Project: Marathon Training with Linear Regression

MAT 4800 Introduction to Data Science

AUTHOR
Le' Sean Roberts

AFFILIATION
City Tech

PUBLISHED
May 4, 2023

Introduction

Source: https://media.giphy.com/media/BDagLpxFIm3SM/giphy.gif

- What features predict whether athletes will perform better than others?

> Specifically, we are interested in marathon runners, and looking at how the maximum distance ran per week during training predicts the time it takes a runner to end the race (race time is the variable `time_hrs`).

We will analyze the `marathon.csv` data using simple linear regression.

Question 1

Load the `marathon.csv` data and assign it to an object called `marathon`.

```r
library(readr)
marathon <- read_csv("C:/Users/verlene/Downloads/marathon.csv")
View(marathon)
head(marathon)
```

```
# A tibble: 6 × 13
    age   bmi female footwear group injury  mf_d mf_di mf_ti   max sprint  mf_s
  <dbl> <dbl>  <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>
1    35  23.6      0        2     1      2 42195     4 10295    60      1  4.10
2    33  22.5      0        2     2      2 42195     3 12292    50      0  3.43
3    38  25.6      0        2     3      1 42195     4 10980    65      0  3.84
4    34  22.6      0        2     1      1 42195     3 10694    88      1  3.95
5    39  25.0      0        2     1      1 42195     2 13452    51      0  3.14
```

```
6    33  24.3      1        2     2      1 42195     3 14940    40      0  2.82
# ℹ 1 more variable: time_hrs <dbl>
```

```r
dim(marathon)
```

```
[1] 929  13
```

Question 2

We will first split the dataset into the training and testing datasets, using 75% of the original data as the training data. Remember, we will be putting the test dataset away in a 'lock box' that we will comeback to later after we choose our final model.

In the `strata` argument of the `initial_split` function, place the variable we are trying to predict.

- What is the variable we are trying to predict?

ANSWER: we are trying to predict the time it takes a runner to end the race (race time is the variable `time_hrs`)

Assign your split dataset to an object named `marathon_split`.

Use the `training` function to assign your training dataset to an object named `marathon_training`. Similarly, use the `testing` function to assign your testing dataset to an object named `marathon_testing`.

```r
# First, selection the variables of interest
prediction_data <- select(marathon, max, time_hrs)
dim(prediction_data)
```

```
[1] 929   2
```

```r
predict_data <- as_tibble(prediction_data)
head(prediction_data)
```

```
# A tibble: 6 × 2
    max time_hrs
  <dbl>    <dbl>
1    60     2.86
2    50     3.41
3    65     3.05
4    88     2.97
5    51     3.74
6    40     4.15
```

```r
dim(predict_data)
```

```
[1] 929   2
```

```r
set.seed(2022)
# Splitting method for training and testing, and setting the [time_hrs] variable we want to predict.
time_split <- initial_split(predict_data, prop = 0.75, strata = time_hrs)

# Now constructing training data and testing data.
```

```r
marathon_training <- training(time_split)
marathon_testing <- testing(time_split)
head(marathon_training)
```

```
# A tibble: 6 × 2
    max time_hrs
  <dbl>    <dbl>
1    60     2.86
2    88     2.97
3    75     2.99
4    45     3.02
5   110     2.63
6   135     2.64
```

```r
dim(marathon_training)
```

```
[1] 696   2
```

```r
head(marathon_testing)
```

```
# A tibble: 6 × 2
    max time_hrs
  <dbl>    <dbl>
1    65     3.05
2    28     4.01
3    55     4.78
4    50     3.53
5    62     3.25
6   120     3.70
```

