Part 1:

Typical Linear Regression Equation -> $E(Y|x) = \beta_0 + \beta_1 x$

The above equation generally conforms to the form of a straight-line eqn. -> y = MX + c. In the above equation:

β_0 = Y-intercept of the line.

β_1 = Coefficient of the independent variable, x. This can also be thought of as the slope of the best-fit regression line.

Y – Dependent variable which needs to be predicted. Denotation E(Y|x) translates to the expected value of Y given x.

a.) **Loss Function** -> This function measures the discrepancy between the observed value and the expected value of Y.

Mathematically -> $L(y, \hat{y}) = (y - \hat{y})^2$

Where, y (hat) = observed value

y = Expected value

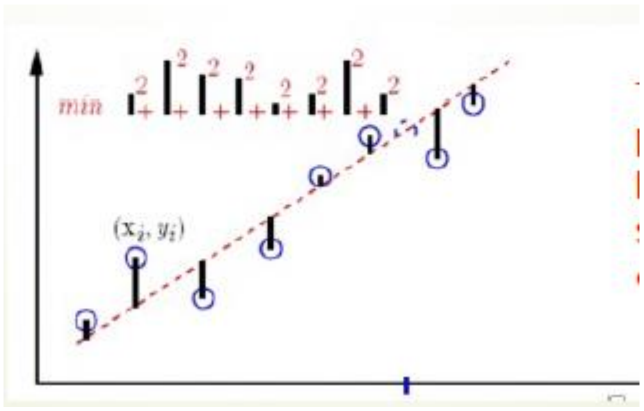
The metric is a scalar quantity and hence is squared to measure the magnitude only.

b.) **Ordinary Least Squares(OLS) Regression** -> This is used to determine the predicted values which will stay closest to the original best fit line of the linear regression. It is done in three steps ->

i) Compute the residual i.e. the loss between the expected and measured values of the dependent variable.

ii) Compute the residual sum of squared loss (RSS) to get an overall picture of the residual spread.

iii) Find the values of the predicted coefficients and dependent variable which give the minimum RSS.



In the above image, the red line depicts the line of best fit with the minimum RSS.

c.) The numerical metrics/measures which depict the accuracy and proper fit of linear regression are:

i) **Residual Standard Error (RSE)**: This is an estimate of the standard deviation of the error term. This simply indicates how much the observations will deviate from the true regression line. Mathematically,

$$\text{RSE} = \sqrt{\frac{1}{n-p-1}\text{RSS}}.$$

For Simple, linear regression, p = 1. Here,

n -> number of observations
p -> number of predictors

Smaller the RSE, the better the model.

ii) **R-squared** -> This measure indicates how much of the underlying variance of the data is captured by the independent variables considered in the model.

Another derived metric, Adjusted R^2 also considers the number of predictors considered in the model. High deviated between the Adjusted R^2 and R^2 can indicate a poor-fitting model. This can also be cross-checked by measures like AIC and BIC to attain a more parsimonious fit.

Larger values of both the Adjusted R^2 and R^2 are better and indicate a better fitting model.

Mathematically,

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where $\text{TSS} = \sum(y_i - \bar{y})^2$ is the *total sum of squares.*

iii.) F – Stat -> Assesses multiple coefficients simultaneously, and compares the fitted model with the original model i.e. the intercept model.

The larger the F-stat, the better the model.

Mathematically,

$$F = \frac{(\text{TSS} - \text{RSS})/p}{\text{RSS}/(n-p-1)}$$

d.) **Training Set** -> The subset of a data or stand-alone data which is used to “train” the model for further pattern recognition.

Training error is measured by the Mean Squared Error(MSE) and Root Mean Squared Error (RMSE).

Testing Set -> The subset of a data or stand-alone data which is used to test the effectiveness of the created model to judge its out-of-sample performance.

Testing error is also measured by the Mean Squared Error(MSE) and Root Mean Squared Error (RMSE).

