Case Study Bellabeat FitBit

Mario Palma

2024-05-22

I've used Excel in the first part of the analysis, with "dailyActivity" and "weightLogInfo" due the size of the files. Also, I wanted to use R in the analysis to practice with a different tool. R is a great tool to use with the CSV files "minuteSleep" due the large size of those files.

"minuteSleep" are the CSV files that I could merged from each month. As some other are just found in one of the month only.

```r
# Install and Load the packages
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```r
install.packages("skimr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```r
install.packages("here")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```r
install.packages("janitor")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```r
install.packages("dplyr")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

```r
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library("skimr")
library("here")
```

```
## here() starts at /cloud/project
```

```r
library("janitor")
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
library("dplyr")
library(ggplot2)
```

Import data:

I first uploaded 2 CSV files minuteSleep1, minuteSleep2.

CSV files Stored in a folder called “Data” in Case Study Bellabeat Fitbit Workspace.

Number 1 and 2 at the end of each file name will indicate the month while exporting the files.

1 = 12/03/16 - 11/04/16

2 = 12/04/16 - 12/05/16

I created a data frame for each data file.

I used this two files about sleep minutes data because both can be found in each month and be merge to get the whole picture.

```r
minuteSleep1_df <- read.csv("/cloud/project/Data/minuteSleep/minuteSleep1.csv")
#View(minuteSleep1_df)

minuteSleep2_df <- read.csv("/cloud/project/Data/minuteSleep/minuteSleep2.csv")
#View(minuteSleep2_df)
```

To get summaries of the data frame and a quick idea of the data set I used the following functions.

- skim_without_charts()
- glimpse()
- head()

```r
colnames(minuteSleep1_df)
```

```
## [1] "Id"    "date"  "value" "logId"
```

```r
head(minuteSleep1_df)
```

```
##           Id                 date value       logId
## 1 1503960366 3/13/2016 2:39:30 AM     1 11114919637
## 2 1503960366 3/13/2016 2:40:30 AM     1 11114919637
## 3 1503960366 3/13/2016 2:41:30 AM     1 11114919637
## 4 1503960366 3/13/2016 2:42:30 AM     1 11114919637
## 5 1503960366 3/13/2016 2:43:30 AM     1 11114919637
## 6 1503960366 3/13/2016 2:44:30 AM     1 11114919637
```

```r
str(minuteSleep1_df)
```

```
## 'data.frame':    198559 obs. of  4 variables:
##  $ Id   : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ date : chr  "3/13/2016 2:39:30 AM" "3/13/2016 2:40:30 AM" "3/13/2016 2:41:30 AM" "3/13/2016 2:42:
##  $ value: int  1 1 1 1 1 1 2 2 1 1 ...
##  $ logId: num  1.11e+10 1.11e+10 1.11e+10 1.11e+10 1.11e+10 ...
```

```r
glimpse(minuteSleep1_df)
```

```
## Rows: 198,559
## Columns: 4
## $ Id    <dbl> 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503~
## $ date  <chr> "3/13/2016 2:39:30 AM", "3/13/2016 2:40:30 AM", "3/13/2016 2:41:~
## $ value <int> 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ logId <dbl> 11114919637, 11114919637, 11114919637, 11114919637, 11114919637,~
```

```r
skim_without_charts(minuteSleep1_df)
```

Table 1: Data summary

Name	minuteSleep1_df
Number of rows	198559
Number of columns	4
Column type frequency:	
character	1
numeric	3
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
date	0	1	19	21	0	54523	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
Id	0	1	4.824304e+09	2.173935e+09	1503960366	2347167796	4702921684	6775888955	8792009665
value	0	1	1.090000e+00	3.100000e-01	1	1	1	1	3
logId	0	1	1.124161e+10	7.969858e+07	11103653021	11165512026	11243951252	11310735495	11374876178

