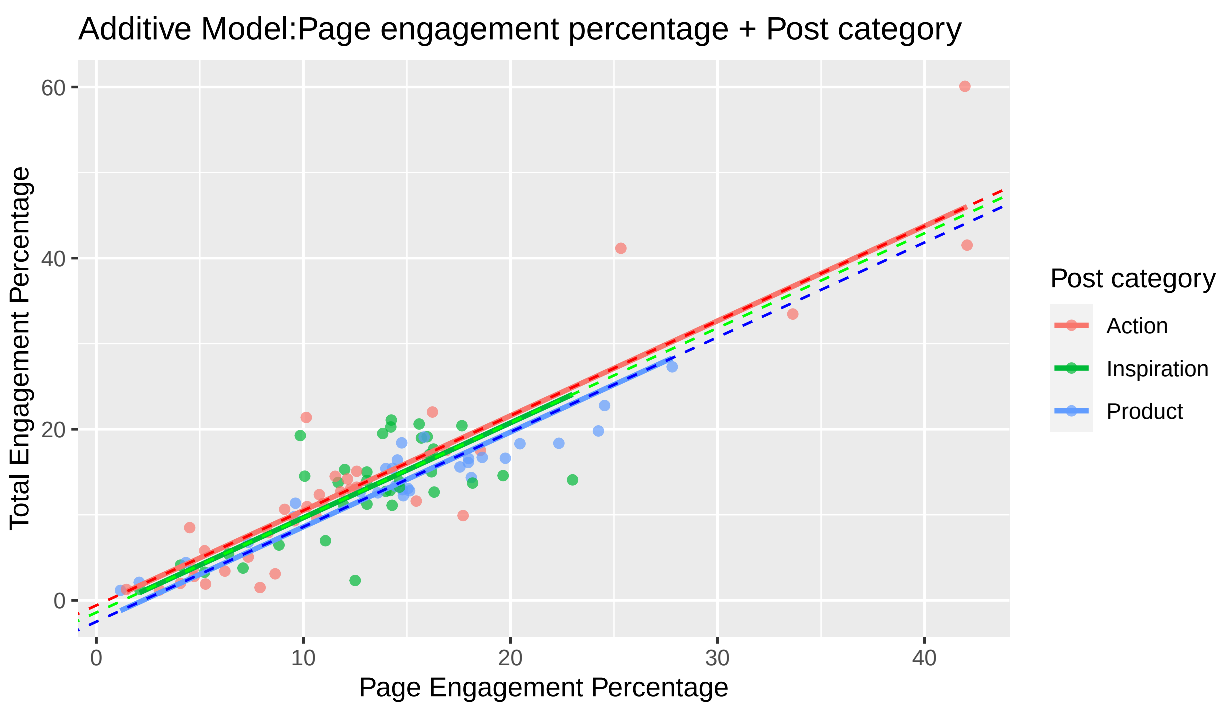
The additive MLR model called `MLR_add_ex2` and a MLR model with interaction effects called `MLR_int_ex2` that determines how `page_engagement_percentage` and `post.category` are associated with `total_engagement_percentage`.

```
MLR_add_ex2 <- lm(total_engagement_percentage~page_engagement_percentage + post_category, data = facebo

MLR_int_ex2 <- lm(total_engagement_percentage~page_engagement_percentage * post_category, data = facebo
```

We store the in-sample predictions from `MLR_add_ex2` in a new column within `facebook_sampling_data` called `pred_MLR_add_ex2`.

Then, using `facebook_sampling_data`, we create a scatterplot of the observed `page_engagement_percentage` on the x-axis versus `total_engagement_percentage` on the y-axis. Moreover, our plot should has three regression lines, one for each `post_category`, according to our in-sample predictions in `pred_MLR_add_ex2`. We colour the points and regression lines by `post_category` and assign your plot to an object called `add_pred_by_category`.

```
predictions_add <- predict(MLR_add_ex2, facebook_sampling_data[,c(2,5)])

facebook_sampling_data$pred_MLR_add_ex2 <- predictions_add

add_pred_by_category <- ggplot(facebook_sampling_data, aes(x= page_engagement_percentage, y = total_enga
  geom_abline(intercept=-0.5835111,slope=1.1082635, col = "red",linetype = "dashed")+
  geom_abline(intercept=-0.5835111+(-0.8329379),slope=1.1082635, col = "green",linetype = "dashed")+
  geom_abline(intercept=-0.5835111+(-1.9117622),slope=1.1082635, col = "blue",linetype = "dashed")

add_pred_by_category
```

