Summarizing The Weather

4/23/2021

Contents

1 Setup and Libraries **1**

2 Introduction **1**

3 Import the source data **1**

4 Confirm the results of the import **2**

5 Calculate the Coefficient of Barometric Pressure **3**

6 Graph Barometric Pressure **4**

1 Setup and Libraries

```
library(magrittr)
library(lubridate) #ymd_hms
library(tidyverse)
library(kableExtra) #kable
```

2 Introduction

This Code Clinic problem is about calculating statistics from a data set. It's easy stuff, but presents a good example of how different languages accomplish common tasks.

3 Import the source data

The data set is weather data captured from Lake Pend O'Reille in Northern Idaho — https://github.com/lyndadotcom/LPO_weatherdata. We have almost 20 megabytes of data from the years 2012 thorough 2015. That data is available in the folder with other exercise files. Each observation in the data includes several variables and the data is straightforward.

```
mytempfile <- tempfile()

readOneFile <- function(dataPath) {
  read.table(dataPath,
             header = TRUE,
```

date	time	Air_Temp	Barometric_Press	Dew_Point	Relative_Humidity	Wind_Dir	Wind_Gust	Wind_Speed
2012_01_01	00:02:14	34.3	30.5	26.9	74.2	346.4	11	3.6
2012_01_01	00:08:29	34.1	30.5	26.5	73.6	349.0	12	8.0
2012_01_01	00:14:45	33.9	30.6	26.8	75.0	217.8	12	9.2

	date	time	Air_Temp	Barometric_Press	Dew_Point	Relative_Humidity	Wind_Dir	Wind_Gust	Wind_Speed
315463	2015_06_04	01:04:21	57.7	29.95	51.22	79.0	179.41	9	6.8
315464	2015_06_04	01:06:59	57.7	29.95	51.28	79.2	167.78	11	8.8
315465	2015_06_04	01:09:21	57.7	29.95	51.22	79.0	163.40	12	10.0

```
             stringsAsFactors = FALSE)
}
```

With the large file, we should create the progress bar to see how long we should know to wait for the reading into r by using `txtProgressBar` function.

```
myProgressBar <- txtProgressBar(min = 2012, max = 2015, style = 3)
```

```
for (dataYear in 2012:2015) {

  dataPath <-
    paste0(
      link,
      dataYear,
      ".txt")

  if (exists("LPO_weather_data")) {
    mytempfile <- readOneFile(dataPath)
    LPO_weather_data <- rbind(LPO_weather_data, mytempfile)
  } else {
    LPO_weather_data <- readOneFile(dataPath)
  }
  setTxtProgressBar(myProgressBar, value = dataYear)

}
```

```
##   |                                                                              |
```

4 Confirm the results of the import

```
kbl(head(LPO_weather_data, n = 3), booktabs = T) %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

```
kbl(tail(LPO_weather_data, n = 3), booktabs = T) %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

```
print(paste("Number of rows imported: ", nrow(LPO_weather_data)))
```

```
## [1] "Number of rows imported:  315465"
```

5 Calculate the Coefficient of Barometric Pressure

The problem is simple: Write a function that accepts … a beginning date and time …and… an ending date and time…

```
startDateTime <- "2014-01-02 12:03:34"
endDateTime <- "2014-01-04 12:03:34"
```

…then… inclusive of those dates and times return the coefficient of the slope of barometric pressure.

helper function to get a subset of LPO_weather_data observations are the date range variables are barometric pressure, date, and time

> Transform dates stored as character or numeric vectors to POSIXct objects. The ymd_hms() family of functions recognizes all non-alphanumeric separators (with the exception of "." if frac = TRUE) and correctly handles heterogeneous date-time representations. For more flexibility in treatment of heterogeneous formats, see low level parser parse_date_time().

```
getBaromPressures <- function(dateTimeInterval) {
  subset(LPO_weather_data,
         ymd_hms(paste(date, time)) %within% dateTimeInterval,
         select = c(Barometric_Press, date, time)
         )
}

calculateBaroPress <- function(startDateTime, endDateTime) {
  dateTimeInterval <- interval(ymd_hms(startDateTime),
                               ymd_hms(endDateTime))

  baroPress <- getBaromPressures(dateTimeInterval)

  slope <- ymd_hms(paste(baroPress$date, baroPress$time))

  lm(Barometric_Press ~ slope, data = baroPress)

}

calculateBaroPress(startDateTime, endDateTime)
```

```
##
## Call:
## lm(formula = Barometric_Press ~ slope, data = baroPress)
##
## Coefficients:
## (Intercept)        slope
##  -3.090e+03    2.245e-06
```

A rising slope indicates an increasing barometric pressure, which typically means fair and sunny weather.