```r
colnames(minuteSleep2_df)
```

```
## [1] "Id"    "date"  "value" "logId"
```

```r
head(minuteSleep2_df)
```

```
##           Id                 date value       logId
## 1 1503960366 4/12/2016 2:47:30 AM     3 11380564589
## 2 1503960366 4/12/2016 2:48:30 AM     2 11380564589
## 3 1503960366 4/12/2016 2:49:30 AM     1 11380564589
## 4 1503960366 4/12/2016 2:50:30 AM     1 11380564589
## 5 1503960366 4/12/2016 2:51:30 AM     1 11380564589
## 6 1503960366 4/12/2016 2:52:30 AM     1 11380564589
```

```
str(minuteSleep2_df)
```

```
## 'data.frame':    188521 obs. of  4 variables:
##  $ Id   : num  1.5e+09 1.5e+09 1.5e+09 1.5e+09 1.5e+09 ...
##  $ date : chr  "4/12/2016 2:47:30 AM" "4/12/2016 2:48:30 AM" "4/12/2016 2:49:30 AM" "4/12/2016 2:50:3
##  $ value: int  3 2 1 1 1 1 1 2 2 2 ...
##  $ logId: num  1.14e+10 1.14e+10 1.14e+10 1.14e+10 1.14e+10 ...
```

```
glimpse(minuteSleep2_df)
```

```
## Rows: 188,521
## Columns: 4
## $ Id    <dbl> 1503960366, 1503960366, 1503960366, 1503960366, 1503960366, 1503~
## $ date  <chr> "4/12/2016 2:47:30 AM", "4/12/2016 2:48:30 AM", "4/12/2016 2:49:~
## $ value <int> 3, 2, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 2, 1, 1, 1, 1, 1, 1~
## $ logId <dbl> 11380564589, 11380564589, 11380564589, 11380564589, 11380564589,~
```

```
skim_without_charts(minuteSleep2_df)
```

Table 4: Data summary

Name	minuteSleep2_df
Number of rows	188521
Number of columns	4
Column type frequency:	
character	1
numeric	3
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
date	0	1	19	21	0	49773	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
Id	0	1	4.996595e+09	2.066950e+09	1503960366	3977333714	4702921684	6962181067	8792009665
value	0	1	1.100000e+00	3.300000e-01	1	1	1	1	3
logId	0	1	1.149611e+10	6.822863e+07	11372227280	11439308639	11501142214	11552534115	11616251768

Correct date column formatting:

The column "date" is not formatted correctly. I changed it for an easy workflow later on. I also checked by columns the number of NA on both files.

```
minuteSleep1_df$date <- trimws(minuteSleep1_df$date)
minuteSleep1_df$date <- as.POSIXct(minuteSleep1_df$date, tryFormats=c("%Y-%m-%d %H:%M:%S", "%d/%m/%Y #%

minuteSleep2_df$date <- trimws(minuteSleep2_df$date)
minuteSleep2_df$date <- as.POSIXct(minuteSleep2_df$date, tryFormats=c("%Y-%m-%d %H:%M:%S", "%d/%m/%Y %H

sum(is.na(minuteSleep1_df$Id))
```

```
## [1] 0
```

```
sum(is.na(minuteSleep1_df$date))
```

```
## [1] 0
```

```
sum(is.na(minuteSleep1_df$value))
```

```
## [1] 0
```

```
sum(is.na(minuteSleep1_df$logId))
```

```
## [1] 0
```

```
sum(is.na(minuteSleep2_df$Id))
```

```
## [1] 0
```

```
sum(is.na(minuteSleep2_df$date))
```

```
## [1] 0
```

```
sum(is.na(minuteSleep2_df$value))
```

```
## [1] 0
```

```
sum(is.na(minuteSleep2_df$logId))
```

```
## [1] 0
```

Functions for cleaning columns:

- rename()
- rename_with()
- clean_names()

To follow with the file-naming convention that I set up at the beginning of the project, I change the name of the column "Id" to "id" for a **camelCase** multiword delimited. It will help to don't have any problems when typing the columns through the analysis.

```
minuteSleep1_df <- tibble(rename(minuteSleep1_df, id=Id))

#Another way of cleaning columns
minuteSleep2_df <- as_tibble(minuteSleep2_df) %>%
  rename(id=Id)
```

First I counted the number of uniques ID on each dataset(month).