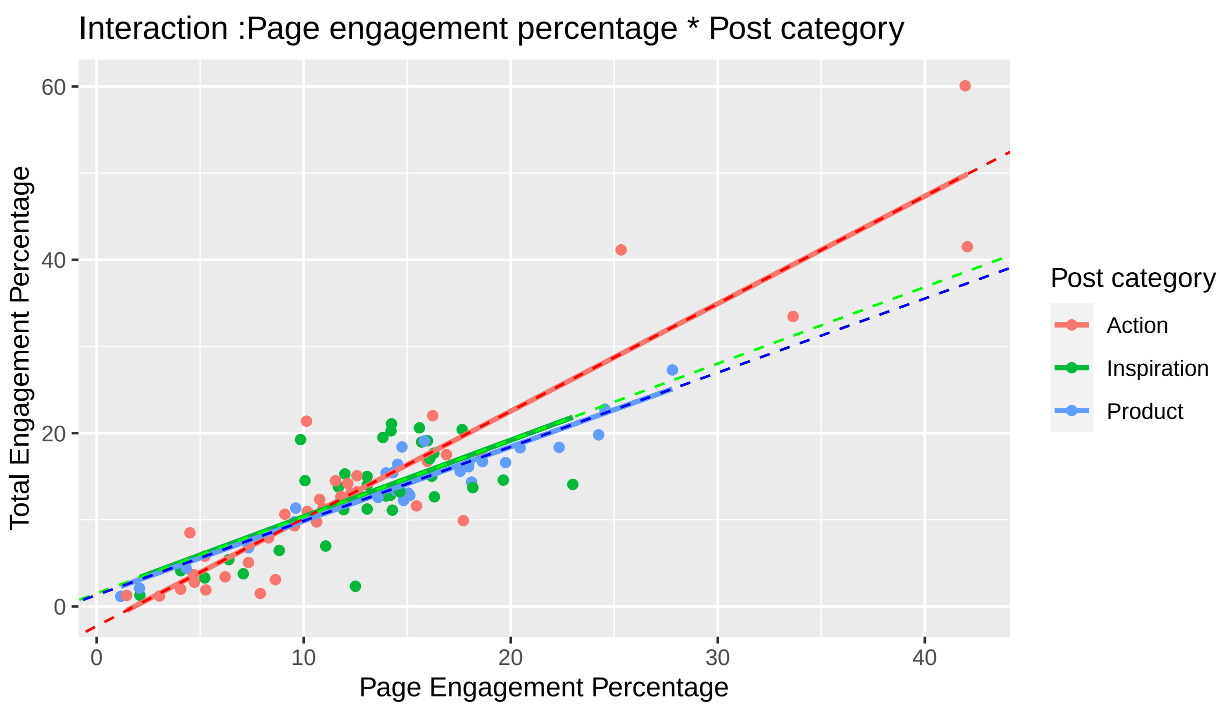


We assume that the relation between the Total engagement percentage and the Page engagement percentage is the same for all Post category.

Store the in-sample predictions from `MLR_int_ex2` in a new column within `facebook_sampling_data` called `pred_MLR_int_ex2`.

Then, using `facebook_sampling_data`, create a scatterplot of the observed `page_engagement_percentage` on the x-axis versus `total_engagement_percentage` on the y-axis. Moreover, your plot should have three regression lines, one for each `post_category`, according to your in-sample predictions in `pred_MLR_int_ex2`. Again, colour the points and regression lines by `post_category`. Include a human-readable legend indicating what colour corresponds to each `post_category`. Ensure that your x and y-axes are also human-readable along with a proper title. Assign your plot to an object called `int_pred_by_category`.

```
predictions_int <- predict(MLR_int_ex2, facebook_sampling_data[,c(2,5)])

df <- tidy(MLR_int_ex2)

facebook_sampling_data$pred_MLR_int_ex2 <- predictions_int
int_pred_by_category <- ggplot(facebook_sampling_data, aes(x= page_engagement_percentage, y = total_enga
    geom_abline(intercept=-2.2690159,slope=1.2403689, col = "red",linetype = "dashed")+
    geom_abline(intercept=-2.2690159+(3.7819843),slope=1.2403689+(-0.3566549), col = "green",linetype =
    geom_abline(intercept=-2.2690159+(3.5893921 ),slope=1.2403689+(-0.3850602), col = "blue", linetype =
    labs(title = 'Interaction :Page engagement percentage * Post category ', x = "Page Engagement Percen

int_pred_by_category
```



