Final Project-M.Viveros

2023-10-31

suppressPackageStartupMessages({  
 suppressWarnings(library(tidyverse)) # for working with the data  
 suppressWarnings(library(lubridate)) # for working with datetime data  
 suppressWarnings(library(skimr)) # generate a text-based overview of the data  
 suppressWarnings(library(visdat)) # generate plots visualizing data types and missingness  
 suppressWarnings(library(plotly)) # generate interactive plots  
 suppressWarnings(library(readxl))  
 suppressWarnings(library(ggplot2)) # needed for the plot of eye metrics  
 suppressWarnings(library(dplyr))  
 suppressWarnings(library(tibble))  
})

# Formulate your question (Item 1-Research Question)

## By analyzing gaze data from 24 participants engaged in eight desktop activities, including time to first fixation (attention) and first fixation duration (initial object impression), this research will enrich our understanding of visual perception.

## *Here we’re specifically interested in: Can the application of first fixation metrics contribute to desktop activity recognition?*

## This isn’t very precise, but that’s okay: Part of the goal of this EDA is to clarify eye-metrics that contribute to understanding of visual perception.

file\_path <- "C:/Users/mv014/OneDrive - State Center Community College Distrct/Desktop/desktop/2022 Requests/UCM/R training/UcmFALL 2023/R2023/Final Project/desktopactivities all.xlsx"

# Get data and store data frames (Item 2-Read in your data)

## We’ll be using eye-data on students as they preform different desktop activities, available on Kaggle. The dataset comprises raw gaze coordinates (x-y) and timestamp data collected from 24 participants who were engaged in eight distinct desktop activities: Read, Browse, Play, Search, Watch, Write, Debug, and Interpret.

## *To get the download URL:*[*https://www.kaggle.com/datasets/namratasri01/eye-movement-data-set-for-desktop-activities*](https://www.kaggle.com/datasets/namratasri01/eye-movement-data-set-for-desktop-activities)*.*

sheet\_names <- excel\_sheets(file\_path) # Get the sheet names  
data\_frames <- list()# Initialize an empty list to store the data frames  
  
# Read each sheet and store the data frames in the list  
for (sheet in sheet\_names) {  
 data\_frames[[sheet]] <- read\_excel(file\_path, sheet = sheet)  
}  
  
  
combined\_data <- do.call(rbind, data\_frames) # Combine all data frames into one big data frame  
  
  
print(combined\_data) # Print the combined data

## # A tibble: 1,505,813 × 6  
## participant set activity x y timestamp  
## \* <chr> <chr> <chr> <dbl> <dbl> <dbl>  
## 1 P24 B WRITE 930 555 0  
## 2 P24 B WRITE 629 426 33  
## 3 P24 B WRITE 224 332 71  
## 4 P24 B WRITE 199 334 101  
## 5 P24 B WRITE 214 342 134  
## 6 P24 B WRITE 224 324 165  
## 7 P24 B WRITE 216 342 195  
## 8 P24 B WRITE 202 342 226  
## 9 P24 B WRITE 202 338 256  
## 10 P24 B WRITE 206 346 287  
## # ℹ 1,505,803 more rows

# Generate data summary (Item 3-Check the packaging)

## Peng and Matsui (2016) use some base R functions to look at dimensions of the dataframe and column (variable) types. Here we use the same from the checklist, for the EDA project.

skim(combined\_data) #3. Check the packaging ~How many rows and columns? Are there any types that might indicate parsing problems? NO

Data summary

|  |  |
| --- | --- |
| Name | combined\_data |
| Number of rows | 1505813 |
| Number of columns | 6 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 3 |
| numeric | 3 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| --- | --- | --- | --- | --- | --- | --- | --- |
| participant | 0 | 1 | 3 | 3 | 0 | 24 | 0 |
| set | 0 | 1 | 1 | 1 | 0 | 3 | 0 |
| activity | 0 | 1 | 4 | 9 | 0 | 8 | 0 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | 1 | 699.35 | 379.71 | -1858 | 352 | 728 | 971 | 3574 | ▁▂▇▁▁ |
| y | 0 | 1 | 445.96 | 217.85 | -1077 | 308 | 428 | 561 | 1917 | ▁▁▇▁▁ |
| timestamp | 0 | 1 | 153451.03 | 91141.00 | 0 | 75082 | 151728 | 228822 | 399184 | ▇▇▇▆▁ |