```r
n_distinct(minuteSleep1_df$id)
```

```
## [1] 23
```

```r
n_distinct(minuteSleep2_df$id)
```

```
## [1] 24
```

Delete duplicates rows and **drop NA values** on each data frame:

- minuteSleep1_df
- minuteSleep2_df

```r
minuteSleep1_df <- minuteSleep1_df %>% drop_na()
minuteSleep2_df <- minuteSleep2_df %>% drop_na()


minuteSleep1_df <- distinct(minuteSleep1_df)
minuteSleep2_df <- distinct(minuteSleep2_df)
```

Organize data:

- arrange()
- group_by()
- filter()
- summarize()
- drop_na()
- mean()
- max()

Organize the data by date for me to have a quick look

```r
arrange(minuteSleep1_df,minuteSleep1_df$date)
```

```
## # A tibble: 198,034 x 4
##            id date                value       logId
##         <dbl> <dttm>              <int>       <dbl>
##  1 5577150313 2016-03-11 21:19:30     1 11109426118
##  2 5577150313 2016-03-11 21:20:30     1 11109426118
##  3 5577150313 2016-03-11 21:21:30     1 11109426118
##  4 5577150313 2016-03-11 21:22:30     1 11109426118
##  5 5577150313 2016-03-11 21:23:30     1 11109426118
##  6 5577150313 2016-03-11 21:24:30     1 11109426118
##  7 5577150313 2016-03-11 21:25:30     1 11109426118
##  8 5577150313 2016-03-11 21:26:30     1 11109426118
##  9 5577150313 2016-03-11 21:27:30     1 11109426118
## 10 5577150313 2016-03-11 21:28:30     1 11109426118
## # i 198,024 more rows
```

```r
arrange(minuteSleep2_df,minuteSleep2_df$date)
```

```
## # A tibble: 187,978 x 4
##            id date                value       logId
##         <dbl> <dttm>              <int>       <dbl>
##  1 2026352035 2016-04-11 20:48:00     2 11372566564
##  2 2026352035 2016-04-11 20:49:00     1 11372566564
##  3 2026352035 2016-04-11 20:50:00     1 11372566564
##  4 2026352035 2016-04-11 20:51:00     1 11372566564
##  5 2026352035 2016-04-11 20:52:00     1 11372566564
##  6 2026352035 2016-04-11 20:53:00     1 11372566564
##  7 2026352035 2016-04-11 20:54:00     1 11372566564
##  8 2026352035 2016-04-11 20:55:00     1 11372566564
##  9 2026352035 2016-04-11 20:56:00     1 11372566564
## 10 2026352035 2016-04-11 20:57:00     1 11372566564
## # i 187,968 more rows
```

I checked what dates time frame are in the files.

```
min(minuteSleep1_df$date)
```

```
## [1] "2016-03-11 21:19:30 UTC"
```

```
max(minuteSleep1_df$date)
```

```
## [1] "2016-04-12 08:35:00 UTC"
```

```
min(minuteSleep2_df$date)
```

```
## [1] "2016-04-11 20:48:00 UTC"
```

```
max(minuteSleep2_df$date)
```

```
## [1] "2016-05-12 09:56:00 UTC"
```

Merging both files.

- after merging there is a total of 382780 rows.

```
# Compare columns of the 2 data frames
compare_df_cols(minuteSleep1_df,
                minuteSleep2_df)
```

```
##   column_name minuteSleep1_df minuteSleep2_df
## 1        date POSIXct, POSIXt POSIXct, POSIXt
## 2          id         numeric         numeric
## 3       logId         numeric         numeric
## 4       value         integer         integer
```