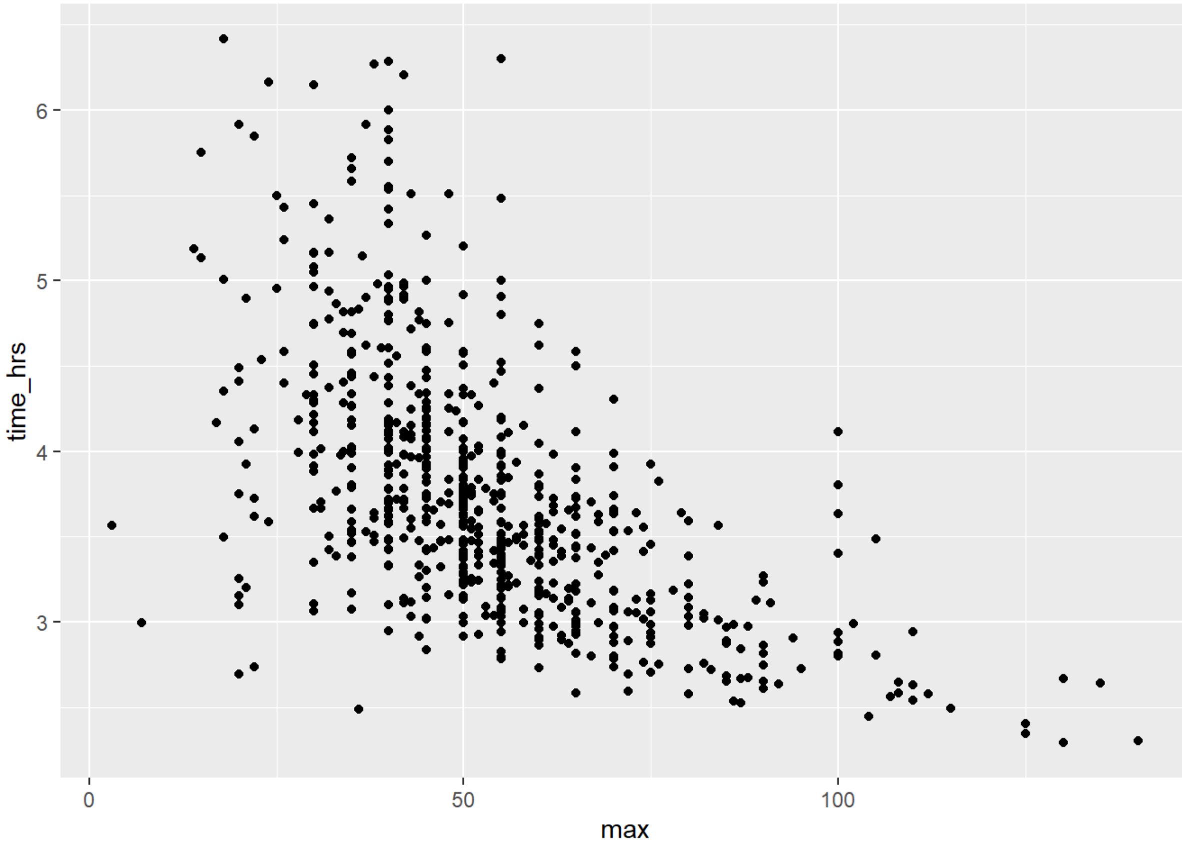
```r
dim(marathon_testing)
```

```
[1] 233   2
```

Question 3

Using only the observations in the training dataset, create a scatterplot to assess the relationship between race time (`time_hrs`) and maximum distance ran per week during training (`max`). Put `time_hrs` on the y-axis and `max` on the x-axis. Assign this plot to an object called `marathon_eda` . Remember to do whatever is necessary to make this an effective visualization.

```r
marathon_eda <- ggplot(marathon_training, aes(x = max, y = time_hrs)) +
  geom_point()

marathon_eda
```



Question 4

Now that we have our training data, the next step is to build a linear regression model specification. Thankfully, building other model specifications is quite straightforward since we will still go through the same procedure (indicate the function, the engine and the mode).

Instead of using the `nearest_neighbor` function, we will be using the `linear_reg` function to let `tidymodels` know we want to perform a linear regression. In the `set_engine` function, we have typically set `"kknn"` there for k-nn. Since we are doing a linear regression here, set `"lm"` as the engine. Finally, instead of setting `"classification"` as the mode, set `"regression"` as the mode.

Assign your answer to an object named `lm_spec`.

```
# Make the linear model specification
lm_spec<- linear_reg() |>
  set_engine("lm") |>
  set_mode("regression")
```

Question 5

After we have created our linear regression model specification, the next step is to create a recipe, establish a workflow analysis and fit our simple linear regression model.

First, create a recipe with the variables of interest (race time and max weekly training distance) using the training dataset and assign your answer to an object named `lm_recipe`.

Then, create a workflow analysis with our model specification and recipe. Remember to fit in the training dataset as well. Assign your answer to an object named `lm_fit`.