## How many rows and columns?

## *1505813 rows (observations); 6 columns (variables)*

## What variables are represented as different variable types? 3 variables are handled as characters, 3 as numeric. 24 participants. There are 3 sets, and 8 activities,

## The data includes rows with unique x, y, and timestamp values, with each row identified by the raw\_row\_number. The variables Participant, Set, and Activity serve as identifiers. Specifically, there are 24 unique participants, 3 unique sets, and 8 unique activities within the dataset. Notably, the Participant, Set, and Activity variables do not contain any missing values.Within the dataset, the variables x and y represent the spatial coordinates, while the timestamp indicates the specific time point when the gaze was at the corresponding x, y coordinate. The timestamp is instrumental in representing durations or the “time spent” on a particular activity. It’s worth noting that the variables representing activity and set also do not contain any missing values.

# For motivating question:

## *Good: eye gaze coordinates and timestamp is 100% complete*

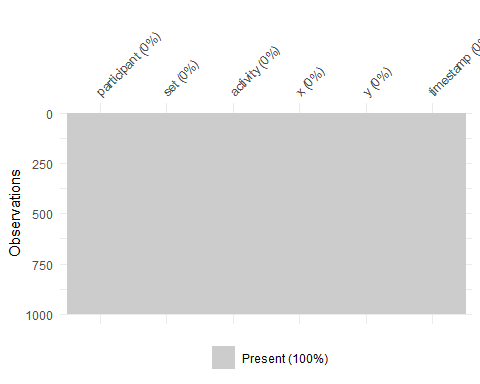
## *Also good: desktop activity is also 100% complete*

## *Potentially worrisome: complicated data in x, y and timestamp, dimensions of desktop not clear as we are only provided the desktop monitor size, we presume 1080 resolution.*

## Missing Values. We can’t use ‘vis\_miss(combined\_data)’ because of large data which causes and error. So we’ll use ‘sample\_n’ to draw a subset

## Arguments in vis\_miss() are useful for picking up patterns in missing values. Here we see there is no missing data.

set.seed(123)  
dataf\_smol = sample\_n(combined\_data, 1000) #sample to big so we'll draw a subset #but no missing data   
  
vis\_miss(dataf\_smol)



combined\_smol = sample\_n(combined\_data, 1000) #subset of combined data

# A Critical Examination

## *Can we identify which activities require more prolonged or shorter periods of visual engagement?*

## By examining the mean duration of the first timestamp for each activity, we can discern patterns related to the amount of time spent on each specific activity. This can help identify which activities require more prolonged or shorter periods of visual engagement.

combined\_data %>%  
 filter(!is.na(activity)) %>%  
 arrange(timestamp) %>%  
 group\_by(activity) %>%  
 summarise(mean\_duration = mean(timestamp - first(timestamp)), #This calculates the mean duration for each group (activity) using the difference between the timestamp and the first timestamp value within each group. It provides the average duration of time spent on the initial instance of each activity.  
 min\_x = first(x),   
 min\_y = first(y))

## # A tibble: 8 × 4  
## activity mean\_duration min\_x min\_y  
## <chr> <dbl> <dbl> <dbl>  
## 1 BROWSE 151700. 32 1131  
## 2 DEBUG 142959. 936 565  
## 3 INTERPRET 143692. 912 499  
## 4 PLAY 148100. 1012 604  
## 5 READ 148165. 863 157  
## 6 SEARCH 151359. 921 200  
## 7 WATCH 182171. 195 1092  
## 8 WRITE 145069. 930 555

# Interactive data exploration (Item 4-Look at the top and the bottom of your data)

## With over 1 million rows, the dataframe is too large to print in a readable way. Instead we’ll use the base R function View() in an interactive session. An Excel-like spreadsheet presentation View() can cause significant problems if you use it with a large dataframe on a slower machine, we’ll use a pipe. Overall, this code block is designed to provide an interactive way to examine the top and bottom rows of the combined\_data and combined\_smol data frames, allowing for easy inspection and understanding of the data structure and contents.