```
mergedData <- full_join(minuteSleep1_df,minuteSleep2_df)
```

```
## Joining with `by = join_by(id, date, value, logId)`
```

```
summary(mergedData)
```

```
##        id                  date                             value      
##  Min.   :1.504e+09   Min.   :2016-03-11 21:19:30.00   Min.   :1.000  
##  1st Qu.:3.977e+09   1st Qu.:2016-03-27 20:15:45.00   1st Qu.:1.000  
##  Median :4.703e+09   Median :2016-04-11 02:27:00.00   Median :1.000  
##  Mean   :4.910e+09   Mean   :2016-04-11 03:11:35.36   Mean   :1.091  
##  3rd Qu.:6.776e+09   3rd Qu.:2016-04-26 04:45:00.00   3rd Qu.:1.000  
##  Max.   :8.792e+09   Max.   :2016-05-12 09:56:00.00   Max.   :3.000  
##      logId          
##  Min.   :1.110e+10  
##  1st Qu.:1.124e+10  
##  Median :1.137e+10
##  Mean   :1.137e+10
##  3rd Qu.:1.150e+10
##  Max.   :1.162e+10
```

I now worked with the "mergedDatacopy" this way if any changes made in the data frame I just need to go back here and not all the way from the beginning.

```r
mergedDataCopy <- arrange(mergedData,id,date)

n_distinct(mergedDataCopy$id)
```

```
## [1] 25
```

```r
n_distinct(mergedDataCopy$value)
```

```
## [1] 3
```

Transform data:

- separate()
- unite()
- mutate()

Split the date column into date and time. This way I can group the count of each sleeping state by days later on .

```r
mergedDataCopy$date = as.POSIXct(mergedDataCopy$date,
                                           format = "%m/%d/%Y %I:%M:%S %p",
                                           tz = Sys.timezone())
mergedDataCopy$time = format(mergedDataCopy$date,
                              format = "%H:%M:%S")
mergedDataCopy$date = as.Date(mergedDataCopy$date,
                               format = "%m/%d/%Y")

sum(is.na(mergedDataCopy$date))
```

```
## [1] 0
```

```r
sum(is.na(mergedDataCopy$time))
```

```
## [1] 0
```

I created a data frame with 5 columns: id,date,value,totalMinutes,state.

The next classification is for values(v) and state(s).

- (v)1 is (s)asleep
- (v)2 is (s)restless
- (v)3 is (s)awake

Lastly, I sum all the values for each user on each day. With this table we'll be able to get interesting insights. Like all the minutes per state for each user.

- Now there's 2480 rows.

```r
groupedMergedData <- mergedDataCopy %>%
group_by(id,date,value) %>%
summarize(totalMinutes = n(), .groups = 'drop') %>%
mutate(state = case_when(
  value == 1 ~ "Asleep",
  value == 2 ~ "Restless",
  value == 3 ~ "Awake"
  ))
```

Analyze data:

The data merged has a total of 25 users.

The data contains id, date, value(each sleeping minutes),total minutes per state, state(split in 3 categories).

- asleep
- is restless
- is awake

```
n_distinct(groupedMergedData$id)
```

```
## [1] 25
```

Summary and Head function for groupedMergedData

```
summary(groupedMergedData)
```

```
##        id                  date                 value        totalMinutes
##  Min.   :1.504e+09   Min.   :2016-03-11   Min.   :1.000   Min.   :  1.0
##  1st Qu.:3.977e+09   1st Qu.:2016-03-27   1st Qu.:1.000   1st Qu.:  6.0
##  Median :4.703e+09   Median :2016-04-11   Median :2.000   Median : 26.0
##  Mean   :4.872e+09   Mean   :2016-04-10   Mean   :1.917   Mean   :154.3
##  3rd Qu.:6.776e+09   3rd Qu.:2016-04-26   3rd Qu.:3.000   3rd Qu.:356.0
##  Max.   :8.792e+09   Max.   :2016-05-12   Max.   :3.000   Max.   :791.0
##     state
##  Length:2480
##  Class :character
##  Mode  :character
##
##
##
```

```
head(groupedMergedData)
```

```
## # A tibble: 6 x 5
##            id date       value totalMinutes state
##         <dbl> <date>     <int>        <int> <chr>
## 1 1503960366 2016-03-13     1          411 Asleep
## 2 1503960366 2016-03-13     2           15 Restless
## 3 1503960366 2016-03-14     1          354 Asleep
## 4 1503960366 2016-03-14     2           27 Restless
## 5 1503960366 2016-03-14     3            5 Awake
## 6 1503960366 2016-03-15     1          312 Asleep
```

I created a data frame for days tracked per user. Also checked the quantity of days in the data set.

- I found that there's one extra day in some users.

Data set days:

  - 1 = 12/03/16 - 11/04/16
  - 2 = 12/04/16 - 12/05/16

- I arranged the data to find that is counting the day 11/03/2016. This day is also counted because users went to sleep before 00:00:00. That's the reason for the extra day counted.