Part 2:

```
# Import the data


cars <- read.csv(file.choose())

View(cars)


# Set the required Seed and subset the data
```

```
set.seed(490)

random_seed <- c("Mersenne-Twister", 490)

index <- sample(1:nrow(cars),0.5*nrow(cars), replace = FALSE)


cars_train <- cars[index,]

cars_test <- cars[-index,]


############################ Simple Linear Regression
########################################


## Training data modeling


lm1 <- lm(mpg~weight, data = cars_train)

summary(lm1)

mse1 <- mean(residuals(lm1)^2)

mse1

rmse1 <- sqrt(mse1)

rmse1

rss <- sum(residuals(lm1)^2)

rss

rse <- sqrt(rss/197)

rse

lm1_predict <- predict(lm1, cars_test)

summary(lm1_predict)

lm1_test_ssl <- sum((cars_test$mpg - lm1_predict)^2)

sprintf("SSL/SSR/SSE: %f", lm1_test_ssl)

lm1_test_mse <- lm1_test_ssl/nrow(cars_test)

sprintf("MSE: %f", lm1_test_mse)
```

```
lm1_test_rmse <- sqrt(lm1_test_mse)

sprintf("RMSE: %f", lm1_test_rmse)

plot(mpg~weight, cars_test)

abline(lm1)


## Full data modeling


lm1_fd <- lm(mpg~weight, data = cars)

summary(lm1_fd)

mse1_fd <- mean(residuals(lm1_fd)^2)

mse1_fd

rmse1_fd <- sqrt(mse1_fd)

rmse1_fd

rss_fd <- sum(residuals(lm1_fd)^2)

rss_fd

rse_fd <- sqrt(rss_fd/197)

rse_fd


################################ Multiple Linear Regression
############################################


## Training data modeling


lm2 <- lm(mpg~cylinders + displacement + weight + acceleration + model.year, data = cars_train)

summary(lm2)

lm2_mse <- mean(residuals(lm2)^2)

lm2_mse

lm2_rmse <- sqrt(lm2_mse)

lm2_rss <- sum(residuals(lm2)^2)
```

```
lm2_rss

lm2_rse <- sqrt(lm2_rss/193)

lm2_rse

lm2_predict <- predict(lm2,cars_test)

summary(lm2_predict)

lm2_test_ssl <- sum((cars_test$mpg - lm2_predict)^2)

sprintf("SSL/SSR/SSE: %f", lm2_test_ssl)

lm2_test_mse <- lm2_test_ssl/nrow(cars_test)

sprintf("MSE: %f", lm2_test_mse)

lm2_test_rmse <- sqrt(lm2_test_mse)

sprintf("RMSE: %f", lm2_test_rmse)

scatter.smooth(cars_test$weight,cars_test$mpg,main = "weight ~ mpg")


## Full data modeling


lm2_fd <- lm(mpg~cylinders + displacement + weight + acceleration + model.year, data = cars)

summary(lm2_fd)

lm2_fd_mse <- mean(residuals(lm2_fd)^2)

lm2_fd_mse

lm2_fd_rmse <- sqrt(lm2_fd_mse)

lm2_fd_rss <- sum(residuals(lm2_fd)^2)

lm2_fd_rss

lm2_fd_rse <- sqrt(lm2_fd_rss/392)

lm2_fd_rse


####################################### Parsimonious Model
#######################################


## Training data modeling
```

```
lm3 <- lm(mpg~cylinders + model.year, data = cars_train)

summary(lm3)

lm3_mse <- mean(residuals(lm3)^2)

lm3_mse

lm3_rmse <- sqrt(lm3_mse)

lm3_rmse

lm3_rss <- sum(residuals(lm3)^2)

lm3_rss

lm3_rse <- sqrt(lm3_rss/196)

lm3_rse

lm3_predict <- predict(lm3,cars_test)

summary(lm3_predict)

lm3_test_ssl <- sum((cars_test$mpg - lm3_predict)^2)

sprintf("SSL/SSR/SSE: %f", lm3_test_ssl)

lm3_test_mse <- lm3_test_ssl/nrow(cars_test)

sprintf("MSE: %f", lm3_test_mse)

lm3_test_rmse <- sqrt(lm3_test_mse)

sprintf("RMSE: %f", lm3_test_rmse)


## Full data modeling


lm3_fd <- lm(mpg~cylinders + model.year, data = cars)

summary(lm3_fd)

lm3_fd_mse <- mean(residuals(lm3_fd)^2)

lm3_fd_mse

lm3_fd_rmse <- sqrt(lm3_fd_mse)

lm3_fd_rmse

lm3_fd_rss <- sum(residuals(lm3_fd)^2)
```

```
lm3_fd_rss

lm3_fd_rse <- sqrt(lm3_fd_rss/395)

lm3_fd_rse
```

Statistic	Simple Linear Regression			Multiple Regression			Multiple Regression (only significant variables)		
	Train	Test	All	Train	Test	All	Train	Test	All
MSE	18.63	18.97	18.78	10.47	13.07	11.65	16.10	19.28	17.68
RMSE	4.31	4.355	4.33	3.23	3.61	3.41	4.01	4.39	4.20
RSS	3707.65	-	7474.81	2084.48	-	4639.8	3205.25	-	7037.21
RSE	4.33	-	6.15	3.28	-	3.44	4.04	-	4.22
R^2	0.6842	-	0.691	0.822	-	0.806	0.724	-	0.708
Adj. R^2	0.6826	-	0.691	0.817	-	0.808	0.727	-	0.709
F-Statistic	426.8	-	888.9	178.8	-	331.4	261	-	483.2

<u>Parsimonious Model</u> -> -16.61 + (-2.99)*cylinders + 0.74 * Model.year [Basis the model based on full dataset]