Linear Regression Model- Facebook Data

Contents

1. Setup

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

2. 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 project, 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.

3. Load Data

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

4. Inference in a Simple Linear Regression (SLR) Model

The equaion should look like:

$$\boldsymbol{total\_engagement\_percentage_i} = \beta_0 + \beta_1 \times \boldsymbol{page\_engagement\_percentage_i} + \epsilon_i$$

Note that in this sample's regression equation, the random component for the ith observation ϵ_i needs to be included. Moreover, $\boldsymbol{total\_engagement\_percentage_i}$ and $\boldsymbol{page\_engagement\_percentage_i}$ are the response and explanatory variables for the ith observation, and β_0 and β_1 are the intercept and slope coefficients respectively. The value of i ranges from 1 to 491 for this dataset.

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, data = 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
```

Use `tidy()` to obtain the estimated coefficients of `SL_reg`, their associated standard errors, and the p-values associated with the t-test.

```
tidy_SL_reg <- tidy(SL_reg) |> mutate_if(is.numeric, round, 3)
tidy_SL_reg
```

```
## # A tibble: 2 x 5
##   term                       estimate std.error statistic p.value
##   <chr>                         <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)                  -0.671     0.324     -2.07   0.039
## 2 page_engagement_percentage    1.03      0.022     45.9    0
```

Question coming out: Is the slope coefficient relating `total_engagement_percentage` and `page_engagement_percentage` significantly different from 0?

We can use null (H_0) and alternative (H_a) hypotheses of the slope coefficient to test this. Then, using the output from `tidy()` above, determine whether we should reject H_0 or not with a significance level $\alpha = 0.05$.

$$H_0 : \beta_1 = 0, i.e., \textit{ the slope is equal to } 0$$
$$H_a : \beta_1 \neq 0, i.e., \textit{ the slope is not equal to } 0$$

Our sample gives us statistical evidence to reject H_0 with a p-value < 0.001, which is smaller than the significance level $\alpha = 0.05$

5. 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`.

$$\boldsymbol{TEP_i} = \beta_0 + \beta_1 \times \boldsymbol{PEP_i} + \beta_2 \times \boldsymbol{SP_i} + \beta_3 \times \boldsymbol{CP_i} + \epsilon_i$$

$TEP_i = total\_engagement\_percentage\_i$

$PEP_i = page\_engagement\_percentage_i$

$SP_i = share\_percentage\_i$

$CP_i = comment\_percentage\_i$

Note that in this sample's regression equation, the random component for the ith observation ϵ_i needs to be included. Moreover, $\boldsymbol{total\_engagement\_percentage\_i}$ is the response whereas $\boldsymbol{page\_engagement\_percentage_i}$, $\boldsymbol{share\_percentage\_i}$, and $\boldsymbol{comment\_percentage\_i}$ are the explanatory variables for the ith observation. The parameter ϵ_0 is the intercept with ϵ_1, ϵ_2, and ϵ_3 as the regression coefficients. The value of i ranges from 1 to 491 for this dataset.

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,
             data = facebook_data)
ML_reg
```

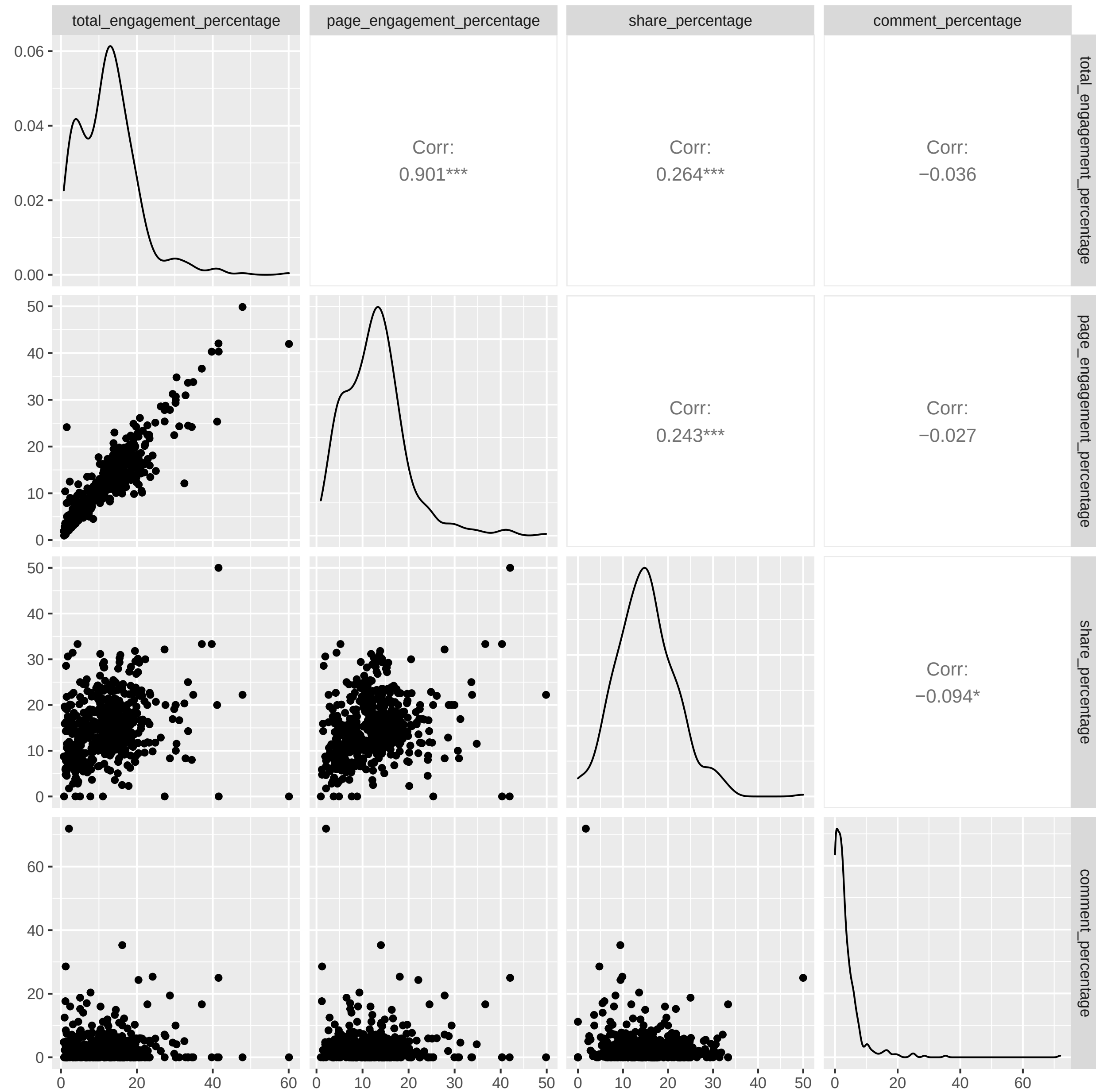
```
##
## Call:
## lm(formula = total_engagement_percentage ~ page_engagement_percentage +
##     share_percentage + comment_percentage, data = facebook_data)
##
## Coefficients:
##                (Intercept)  page_engagement_percentage
##                   -1.29650                     1.01558
##   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

