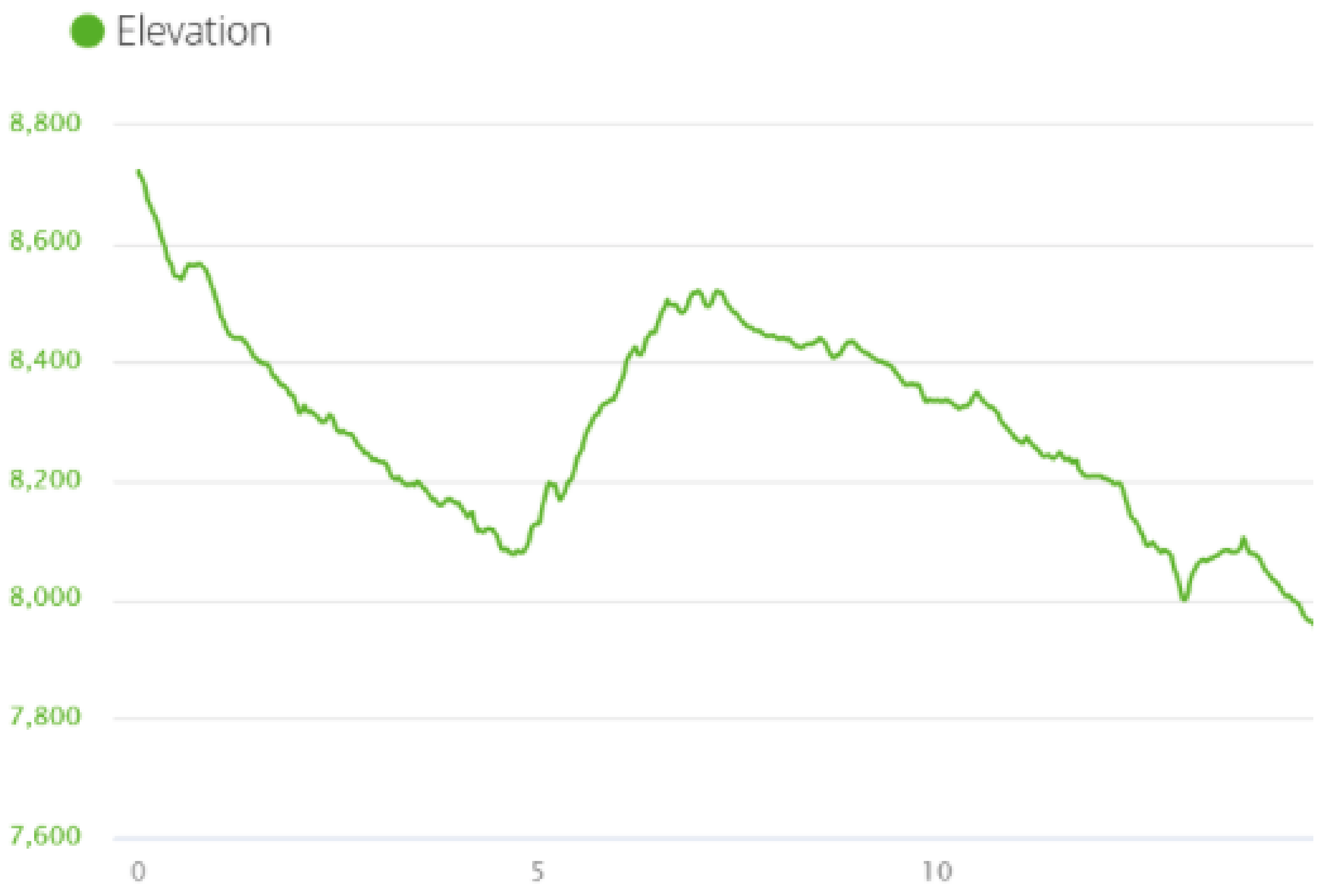
In-class exercise November 19th

Contents

Build a model to predict running times

Use the dataset to fit a linear model to predict how long it takes to a **26.2** mile race with **1,500** feet of vertial gain. Assume race take took place in May 2019. It is an out and back course. The figure below, of the elevation profile may be useful.



Getting started

- create a new RMarkdown document for these exercises
- download the data from google drive
- save the data in a sensible location
- take a look at the data

Load a data file

- use the `list.files()` function to create a vector of the **.csv** file names in the data, which does not contain the `elevation.csv` file
- write a function to load one of the running **.csv** files (note: you'll need the `readr` package)
- set the function to load the all columns as class **character**.

Try using:

```
col_types = cols(.default = "c")
```

Clean the data

- take a look at the file you just loaded
- write a function to tidy the column names

Update your file loading function to:

- clean the column names
- filter only rows that contain data for complete miles
- convert the time strings to `hms` class using `lubridate`
- convert the remaining character columns to numeric class
- add a column containing the file name
- reload a data file using the updated function

The `rename_all()`, `filter()`, `mutate_if()`, `mutate()` functions might be useful.

Load all the files

- load all the .csv files from the running folder into a list using `map()` from the `purrr` package
- how many files were loaded?

Summarise each run

- Write a function to summarize each training run. Think about what some useful summary statistics might be. Be sure to include the total distance of each run, average pace, and the file name in your summary.

- Use `map_df()` to create a dataframe summarizing each run.

- Add a variable for the total time each run takes

```r
# e.g.
run_sum <- function(x){
  summarise(x, distance = sum(distance, na.rm = TRUE), ...)
}
```

Merge the elevation dataset

The `elevation.csv` file contains the minimum, maximum, and mean elevation for each run.

- read in the `elevation.csv` file
- join the elevation data to the run summary file
- remove the `file_name` column

MFP package

The `mfp` package contains functions to identify variable combinations to include in linear models.

- use the (http://mfp.imbi.uni-freiburg.de)[mfp] library to identify the best model to predict duration

- the first argument the `mfp` function requires in a `formula` object. The formula should contain all the candidate variables.

- look at the website link above to get an idea what the `mfp` function does.

Create a formula object

Example formula:

```r
eqn <- formula(duration ~ fp(distance) + ...)
```

Run MFP

- run the `mfp()` function on the data set

```r
# example `mfp` code:
library(mfp)
mfp_run <- mfp(eqn,
               data = run_data,
               select = 0.05)
```

Extract the best model

- use the `summary()` function to display the output from `mfp`
- use the `$` operator to extract the model formula from the `mfp` object.
- use the model formula to fit a linear model to the running data
- print a summary of the linear model

Check the model fit

- you can use the `autoplot` function to check the model diagnostics
- do you want to remove any of the data points from your analysis

```r
library(ggfortify)
autoplot(mod)
```

Make your prediction!

- use the `predict.lm()`