The estimated relationship between between `total_engagement_percentage` and `page_engagement_percentage` is not the same for all levels. The slope of the reference Post category i.e. Action measures the relationship between the `total_engagement_percentage` and `page_engagement_percentage`. For the level Inspiration `page_engagement_percentage:post_categoryInspiration` estimate gets added to the reference slope estimate because `page_engagement_percentage:post_categoryInspiration` estimate measures the difference in estimated slopes of Post category Action and Inspiration. For the level Product `page_engagement_percentage:post_categoryProduct` estimate gets added to the reference slope estimate because `page_engagement_percentage:post_categoryProduct` estimate measures the difference in estimated slopes of Post category Action and Product.

```
tidy(MLR_int_ex2) %>% mutate_if(is.numeric, round, 2)
```

```
## # A tibble: 6 x 5
```

```
##   term                                     estimate std.error statistic p.value
##   <chr>                                       <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)                                 -2.27      1.1      -2.06    0.04
## 2 page_engagement_percentage                   1.24      0.07     17.9     0
## 3 post_categoryInspiration                     3.78      2.38      1.59    0.12
## 4 post_categoryProduct                         3.59      2.22      1.61    0.11
## 5 page_engagement_percentage:post_category~   -0.36      0.17     -2.12    0.04
## 6 page_engagement_percentage:post_category~   -0.39      0.14     -2.8     0.01
```

The estimate of (Intercept) for `page_engagement_percentage`: is the estimated intercept of the line of the reference Product Category i.e. Action red line.

The estimate for `post_categoryProduct`: is the estimated difference between intercepts of the line of `post_categoryProduct` vs that of the reference `post_categoryAction`(blue vs red lines)

Goodness of Fit

$$\texttt{total\_engagement\_percentage}_i = \beta_0 + \beta_1 \times \texttt{page\_engagement\_percentage}_i + \varepsilon_i$$

We will now *quantify* the model's goodness of fit.

We estimate a model called `SLR_ex3` with the SLR above. Then use `broom::augment()` to calculate the predicted value and the residual for each observation (amongst other things) and add them to the `facebook_data` tibble.

```
SLR_ex3<- lm(total_engagement_percentage~page_engagement_percentage, facebook_data)
facebook_data <-augment(SLR_ex3)
facebook_data
```

```
## # A tibble: 491 x 8
##    total_engagement_p~ page_engagement_pe~ .fitted .resid     .hat .sigma .cooksd
##                  <dbl>               <dbl>   <dbl>  <dbl>    <dbl>  <dbl>   <dbl>
##  1                6.47                7.26    6.79 -0.326 0.00333   3.42 1.53e-5
##  2               13.9                18.1    18.0  -4.05  0.00330   3.41 2.34e-3
##  3                7.34                8.78    8.36 -1.03  0.00271   3.42 1.24e-4
##  4                4.41                4.32    3.78  0.632 0.00509   3.42 8.83e-5
##  5                9.26               12.4    12.1  -2.80  0.00204   3.41 6.89e-4
##  6               11.4                12.9    12.6  -1.27  0.00204   3.42 1.41e-4
##  7                4.11                4.06    3.51  0.605 0.00528   3.42 8.38e-5
##  8                3.91                3.82    3.26  0.658 0.00547   3.42 1.03e-4
##  9               12.9                15.8    15.6  -2.66  0.00245   3.41 7.49e-4
## 10                5.97                8.56    8.14 -2.17  0.00279   3.41 5.68e-4
## # ... with 481 more rows, and 1 more variable: .std.resid <dbl>
```

We can use `R-squared`, computed with the dataset of n observations, to compare the performance of our linear model with the null model (i.e., the model that simply predicts the mean observed value of `total_engagement_percentage`) using the formula:

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

where y_i is the observed response for the ith observation, $\hat{y}_i$ is the predicted value for the ith observation, and $\bar{y}$ is the sample mean of the n observed responses.