```
facebook_data[, 1:4] %>%
  ggpairs(progress = FALSE)
```



The variables `total_engagement_percentage` and `page_engagement_percentage` have the highest correlation of 0.901.

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) |> mutate_if(is.numeric, round, 3)

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   0.004
## 2 page_engagement_percentage      1.02     0.023     44.1    0
```

```
## 3 share_percentage               0.055      0.024      2.31     0.021
## 4 comment_percentage            -0.011      0.03      -0.368    0.713
```

According to the output, the regression coefficients associated to `page_engagement_percentage` and `share_percentage` are statistically significant with p values < 0.05. The regression coefficient associated to `comment_percentage` is not significant in this model.

predictions

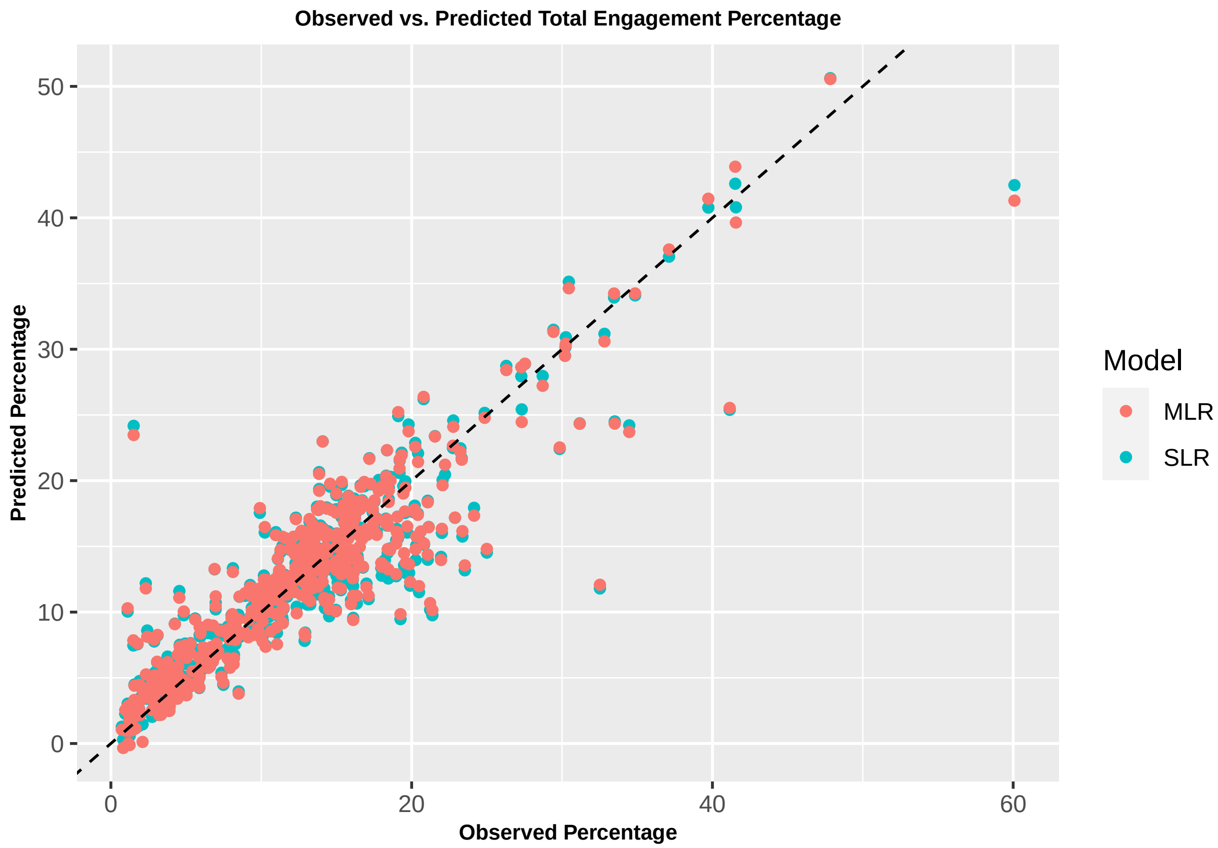
Now we use both the `SL_reg` and `ML_reg` to make predictions of their response. Plot the **in-sample** *predicted values* on the y-axis versus the *observed values* on the x-axis of the response (using `geom_point()`) to check the goodness of fit of the two models **visually**. We can assess this by 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`.

```
predicted_response <- facebook_data %>%
  mutate(
    SLR = predict(SL_reg),
    MLR = predict(ML_reg),
    ) %>%
  select(total_engagement_percentage, SLR, MLR) %>%
  gather(Model, predicted_percentage, SLR, MLR)

prediction_plot <- ggplot(predicted_response, aes(
  x = total_engagement_percentage,
  y = predicted_percentage,
  color = Model
)) +
  geom_point() +
  labs(
    title = "Observed vs. Predicted Total Engagement Percentage",
    x = "Observed Percentage",
    y = "Predicted Percentage"
    ) +
  theme(
    plot.title = element_text(
      face = "bold",
      size = 8,
      hjust = 0.5
    ),
    axis.title = element_text(face = "bold", size = 8)
  ) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed")

prediction_plot
```