```
daysTrackedPerUser <- groupedMergedData %>%
  group_by(id) %>%
  summarise(daysTracked = n_distinct(date), .groups = 'drop')

n_unique(groupedMergedData$date)
```

```
## [1] 63
```

```
head(arrange(mergedData, date),10)
```

```
## # A tibble: 10 x 4
##            id date                value       logId
##         <dbl> <dttm>              <int>       <dbl>
##  1 5577150313 2016-03-11 21:19:30     1 11109426118
##  2 5577150313 2016-03-11 21:20:30     1 11109426118
##  3 5577150313 2016-03-11 21:21:30     1 11109426118
##  4 5577150313 2016-03-11 21:22:30     1 11109426118
##  5 5577150313 2016-03-11 21:23:30     1 11109426118
##  6 5577150313 2016-03-11 21:24:30     1 11109426118
##  7 5577150313 2016-03-11 21:25:30     1 11109426118
##  8 5577150313 2016-03-11 21:26:30     1 11109426118
##  9 5577150313 2016-03-11 21:27:30     1 11109426118
## 10 5577150313 2016-03-11 21:28:30     1 11109426118
```

Find the percentage of whole case study users tracking sleep and The average of days tracked users have tracked data.

- The whole case study data has a total of 35 users as found previously in the Excel data activity.
- We have data from 25 users this means that 71% of the users tracked their sleep.
- We can see that tracked users have an average of 57% of days tracked.

```
totalWholeStudyUsers = 35
#Percentage of the case study users that tracked their sleep
n_distinct(groupedMergedData$id)/totalWholeStudyUsers*100
```

```
## [1] 71.42857
```

```
#Percentage of tracked days by tracked users
percentSleepDaysTracked <- mean(daysTrackedPerUser$daysTracked)/n_unique(groupedMergedData$date)*100
percentSleepDaysTracked
```