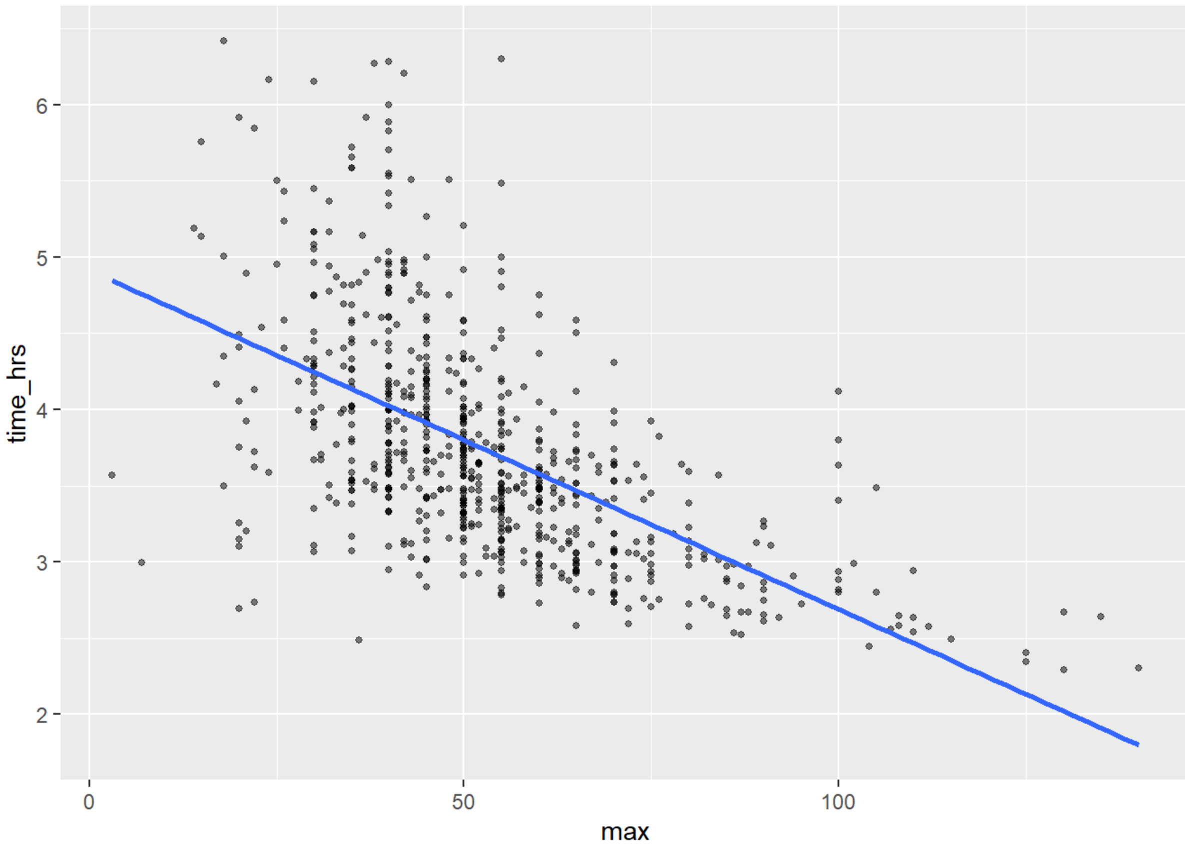
```r
# Put it all together in a workflow, then fit
lm_recipe <- recipe(time_hrs ~ max, data = marathon_training)
lm_fit <- workflow() |>
  add_recipe(lm_recipe) |>
  add_model(lm_spec) |>
  fit(data = marathon_training)
lm_fit
```

```
══ Workflow [trained] ══════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: linear_reg()

── Preprocessor ────────────────────────────────────────────────────────────────
0 Recipe Steps

── Model ───────────────────────────────────────────────────────────────────────

Call:
stats::lm(formula = ..y ~ ., data = data)

Coefficients:
(Intercept)          max
    4.91318     -0.02225
```

Question 6

Now, let's visualize the model predictions as a straight line overlaid on the training data. Use `geom_smooth` with `method = "lm"` and `se = FALSE` to visualize the predictions as a straight line. Name your plot `lm_predictions`.

```r
library(tidyverse)
library(tidymodels)
lm_predictions <- ggplot(marathon_training, aes(x = max, y = time_hrs)) +
  geom_point(alpha = 0.5, size = 1) +
  geom_smooth(method = lm, se = FALSE)
lm_predictions
```



Question 7

Great! We can now see the line of best fit on the graph. Now let's calculate the **RMSPE** using the **test data**. To get to this point, first, use the `lm_fit` to make predictions on the test data. Remember to bind the appropriate columns for the test data. Afterwards, collect the metrics and store it in an object called `lm_test_results`.