There is little difference between the SLR and MLR models. The models produce reasonable predictions at low total engagement percentages. Based on the scatterplot's visual evidence, we can conclude that the incorporation of `share_percentage` and `comment_percentage` does not improve the in-sample prediction accuracy in this dataset.

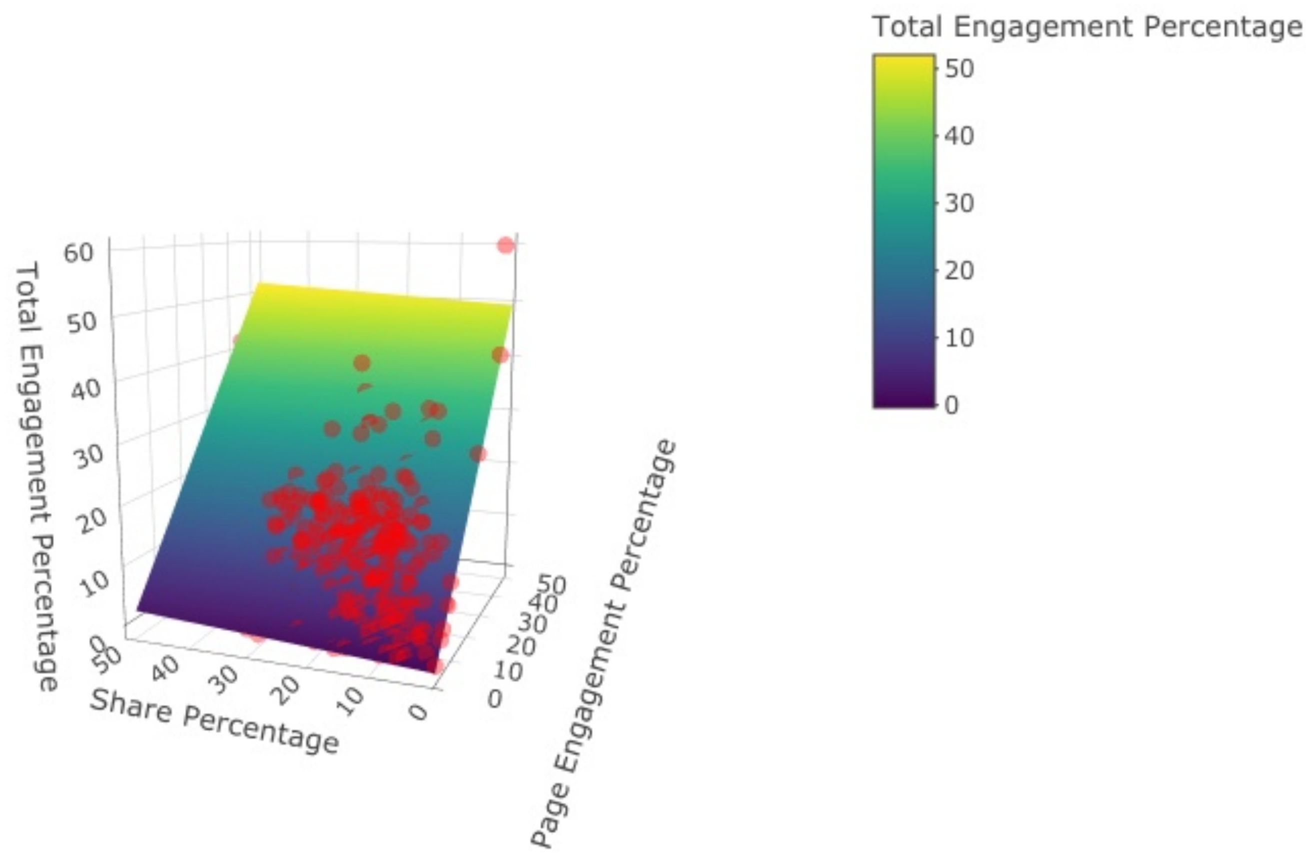
It is important to remember that linear regression is not confined to a 2-dimensional space and "lines". In this question we are working with 4-dimensional data (3 inputs and 1 response). While visualizing 4 dimensions is tricky, we can comfortably visualize 3 dimensions. Hence, let us fit a MLR model with `total_engagement_percentage`, `page_engagement_percentage`, and `share_percentage`. One of these variables has to be considered as the response (following the modelling setup of the previous sections).