```
## [1] 57.20635
```

Checked the number of entries on each day to see it in a graph and see the trends.

- We have a range between 11 and 18 users per day and a mean of 14 users.
- There is lower number of entries on the first and last day but it's because in those days the number of entries counted the early sleeping time from day before or after. As mentioned before.
- Seems like there is no outliers.

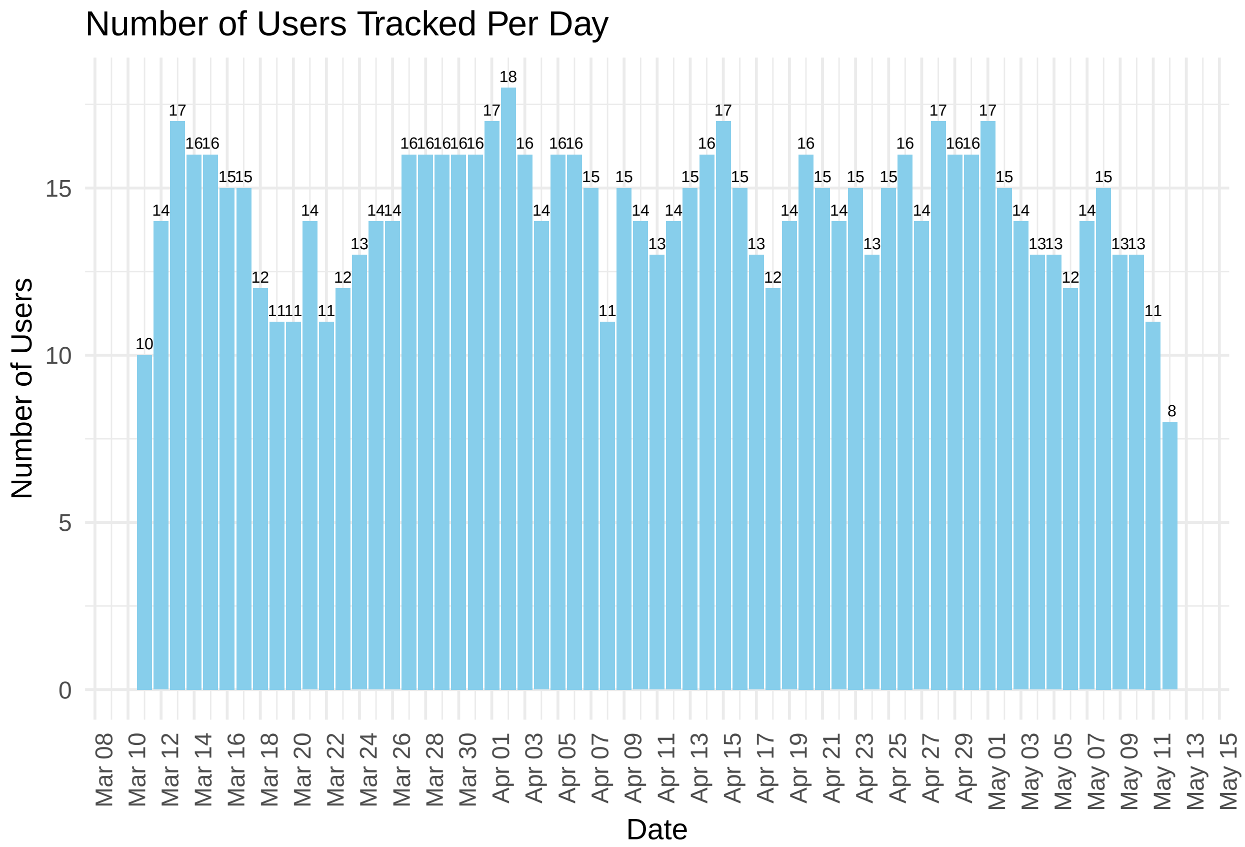
```
entriesPerDay <- groupedMergedData %>%
  group_by(date) %>%
  summarise(users = n_distinct(id), .groups = 'drop')

round(mean(entriesPerDay$users))
```

```
## [1] 14
```

```
EntriesPerDayPlot <- ggplot(entriesPerDay, aes(x = date, y = users)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(aes(label = format(users)), vjust = -0.5, size = 2) +
  scale_x_date(date_breaks = "2 day", date_labels = "%b %d") +
  labs(title = "Number of Users Tracked Per Day",
       x = "Date",
       y = "Number of Users") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

  print(EntriesPerDayPlot)
```



I create a data frame to get the average minutes per user in each state, to see users average sleeping minutes and hours.

- We can see that users sleep an average of 6,5 hours and around 30 minutes in restless state. It seems just below the recommended amount of sleep, 7-9 hours.

