Data Transformation Wifi

10/20/2020

```
options(digits = 3, warn = -1)
```

Input: *Aggregated_Reps_Wifi.csv*

Factors (Perfume.js metrics):

1. **headerSize** - size of the header
2. **timeToFirstByte** (ms) - Time to First Byte (TTFB) - The amount of time ittakes for the server to send the first payload to the client
3. **downloadTime** - Download Time (ms) - Response time only (download)
4. **totalTime** - Total time (ms) - Request plus response time (network only)
5. **totalLoadTime** (ms) - The time taken for the entire page to be loaded (the time taken until the onload callback is called)
6. **fcp** - First Contentful Paint (ms) - The time from when the page starts loading to when the first bit of meaningful content is rendered
7. **fp** - First Paint (ms) - The exact time taken for the browser to render anything as visually different from what was on the screen before navigation

Numerical outcome (dependent variable):

batterystats_Joule_calculated - energy efficiency measured with battery stats (Joules)

Read the data:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------------------------- tidyverse
1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.3
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------ tidyverse_conflic
ts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
read_data <- function(file_path) {
  wifi_df = read_csv(file_path, col_names = TRUE)

  #Keep only the relevant columns
  keeps <- c("subject","headerSize", "fetchTime", "timeToFirstByte", "downloadTime", "totalTime", "totalLoadTime", "fcp", "f
p", "batterystats_Joule_calculated")
  wifi_df = wifi_df[keeps]
}

wifi_df = read_data("./Aggregated_Reps_Wifi_all.csv")
```

```
##
## -- Column specification ------------------------------------------------------------------------------------
-------
## cols(
##   X1 = col_double(),
##   device = col_character(),
##   subject = col_character(),
##   browser = col_character(),
##   batterystats_Joule_calculated = col_double(),
##   headerSize = col_double(),
##   workerTime = col_double(),
##   fetchTime = col_double(),
##   timeToFirstByte = col_double(),
##   downloadTime = col_double(),
##   totalTime = col_double(),
##   totalLoadTime = col_double(),
##   dnsLookupTime = col_double(),
##   fcp = col_double(),
##   fp = col_double(),
##   timestamp = col_character()
## )
```

```
#Count of rows and columns
dim(wifi_df)
```

```
## [1] 330  10
```

```
#View top rows of the dataset
View(wifi_df)
```

Aggregate runs for each subject (obtain 1 row per subject) We first order by subject, then average the measurements from each subject.

```
library(dplyr)
wifi_df_sorted = arrange(wifi_df, subject)

n = dim(wifi_df_sorted)[1]

wifi_df_aggregated = data.frame()
i = 1

while (i<=330){
  j = i + 10
  subj = as.character(unlist(wifi_df_sorted[i,]["subject"]))
  headerSizeAvg = colMeans(slice(wifi_df_sorted, i:j)["headerSize"])
  fetchTimeAvg = colMeans(slice(wifi_df_sorted, i:j)["fetchTime"])
  timeToFirstByteAvg = colMeans(slice(wifi_df_sorted, i:j)["timeToFirstByte"])
  totalTimeAvg = colMeans(slice(wifi_df_sorted, i:j)["totalTime"])
  downloadTimeAvg = colMeans(slice(wifi_df_sorted, i:j)["downloadTime"])
  totalLoadTimeAvg = colMeans(slice(wifi_df_sorted, i:j)["totalLoadTime"])
  fcpAvg = colMeans(slice(wifi_df_sorted, i:j)["fcp"])
  fpAvg = colMeans(slice(wifi_df_sorted, i:j)["fp"])
  batteryStatsAvg = colMeans(slice(wifi_df_sorted, i:j)["batterystats_Joule_calculated"])

  i = i+11

  avgRun = data.frame(subj, fetchTimeAvg, headerSizeAvg, timeToFirstByteAvg, totalTimeAvg, downloadTimeAvg, totalLoadTimeAv
g, fcpAvg, fpAvg, batteryStatsAvg)
  names(avgRun) = c("subject", "fetchTimeAvg", "headerSizeAvg", "timeToFirstByteAvg", "totalTimeAvg", "downloadTimeAvg", "to
talLoadTimeAvg", "fcpAvg", "fpAvg", "batteryStatsAvg")
  wifi_df_aggregated = rbind(wifi_df_aggregated, avgRun)
}

dim(wifi_df_aggregated)
```

```
## [1] 30 10
```

Normalize the data by scaling the score values to range [0, 1], having observed the minimum and the maximum of the Perfume.js metrics.

```
normalize = function(x){
  return((x-min(x)) /(max(x)-min(x)))
}
```

Use the **normalize** function to perform min-max normalization on the Perfume.js metrics.