Figure 1: Barometric ~ rising slope (adapted from LinkedIn Learning)

A falling slope indicates a decreasing barometric pressure, which typically means stormy weather.

We're only asking for the coefficient – but some may choose to generate a graph of the results as well.

6 Graph Barometric Pressure

```
graphBaroPressure <- function(startDateTime, endDateTime ) {

  dateTimeInterval <- interval(ymd_hms(startDateTime),
                               ymd_hms(endDateTime))

  baroPress <- getBaromPressures(dateTimeInterval)

  thisDateTime <- ymd_hms(paste(baroPress$date, baroPress$time))

  plot(
    x = thisDateTime,
    y = baroPress$Barometric_Press,
    xlab = "Date and Time",
    ylab = "Barometric Pressure",
    main = paste(
      "Barometric Pressure from ",
      ymd_hms(startDateTime),
      "to",
      ymd_hms(endDateTime)
    )
```

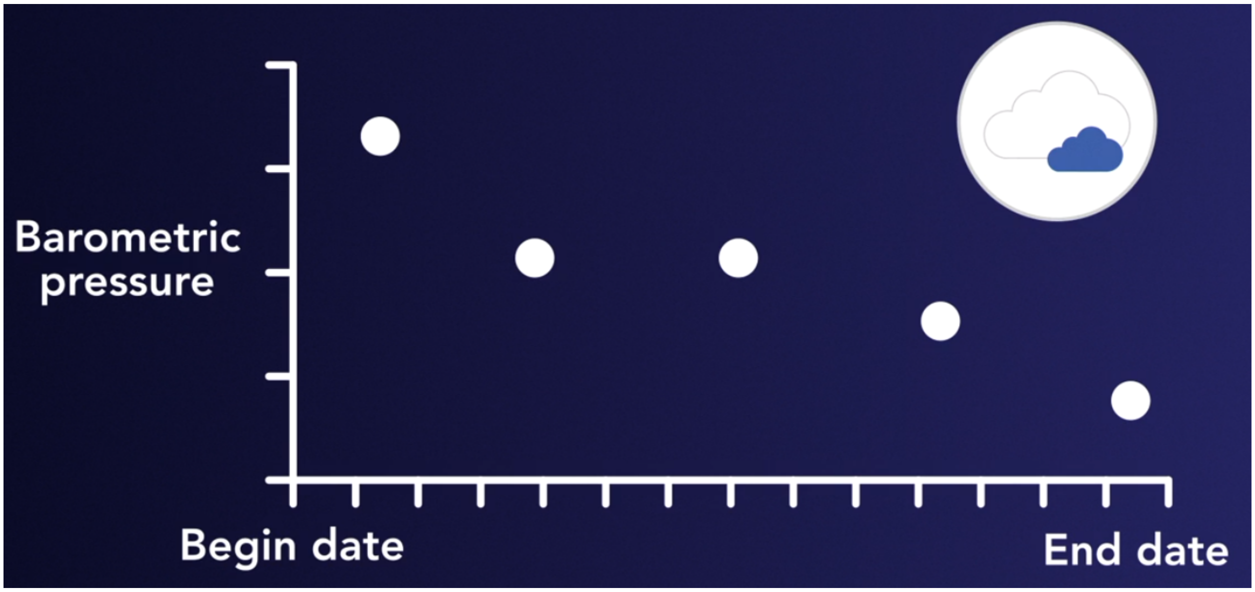
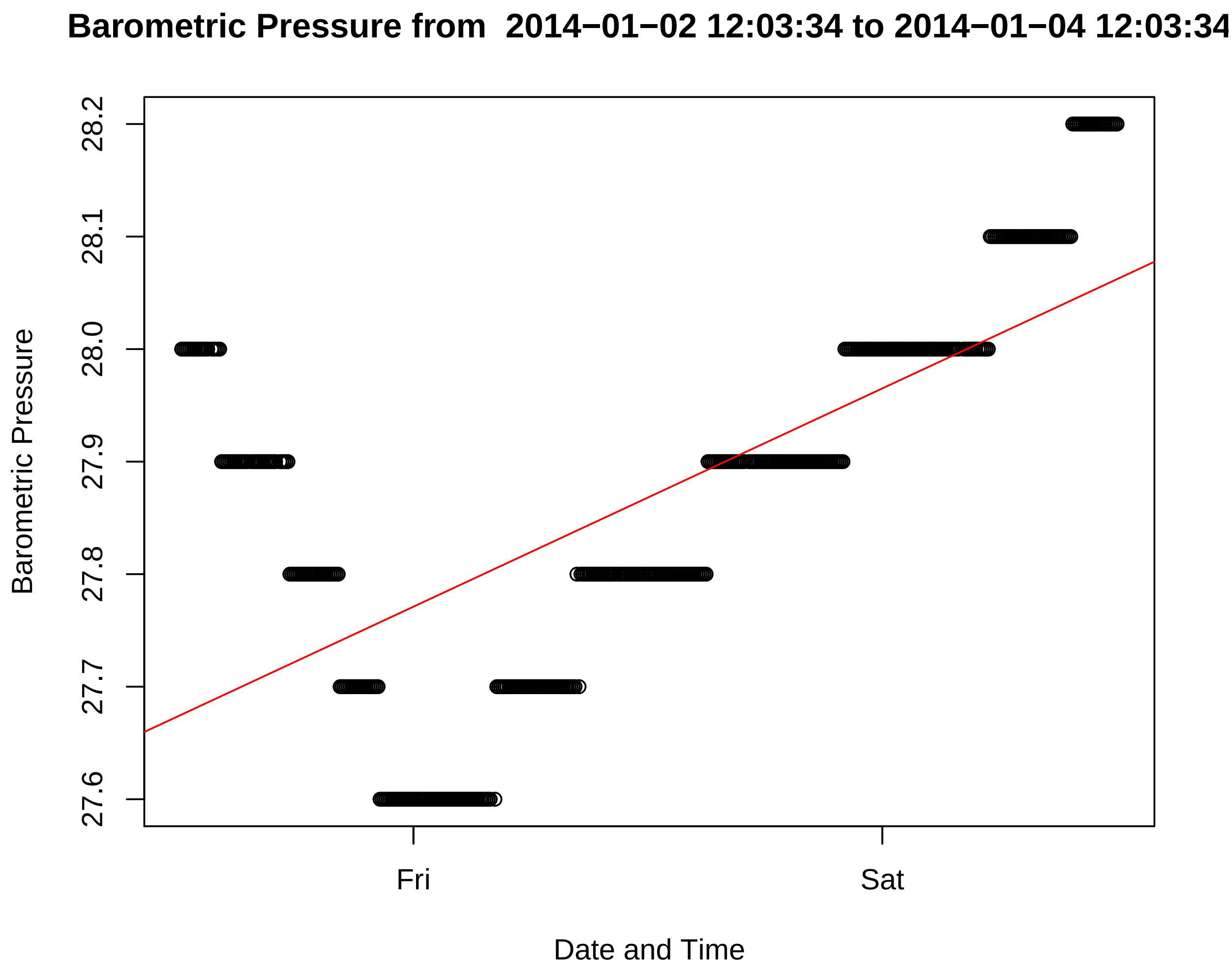


Figure 2: Barometric ~ falling slope (adapted from LinkedIn Learning)

```
  )
  abline(calculateBaroPress(startDateTime, endDateTime), col = "red")
}

graphBaroPressure(startDateTime, endDateTime)
```


Barometric Pressure from 2014−01−02 12:03:34 to 2014−01−04 12:03:34
Barometric Pressure
28.2
28.1
28.0
27.9
27.8
27.7
27.6
Fri
Sat
Date and Time