In 3-dimensions, we have a regression plane, not a line. We are using `plotly` package to do it.

```
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```r
# Create the regression model.

model_3d <- lm(total_engagement_percentage ~ page_engagement_percentage +
   share_percentage,
data = facebook_data
)

# Get coordinates of points for 3D scatterplot.

x_values <- facebook_data$page_engagement_percentage
y_values <- facebook_data$share_percentage
z_values <- facebook_data$total_engagement_percentage

# Define regression plane

# Construct x and y grid elements

page_engagement_percentage <- seq(from = min(x_values), to = max(x_values), length = 50)
share_percentage <- seq(from = min(y_values), to = max(y_values), length = 50)

# Construct z grid by computing predictions for
# all age_engagement_percentage/share_percentage pairs

fitted_values <- crossing(page_engagement_percentage, share_percentage) %>%
  mutate(total_engagement_percentage = predict(model_3d, .))

# Transform grid predictions to a matrix

z_grid <- fitted_values %>%
  pull(total_engagement_percentage) %>%
  matrix(nrow = length(share_percentage))
x_grid <- page_engagement_percentage
y_grid <- share_percentage

# Plot using plotly

facebook_data %>%
  plot_ly() %>%
  add_markers(
    x = ~ as.numeric(page_engagement_percentage),
    y = ~ as.numeric(share_percentage),
    z = ~ as.numeric(total_engagement_percentage),
    marker = list(size = 5, opacity = 0.4, color = "red")
  ) %>%
  layout(scene = list(
    xaxis = list(title = "Page Engagement Percentage"),
    yaxis = list(title = "Share Percentage"),
    zaxis = list(title = "Total Engagement Percentage")
  )) %>%
  add_surface(
    x = ~x_grid, y = ~y_grid, z = ~z_grid,
    colorbar = list(title = "Total Engagement Percentage")
  )
```



Total Engagement Percentage
50
40
30
20
10
0
60
Total Engagement Percentage
Share Percentage
Page Engagement Percentage

6. 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`.

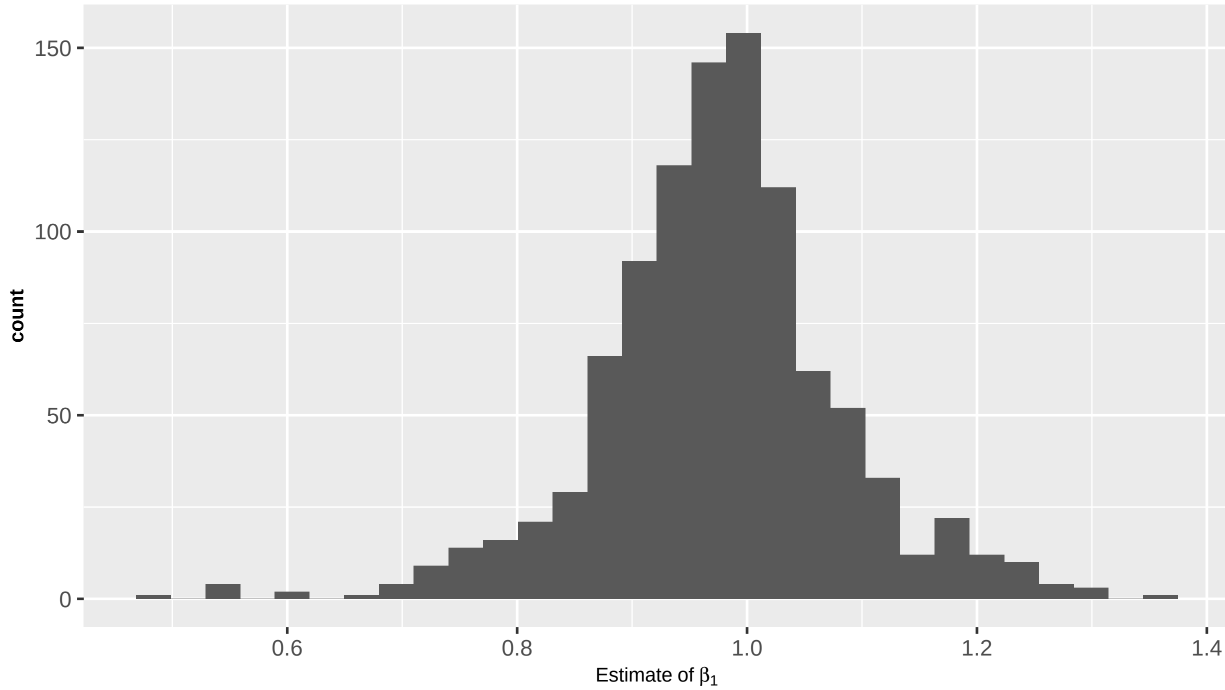
```
set.seed(1234)
facebook_base_sample <- sample_n(facebook_data, 50)
```

```
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 first draws N bootstrap samples from facebook_base_sample (the same size as the sample facebook_base_sample). It then creates a linear model using lm() for each sample and extracts each model's coefficients. After some wrangling, we then pull out the $\hat{\beta}_1$ coefficient for each bootstrap sample and plot the resulting distribution using a histogram. The bootstrap sampling distribution does not entirely resemble a normal distribution and is slightly right-skewed - which is likely due to our small sample size of n = 50 observations.

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, data = facebook_base_sample)
tidy_SL_reg_n50 <- tidy(SL_reg_n50)

boot_mean
```

