4.1 Linear Regression

Table of Content


```
# Preparation
library(gapminder)
library(Metrics)
library(tidyverse)
```

Introduction

They are two main components to linear models:

1. Y-intercept (bias)
2. Weights for attributes/features

lm(y_var ~ x_var, data = dataset)

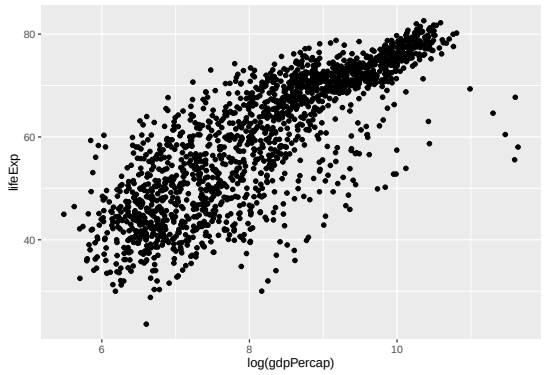
- lm is the function that is used to define a linear model
- y_var is the dependent variable
- x_var is the independent variable (aka attributes/features)
- We can have multiple features: lm(y_var ~ x_var_1 + x_var_2 + x_var_3, data=dataset)

Loss function

The goal of a linear model is to predict values that are as close as possible to the actual values. For instance if we make a linear model that predicts stock prices for tomorrow, ideally we would like the price to be close to what it will be. Say the stock A is at 10 CAD today and your models predicts that the price will be 5 CAD tomorrow, you might go ahead and sell your stocks. But if the tomorrow's price turns out to be 100 then your model made a huge mistake. So you might want to change your model so the error is lower. In this case you will need to have a metric to assess the accuracy of your model. One of the metrics used for regression tasks is mae. We will use the mae function to demonstrate the accuracy of various models.

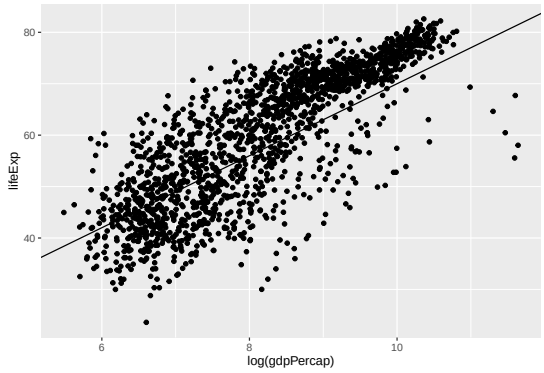
Case Study: Predicting GDP per Capita GDP per Capita is an economic metric used to assess the living situation in a country. The gapminder dataset has columns that we can use to assess predict GDP per Capita. For instance can we predict the

```
# Make the same chart, but look at the log of gdp per capita instead
ggplot(gapminder, aes(log(gdpPercap), lifeExp ) ) +
  geom_point()
```




```
# Use geom_abline to guess the relationship
ggplot(gapminder, aes(log(gdpPercap), lifeExp ) ) +
  geom_point() +
  geom_abline(incercept = 0, slope = 7)
```

```
## Warning in geom_abline(incercept = 0, slope = 7): Ignoring unknown parameters:
## `incercept`
```



Model fitting in R syntax

"Formula syntax" takes the form: lm(y_var ~ x_var, data = dataset)

The formula syntax is more commonly used and we will focus on that today

Predicting gdpPerCap

In order to illustrate linear models, models with different features are ran.

a. lifeExp is used solely to get the gdpPercap.
b. pop is used solely to get the gdpPercap.
c. [year, pop, gdpPercap] are used to get the gdpPercap.

```
# a. x = lifeExp
model1 <- lm( gdpPercap ~ lifeExp, data = gapminder)

# b. x = pop
model2 <- lm( gdpPercap ~ pop, data = gapminder)

# b. x = [year, pop, gdpPercap]
model3 <- lm(gdpPercap ~ pop + year + lifeExp, data = gapminder)
```

Making predictions

In order to make predictions, we use the predict function.

```
predictions <- predict(model1, gapminder)
```

Validating models

In order to decide which features should be used, we run the loss functions to see which model has a high accuracy and lower error.

```
# Getting the predictions
model1_pred <- predict(model1, gapminder)
model2_pred <- predict(model2, gapminder)
model3_pred <- predict(model3, gapminder)

# Loss functions

# MAE
mae <- function(y_true, y_pred) { sum(abs(y_true - y_pred)) / NROW(y_true) }

# MSE
mse <- function(y_true, y_pred) { sum((y_true - y_pred) ** 2 / NROW(y_true)) }

# RMSE
rmse <- function(y_true, y_pred) { sqrt(sum((y_true - y_pred) ** 2 / NROW(y_true)) ) }

error <- c("MAE", "RMSE", "MSE")
error_model1 <- c(mae(model1_pred, gapminder$gdpPercap), rmse(model1_pred, gapminder$gdpPercap), mse(mo
error_model2 <- c(mae(model2_pred, gapminder$gdpPercap), rmse(model2_pred, gapminder$gdpPercap), mse(mo
error_model3 <- c(mae(model3_pred, gapminder$gdpPercap), rmse(model3_pred, gapminder$gdpPercap), mse(mo

errors <- data.frame(error, error_model1, error_model2, error_model3)
head(errors)
```

```
##   error error_model1 error_model2 error_model3
## 1   MAE     4801.936     6535.112     4763.591
## 2  RMSE     8001.558     9851.332     7972.770
## 3   MSE 64024938.216 97048744.296 63565055.580
```