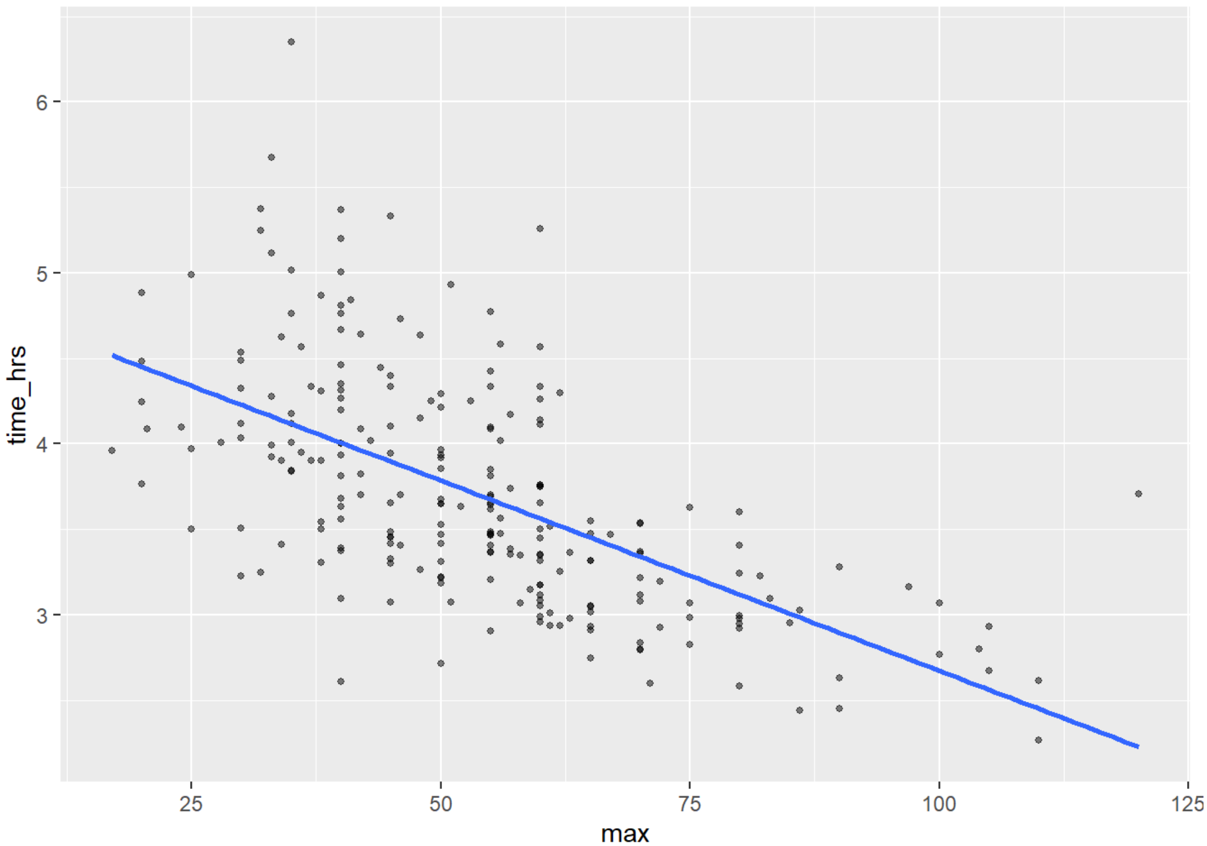
From `lm_test_results`, extract the RMPSE and return a single numerical value. Assign your answer to an object named `lm_rmspe`.

```
library(tidyverse)
library(tidymodels)
library(dplyr)
lm_test_results <- lm_fit |>
  predict(marathon_testing) |>
  bind_cols(marathon_testing) |>
  metrics(truth = time_hrs, estimate = .pred)
lm_rmspe <- lm_test_results |>
          filter(.metric == 'rmspe') |>
          select(.estimate) |>
  pull()
```

Question 8

Now, let's visualize the model predictions as a straight line overlaid on the test data. Use `geom_smooth` with `method = "lm"` and `se = FALSE` to visualize the predictions as a straight line. Name your plot `lm_predictions_test`. Remember to do whatever is necessary to make this an effective visualization.

```r
library(tidyverse)
library(tidymodels)
lm_predictions_testing <- ggplot(marathon_testing, aes(x = max, y = time_hrs)) +
  geom_point(alpha = 0.5, size = 1) +
  geom_smooth(method = lm, se = FALSE)
lm_predictions_testing
```



Given that the linear regression model is a straight line, we can write our model as a mathematical equation. We can get the two numbers we need for this from the coefficients, `(Intercept)` and `time_hrs`.

```r
# run this cell
lm_fit
```

```
══ Workflow [trained] ══════════════════════════════════════════════════════════
Preprocessor: Recipe
Model: linear_reg()

── Preprocessor ────────────────────────────────────────────────────────────────
0 Recipe Steps

── Model ───────────────────────────────────────────────────────────────────────

Call:
stats::lm(formula = ..y ~ ., data = data)

Coefficients:
```

```
(Intercept)          max
    4.91318     -0.02225
```

Question 9

Which of the following mathematical equations represents the model based on the numbers output in the cell above?

A. $Predicted\ race\ time\ (in\ hours) = 4.86 - 0.02 * max\ (in\ miles)$

B. $Predicted\ race\ time\ (in\ hours) = -0.02 + 4.86 * max\ (in\ miles)$

C. $Predicted\ max\ (in\ miles) = 4.86 - 0.02 *\ race\ time\ (in\ hours)$

D. $Predicted\ max\ (in\ miles) = -0.02 + 4.86 *\ race\ time\ (in\ hours)$

ANSWER: A IS THE CLOSEST.