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

```
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 two estimates, **boot_mean** and the one found **SL_reg_n50**, are practically equal. The center of the sampling distribution is close to the estimated slope using the original sample.

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`.

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


boot_ci
```

```
## [1] 0.7545946 1.2058666
```

```
tidy_SL_reg_n50
```

```
## # A tibble: 2 x 7
##   term                       estimate std.error stati~1  p.value conf.~2 conf.~3
##   <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_percentage   0.976     0.0764 12.8    4.65e-17   0.822    1.13
## # ... with abbreviated variable names 1: statistic, 2: conf.low, 3: conf.high
```

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

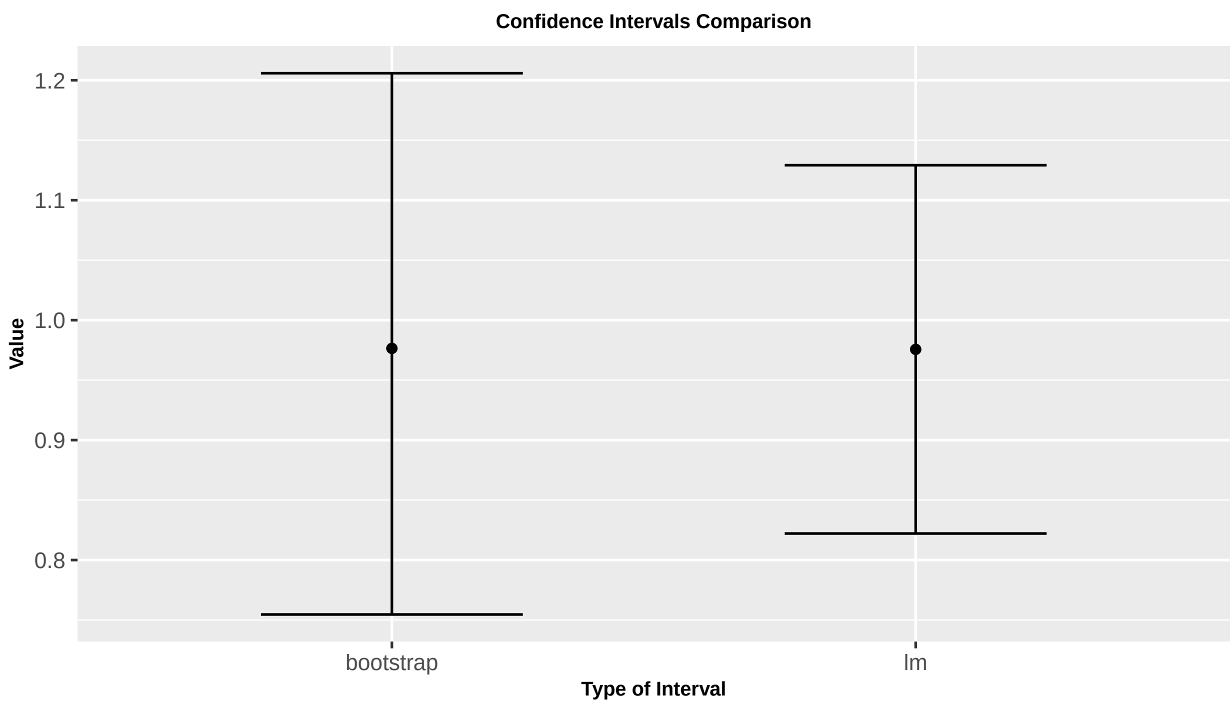
- `type`, the label `"bootstrap"` for the CI in `boot_ci` and `"lm"` for the one in `tidy_SL_reg_n50`.
- `mean`, the center of each CI (`boot_mean` for the bootrapping CI and the `estimate` found `tidy_SL_reg_n50`).
- `conf.low`, the respective lower bound in `boot_ci` and `tidy_SL_reg_n50`.
- `conf.high`,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

```
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_SL_reg_n50$conf.high[2])
)

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`.