```
headerSizeNormalized = as.vector(unlist(lapply(wifi_df_aggregated['headerSizeAvg'], normalize)))
fetchTimeNormalized = as.vector(unlist(lapply(wifi_df_aggregated['fetchTimeAvg'], normalize)))
timeToFirstByteNormalized = as.vector(unlist(lapply(wifi_df_aggregated['timeToFirstByteAvg'], normalize)))
downloadTimeNormalized = as.vector(unlist(lapply(wifi_df_aggregated['downloadTimeAvg'], normalize)))
totalTimeNormalized = as.vector(unlist(lapply(wifi_df_aggregated['totalTimeAvg'], normalize)))
totalLoadTimeNormalized = as.vector(unlist(lapply(wifi_df_aggregated['totalLoadTimeAvg'], normalize)))
fcpNormalized = as.vector(unlist(lapply(wifi_df_aggregated['fcpAvg'], normalize)))
fpNormalized = as.vector(unlist(lapply(wifi_df_aggregated['fpAvg'], normalize)))
```

Add normalized columns to wifi_df_aggregated

```
wifi_df_aggregated$headerSizeNormalized = headerSizeNormalized
wifi_df_aggregated$fetchTimeNormalized = fetchTimeNormalized
wifi_df_aggregated$timeToFirstByteNormalized = timeToFirstByteNormalized
wifi_df_aggregated$downloadTimeNormalized = downloadTimeNormalized
wifi_df_aggregated$totalTimeNormalized = totalTimeNormalized
wifi_df_aggregated$totalLoadTimeNormalized = totalLoadTimeNormalized
wifi_df_aggregated$fcpNormalized = fcpNormalized
wifi_df_aggregated$fpNormalized = fpNormalized
dim(wifi_df_aggregated)
```

```
## [1] 30 18
```

Obtain a composed performance score for each of the categories (navigation timing and load speed), by computing an average of the associated metrics.

Consider **headerSize**, **fetchTime**, **timeToFirstByte**, **downloadTime**, **totalTime** for navigation timing

```
 wifi_df_aggregated$navigationMean <- rowMeans(wifi_df_aggregated[,c('headerSizeNormalized', 'fetchTimeNormalized', 'timeToF
irstByteNormalized', 'downloadTimeNormalized', 'totalTimeNormalized')], na.rm=TRUE)
```

Consider **fp**, **fcp**, **totalLoadTime** for load speed.

```
wifi_df_aggregated$loadMean <- rowMeans(wifi_df_aggregated[,c('fpNormalized', 'fcpNormalized', 'totalLoadTimeNormalized')],
 na.rm=TRUE)
```

In order to simplify the statistical analysis, quantify the scores for each performance category by transforming the ratio measures into ordinal ones. To this end, define three levels for navigation timing/load speed, respectively, using two cutting points. We obtain an equal number of subjects on each level (low,average and high).

Sort **wifi_df** by average navigation timing **navigationMean**.

```
library(dplyr)
wifi_navigation_df = arrange(wifi_df_aggregated, navigationMean)
```

Sort **wifi_df** by load speed average **loadMean**.

```
wifi_load_df = arrange(wifi_df_aggregated, loadMean)
```

Append low/average/high performance levels (ordinal measure).

```
performanceLvl <- numeric(30)
for (i in 1:10) {
  performanceLvl[i] <- "low"
}
for (i in 11:20) {
  performanceLvl[i] <- "average"
}
for (i in 21:30) {
  performanceLvl[i] <- "high"
}

# Append new column (performanceLvl) to wifi_navigation_df
wifi_navigation_df = cbind(wifi_navigation_df, data.frame(performanceLvl, stringsAsFactors=FALSE))
# Append new column (performanceLvl) to wifi_load_df
wifi_load_df = cbind(wifi_load_df, data.frame(performanceLvl, stringsAsFactors=FALSE))
```

Clean dfs, only keep relevant columns

```
keeps <- c("subject", "navigationMean","headerSizeAvg", "timeToFirstByteAvg", "fetchTimeAvg", "downloadTimeAvg", "totalTimeA
vg", "navigationMean","headerSizeNormalized", "timeToFirstByteNormalized", "fetchTimeNormalized", "downloadTimeNormalized",
"totalTimeNormalized", "batteryStatsAvg", "performanceLvl")
wifi_navigation_df = wifi_navigation_df[keeps]

keeps <- c("subject", "totalLoadTimeAvg", "fcpAvg", "fpAvg", "totalLoadTimeNormalized", "fcpNormalized", "fpNormalized", "lo
adMean", "batteryStatsAvg", "performanceLvl")
wifi_load_df = wifi_load_df[keeps]
```

Create new **website** column with shortened names.

```
shorten_webpage_name <- function(fullName){
  name <- strsplit(as.character(fullName), '[-]')
  shortName <- paste(name[[1]][8], name[[1]][9], name[[1]][10], sep=".")
  return(shortName)
}

for (i in 1:length(wifi_navigation_df$subject)) {
  subject = wifi_navigation_df$subject[i]
  wifi_navigation_df$website[i] = shorten_webpage_name(subject)
  wifi_load_df$website[i] = shorten_webpage_name(subject)
}
```

Save transformed data for navigation timing:

```
write.csv(wifi_navigation_df,"NavigationTiming_Wifi.csv", row.names = TRUE)
```

Save transformed data for load speed:

```
write.csv(wifi_load_df,"LoadSpeed_Wifi.csv", row.names = TRUE)
```