```
sleepAvgState <- groupedMergedData %>%
  group_by(state) %>%
  summarize(avgMinutes = mean(totalMinutes), avgHours = mean(totalMinutes)/60, .groups = 'drop')
```

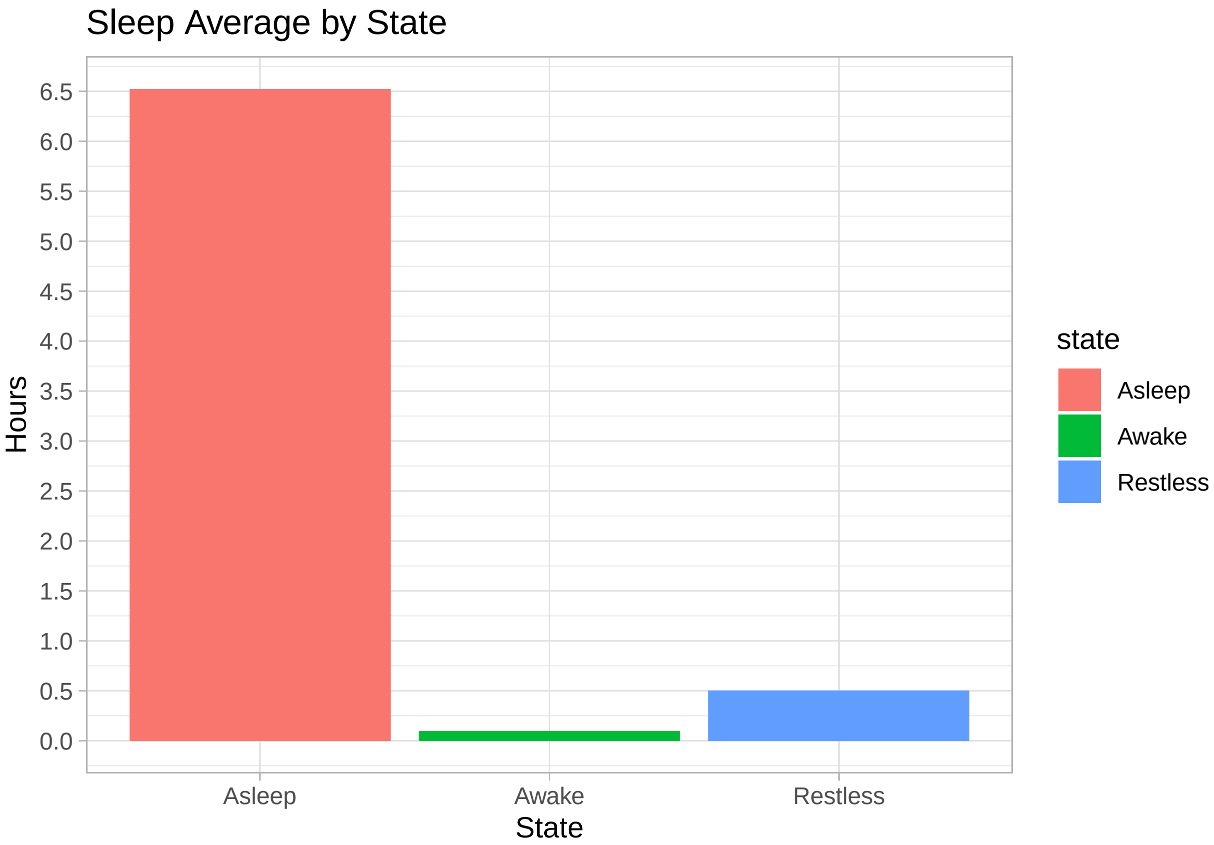
```
print(sleepAvgState)
```

```
## # A tibble: 3 x 3
##   state    avgMinutes avgHours
##   <chr>         <dbl>    <dbl>
## 1 Asleep       391.     6.52
## 2 Awake          5.79   0.0965
## 3 Restless      30.2    0.504
```

Sleep Average State Graph

```
sleepAverageStatePlot <- ggplot(data = sleepAvgState, aes(x = state, y= avgHours, fill = state)) +
  geom_col() +
  labs(title = "Sleep Average by State",
       x = "State",
       y = "Hours") +
  theme_light()+
  scale_y_continuous(breaks = seq(0, 10, by = 0.5))