```r
ci_plot <- ggplot(ci_data, aes(x = type, y = mean)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high),
    width = 0.5
) +
  theme(
    plot.title = element_text(face = "bold", size = 8, hjust = 0.5),
    axis.title = element_text(face = "bold", size = 8)
  ) +
  labs(
    title = "Confidence Intervals Comparison", x = "Type of Interval", y = "Value"
  )

ci_plot
```



7. 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.

> There will be three parameters. One is the intercept, and then two more parameters are used to encode the three levels of the categorical variable post_category. Recall that R uses k-1 encoded parameters to describe a categorical variable with k levels.

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

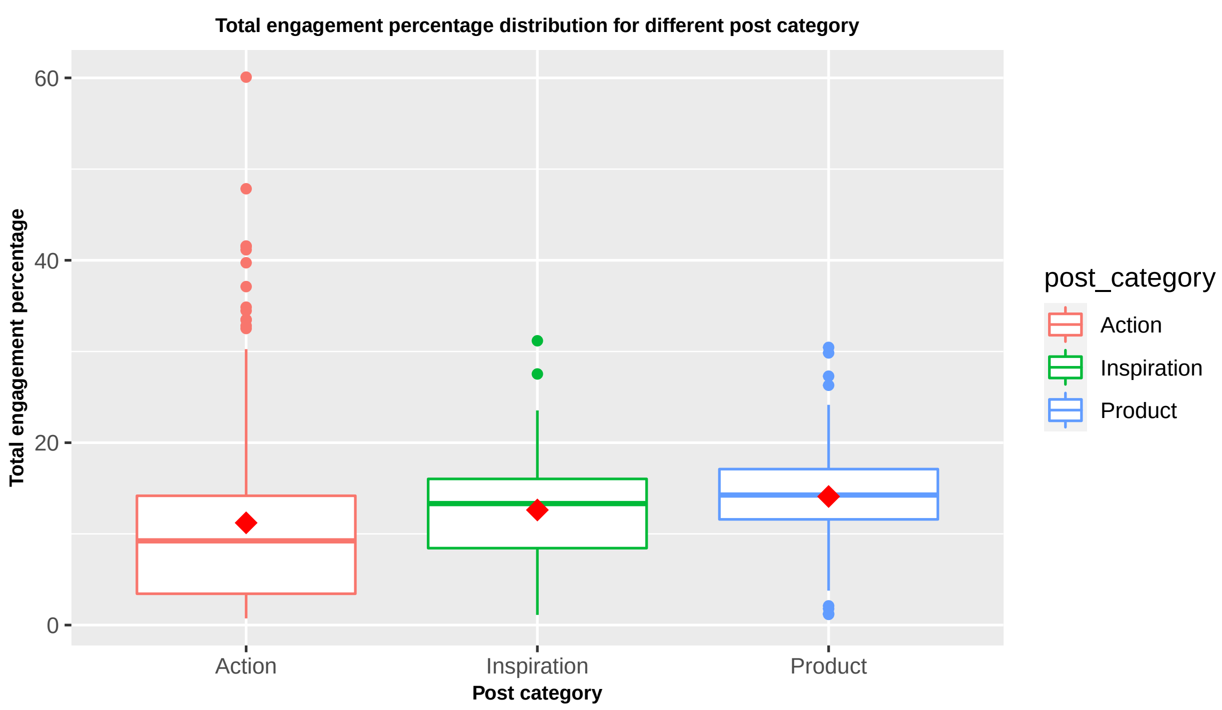
```r
LR_post_category <- lm(total_engagement_percentage ~ post_category, data = facebook_data)
LR_post_category
```

```
##
```

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

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`.

```r
post_cat_plot <- ggplot(
  facebook_data,
  aes(x = post_category,
      y = total_engagement_percentage,
      group = post_category,
      color = post_category)
  ) +
  geom_boxplot() +
  stat_summary(fun = mean,color = "red", geom = "point", shape = 18, size = 4) +
  labs(x ="Post category", y = "Total engagement percentage",
       title= "Total engagement percentage distribution for different post category"
  ) +
  theme(
    plot.title = element_text(face = "bold", size = 8, hjust = 0.5),
    axis.title = element_text(face = "bold", size = 8)
  )

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?

For level **Inspiration**

$$H_0 : \text{The mean of the group 'Inspiration' is same as the mean of the group 'Action'},$$

$$H_a : \text{The mean of the group 'Inspiration' is not same as the mean of the group 'Action'}.$$

Our sample gives us statistical evidence to fail to reject H_0 with a p-value > 0.05, which is larger than the significance level $\alpha = 0.05$. Thus, we do not have enough evidence that the group means of **Inspiration** and **Action** are different in terms of **total__engagement__percentage**.

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?

For level **Product**

$$H_0 : \text{The mean of the group 'Product' is same as the mean of the group 'Action'},$$

$$H_a : \text{The mean of the group 'Product' is not same as the mean of the group 'Action'}.$$

Our sample gives us statistical evidence to reject H_0 with a p - value < 0.001, which is smaller than the significance level $\alpha = 0.05$. Thus, we have enough evidence that the group means of Product and Action are different

8. 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.