Using the right columns of `facebook_data`,we calculate R^2 manually using the equation above. Bind our results to the **numeric vector-type** variable `R_squared_SLR_ex3`.

```
R_squared_SLR_ex3 <- 1-(sum((facebook_data$total_engagement_percentage - facebook_data$.fitted)^2)/sum(

R_squared_SLR_ex3
```

```
## [1] 0.8113834
```

Yes the SLR fits the data better than the a null model. Since the R^2 is close to 1 i.e. 0.8113834 and is positive this indicates that the SLR fits better than the null model. The R^2 is the increase in predicting the response using the SLR rather than the null model, i.e. the sample mean, in terms of total response variation.

`broom::glance()` provides key statistics for interpreting model's goodness of fit. We use `broom::glance()` to verify our result and bind our results to the variable `key_stats_ex3`.

```
key_stats_ex3 <- glance(SLR_ex3)

key_stats_ex3
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic   p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.811         0.811  3.41     2104. 3.03e-179     1 -1298. 2603. 2615.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Nested Models

Typically we want to build a model that is a good fit for our data. However, if we make a model that is too complex, we risk overfitting. How do we decide whether a more complex model contributes additional useful information about the association between the response and the explanatory variables or whether it is just overfitting? One method is to compare and test nested models. Two models are called "nested" if both models contain the same terms, and one has at least one additional term, e.g.:

Model 1:

$$Y_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + \varepsilon_i$$

Model 2:

$$Y_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + \beta_3 X_{3,i} + \varepsilon_i$$

In the above example we would say that Model 1 is nested within Model 2.

We build the following two models using the `facebook_data`:

`SLR_ex4`:

$$\texttt{total\_engagement\_percentage}_i = \beta_0 + \beta_1 \times \texttt{page\_engagement \_percentage}_i + \varepsilon_i$$

`MLR_ex4`:

$$\texttt{total\_engagement\_percentage}_i = \beta_0 + \beta_1 \times \texttt{page\_engagement\_percentage}_i + \beta_2 \times \texttt{comment\_percentage}_i + \varepsilon_i$$

```r
facebook_data <- read_csv("data/facebook_data.csv")
```

```
## Rows: 491 Columns: 5

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): post_category
## dbl (4): total_engagement_percentage, page_engagement_percentage, share_perc...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
SLR_ex4 <- lm(total_engagement_percentage~page_engagement_percentage, facebook_data)

MLR_ex4 <- lm(total_engagement_percentage~page_engagement_percentage + comment_percentage, facebook_data
```

The F-statistic is similar to R^2 in that it measures goodness of fit. We use `broom::glance()` to observe the F-statistics and the corresponding p-values for each model `SLR_ex4` and `MLR_ex4` created above. Store our results in `key_stats_SLR_ex4` and `key_stats_MLR_ex4`.

```r
key_stats_SLR_ex4 <- glance(SLR_ex4)
key_stats_MLR_ex4 <- glance(MLR_ex4)

key_stats_SLR_ex4
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic   p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.811         0.811  3.41     2104. 3.03e-179     1 -1298. 2603. 2615.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```r
key_stats_MLR_ex4
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic   p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.812         0.811  3.41     1051. 1.48e-177     2 -1298. 2604. 2621.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

- At a significance level of $\alpha = 0.05$

Yes our models perform better than the null model. The R^2 for the SLR model is closer to 1 and is positive so the model fits better than the null model.Also the p-value is lesser than the significance level so we are able to reject the null hypothesis i.e. R^2 =0. The R^2 for the MLR model is closer to 1 and is positive so the model fits better than the null model.Also the p-value is lesser than the significance level so we are able to reject the null hypothesis i.e. R^2 =0.

To compare both models, we can use the `anova()` function to perform an appropriate F-test. Perform an F-test to compare `SLR_ex4` with `MLR_ex4` and bind our results to the object `F_test_ex4`.

```r
F_test_ex4 <- anova(SLR_ex4,MLR_ex4)

df <-as.data.frame(F_test_ex4)
df
```

```
##   Res.Df      RSS Df Sum of Sq         F    Pr(>F)
## 1    489 5693.240 NA        NA        NA        NA
## 2    488 5689.374  1  3.866091 0.3316099 0.5649781
```

Since the p-value is greater than the significance level we fail to reject the null hypothesis.And there is no evidence that the MLR model fits the data better than the SLR model.And no evidence to claim that addition of comment_percentage to the model improves the model.