print(sleepAverageStatePlot)
```



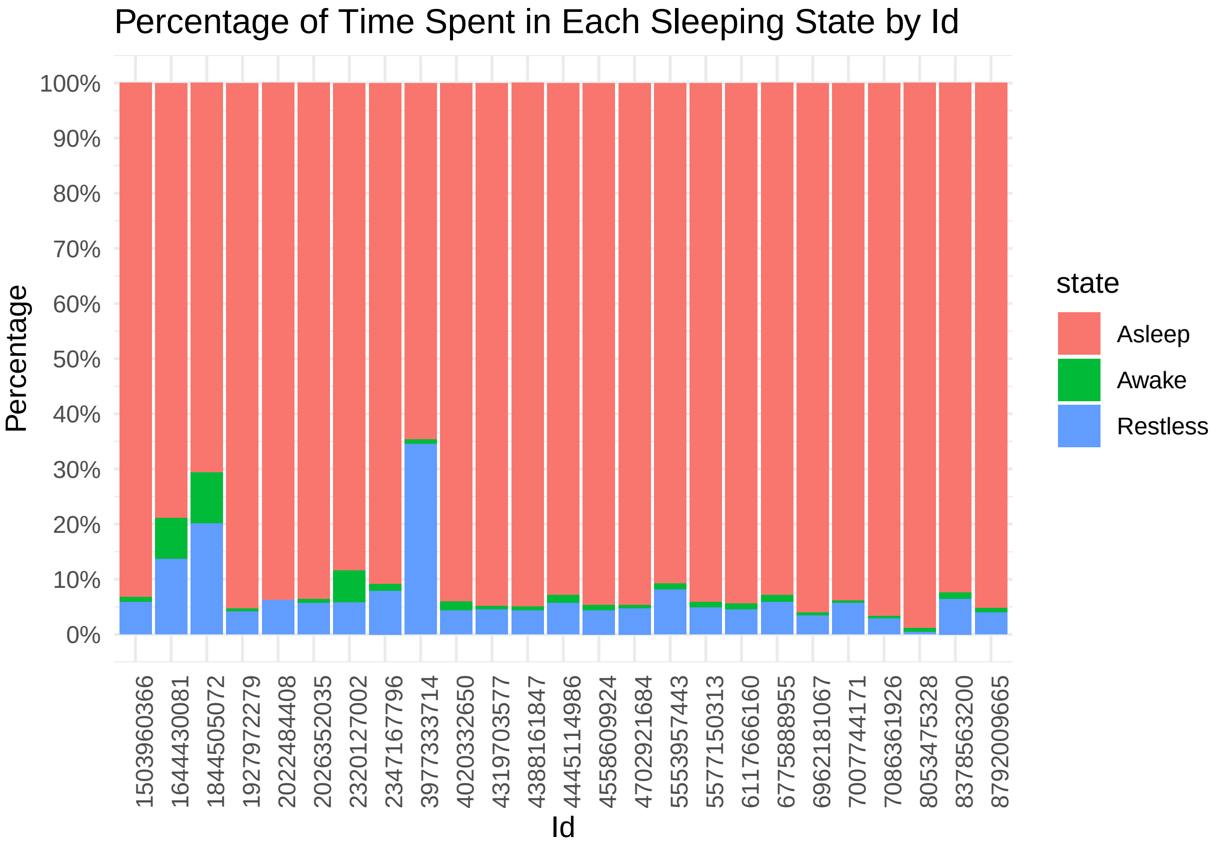
In this stacked bar graph you can find each of the tracked users with a percentage for each state.

In the graph we can see that 3 of the users are decreasing the "sleep" state but all the rest are around 90% of the sleeping time actually sleeping.

```
groupedMergedData$id <- as.character(groupedMergedData$id)
```

```
userSleepPlot <- ggplot(data = groupedMergedData, mapping = aes(x = id, y = totalMinutes, fill = state)
  geom_bar(stat = "identity", position = "fill") +
  labs(title = "Percentage of Time Spent in Each Sleeping State by Id",
       x = "Id",
       y = "Percentage") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  scale_y_continuous(labels = scales::percent,breaks = seq(0, 1, by = 0.1))

print(userSleepPlot)
```



Share:

1. In this part of the analysis with R, the merged "minute Sleep" CSV file we have a total of 25 user tracked. The whole case study data has a total of 35 users as found previously in the Excel data activity.

- This means that from the whole case study data 71% of the users tracked their sleep but at the same time we can see that from the 71% tracked users there is an average of 57% of days tracked.

2. We have a range between 11 and 18 users per day tracking their sleep with a mean of 14 user per day.
3. We can see that users sleep an average of 6,5 hours and around 30 minutes in restless state. It seems just below the recommended amount of sleep, 7-9 hours.

[National Heart,Lung and Blood Institute, An official website of the United States government] (https://www.nhlbi.nih.gov/health/sleep/how-much-sleep)

- We can see that 3 of the users have between 20%-30% of the sleeping time in restless and awake state. But all the rest are around 90% of the sleeping time in sleeping state.

- We can say that 3 of the user are suffering from some kind sleep disorder.