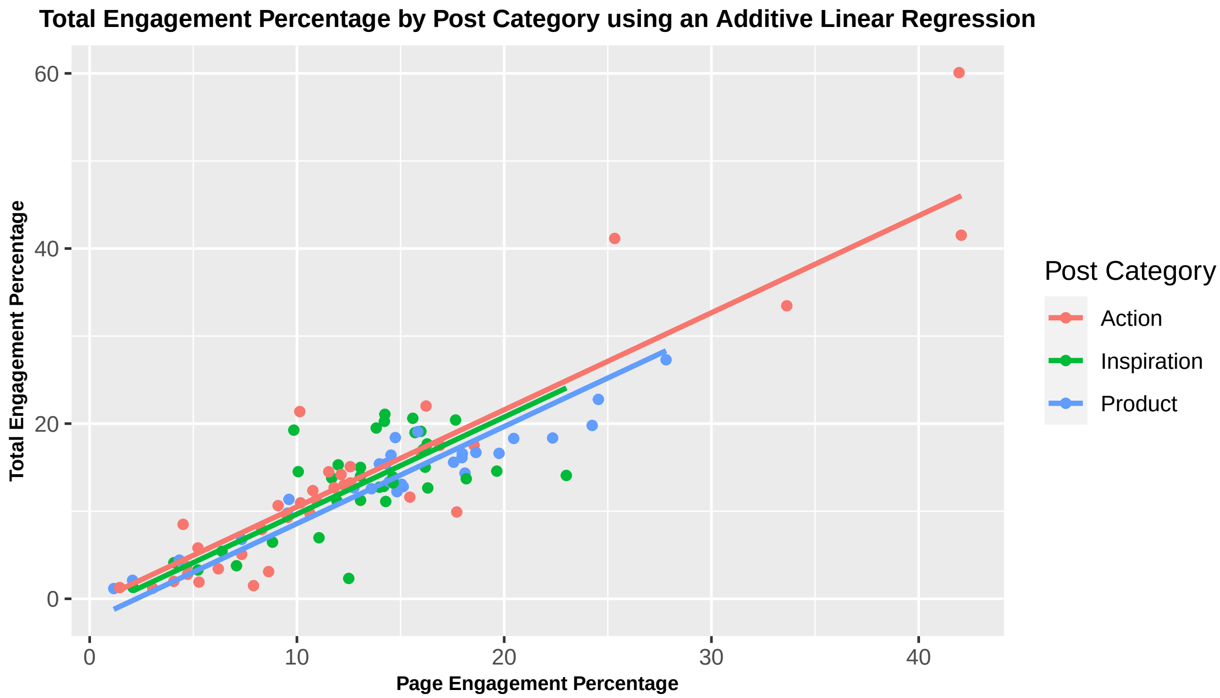
The additive MLR model called `MLR_add_ex2` and a MLR model with interaction effects called `MLR_int_ex2` that determines how `page_engagement_percentage` and `post.category` are associated with `total_engagement_percentage`.

```r
MLR_add_ex2 <- lm(total_engagement_percentage~page_engagement_percentage + post_category,
                  data = facebook_sampling_data)

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

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`.

```r
facebook_sampling_data$pred_MLR_add_ex2 <- predict(MLR_add_ex2)

add_pred_by_category <- ggplot(facebook_sampling_data, aes(
  x = page_engagement_percentage,
  y = total_engagement_percentage,
  color = post_category
)) +
  geom_point() +
  geom_line(aes(y = pred_MLR_add_ex2), size = 1) +
  labs(
    title = "Total Engagement Percentage by Post Category using an Additive Linear Regression",
    x = "Page Engagement Percentage",
    y = "Total Engagement Percentage"
  ) +
  theme(
    plot.title = element_text(face = "bold", size = 10, hjust = 0.5),
    axis.title = element_text(face = "bold", size = 8) ) +
  labs(color = "Post Category")

add_pred_by_category
```



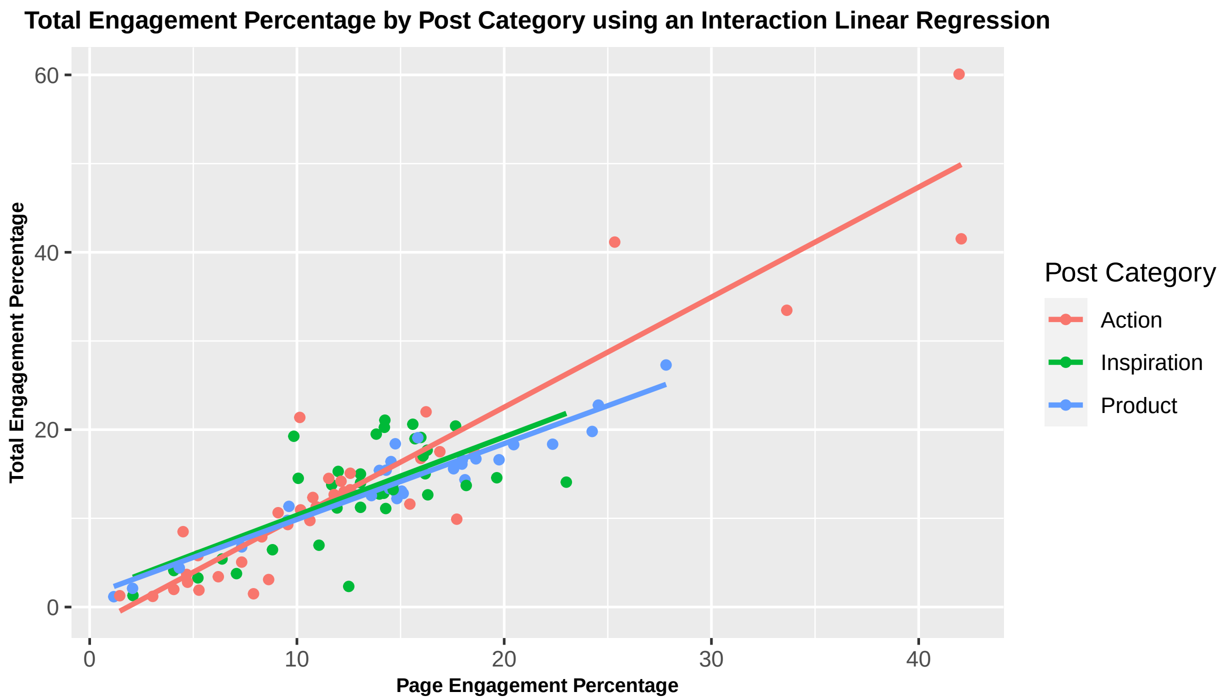
> The key assumption we are making in our additive MLR models is that all treatment groups in `post_category` have the same slope relating `page_engagement_percentage` to `total_engagement_percentage`.

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`.

```r
facebook_sampling_data$pred_MLR_int_ex2 <- predict(MLR_int_ex2)

int_pred_by_category <- ggplot(facebook_sampling_data, aes(
  x = page_engagement_percentage,
  y = total_engagement_percentage,
  color = post_category
)) +
  geom_point() +
  geom_line(aes(y = pred_MLR_int_ex2), size = 1) +
  labs(
    title = "Total Engagement Percentage by Post Category using an Interaction Linear Regression",
    x = "Page Engagement Percentage",
    y = "Total Engagement Percentage"
  ) +
  theme(
    plot.title = element_text(face = "bold", size = 10, hjust = 0.5),
    axis.title = element_text(face = "bold", size = 8) ) +
  labs(color = "Post Category")
```

```
int_pred_by_category
```



In this MLR model, the estimated relationship between total_engagement_percentage and page_engagement_percentage is different from each level of post_category as lines have different slopes. We note that switching from the baseline level Action to Inspiration or Product decreases the regression's slope.

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

```
## # A tibble: 6 x 5
##   term                                         estim~1 std.e~2 stati~3 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_categoryInspi~ -0.36    0.17   -2.12    0.04
## 6 page_engagement_percentage:post_categoryProdu~ -0.39    0.14   -2.8     0.01
## # ... with abbreviated variable names 1: estimate, 2: std.error, 3: statistic
```

The estimate of the coefficient is -0.39. This can be interpreted as the difference in the slope of product compared to the slope of action, i.e., the slope of action for one unit increase/decrease in page_engagement_percentage is 1.24, and the slope of product is 1.24 + (-0.39) = 0.85.

9. Goodness of Fit

We estimated a SLR model with the folling sample's regression equation using the facebook_data:

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

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,
             data = facebook_data
             )
facebook_data <- augment(SLR_ex3, facebook_data)
```

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`.

```
# Obtaining the Residual Sum of Squares (RSS)

RSS_SLR_ex3 <- sum(facebook_data$.resid^2)

# Obtaining the Total Sum of Squares (TSS)

TSS_SLR_ex3 <- sum((facebook_data$total_engagement_percentage -
                      mean(facebook_data$total_engagement_percentage))^2)

# Computing R-squared

R_squared_SLR_ex3 <- 1 - RSS_SLR_ex3/TSS_SLR_ex3

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.squ~1 sigma stati~2   p.value    df logLik   AIC   BIC devia~3
##       <dbl>       <dbl> <dbl>   <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1     0.811       0.811  3.41   2104. 3.03e-179     1 -1298. 2603. 2615.   5693.
## # ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
## #   variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

10. 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`:

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

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

MLR_ex4 <- lm(total_engagement_percentage ~ page_engagement_percentage + comment_percentage,
              data = 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`.

```
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.squ~1 sigma stati~2   p.value    df logLik   AIC   BIC devia~3
##       <dbl>       <dbl> <dbl>   <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1     0.811       0.811  3.41   2104. 3.03e-179     1 -1298. 2603. 2615.   5693.
## # ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
## #   variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

```
key_stats_MLR_ex4
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squ~1 sigma stati~2   p.value    df logLik   AIC   BIC devia~3
##       <dbl>       <dbl> <dbl>   <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1     0.812       0.811  3.41   1051. 1.48e-177     2 -1298. 2604. 2621.   5689.
## # ... with 2 more variables: df.residual <int>, nobs <int>, and abbreviated
## #   variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

The p-values associated with the F-statistic of **SLR_ex4** and **MLR_ex4** are smaller than α = 0.05, indicating that the fit of each model is significantly better than the null model.

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`.

```
F_test_ex4 <- as.data.frame(anova(SLR_ex4, MLR_ex4))

F_test_ex4
```

```
##   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
```

Results from the ANOVA table show p-value of 0.565. We would therefore reject that **MLR_ex4** (with the additional explanatory variable comment_percentage) significantly improves the fit to the data, and would stick with **SLR_ex4**. More specifically, in this test we are testing if β_2 (the additional coefficient in MLR_ex4) is zero - and we do not reject that null hypothesis.