if (interactive()) {  
 combined\_data |>   
 head() |>   
 View()  
   
 combined\_data |>   
 tail() |>   
 View()  
   
 View(combined\_smol) #4 Check Top and Bottom  
}

# Some of my observations:

## We use skimr to check data quality by looking at the minimum and maximum values. All of the ranges make sense for what we expect the variable to be.

skim(combined\_data)

Data summary

|  |  |
| --- | --- |
| Name | combined\_data |
| Number of rows | 1505813 |
| Number of columns | 6 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 3 |
| numeric | 3 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| --- | --- | --- | --- | --- | --- | --- | --- |
| participant | 0 | 1 | 3 | 3 | 0 | 24 | 0 |
| set | 0 | 1 | 1 | 1 | 0 | 3 | 0 |
| activity | 0 | 1 | 4 | 9 | 0 | 8 | 0 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | 1 | 699.35 | 379.71 | -1858 | 352 | 728 | 971 | 3574 | ▁▂▇▁▁ |
| y | 0 | 1 | 445.96 | 217.85 | -1077 | 308 | 428 | 561 | 1917 | ▁▁▇▁▁ |
| timestamp | 0 | 1 | 153451.03 | 91141.00 | 0 | 75082 | 151728 | 228822 | 399184 | ▇▇▇▆▁ |

# Data Summary (Item 5-Check your “n”s)

## Since there is not another duplicate set or typical data set for desktop activities and eye-metric times, we cannot check against n’s, as the total expected are in the dataframe. So here we are checking the sample to see if timescale for one activity matches with the dataset. In this example, by downloading the dataset from the kaggle website, you can see the first, ‘x’ and ‘y’ coordinates in ‘write’ activity for participant no.24, assigned in set ‘B’ match with the sample below.

# Given table  
data <- tibble::tribble(  
 ~participant, ~set, ~activity, ~x, ~y, ~timestamp, #plot of head  
 "P24", "B", "WRITE", 930, 555, 0,  
 "P24", "B", "WRITE", 629, 426, 33,  
 "P24", "B", "WRITE", 224, 332, 71,  
 "P24", "B", "WRITE", 199, 334, 101,  
 "P24", "B", "WRITE", 214, 342, 134,  
 "P24", "B", "WRITE", 224, 324, 16  
)

# Time stamps for activities (Item 6-Validate with at least one external data source).

## A web search leads us to the website for the article where the data set is used: [*https://www.researchgate.net/publication/329955224\_Combining\_Low\_and\_Mid-Level\_Gaze\_Features\_for\_Desktop\_Activity\_Recognition*](https://www.researchgate.net/publication/329955224_Combining_Low_and_Mid-Level_Gaze_Features_for_Desktop_Activity_Recognition) *and published.*

## In the methods section, the authors detail that all of the activities last about 5-6 minutes. Therefore the timestamps for the coordinates for each activity for all participants should not exceed this time, and indeed the max duration for one activity in going through all the values is ~6 minutes (399184 ms).

combined\_data %>%  
 filter(!is.na(activity)) %>%  
 arrange(timestamp) %>%  
 group\_by(activity, participant) %>%  
 summarise(  
 max\_duration = max(timestamp - first(timestamp)),  
 min\_x = first(x),  
 min\_y = first(y)  
 ) %>%  
 arrange(desc(max\_duration))#By adding arrange(desc(max\_duration)) at the end of the pipeline, the data will be sorted in descending order based on the max\_duration column, showing the highest amounts of max\_duration first.

## `summarise()` has grouped output by 'activity'. You can override using the  
## `.groups` argument.

## # A tibble: 192 × 5  
## # Groups: activity [8]  
## activity participant max\_duration min\_x min\_y  
## <chr> <chr> <dbl> <dbl> <dbl>  
## 1 WATCH P24 399184 195 1092  
## 2 WATCH P07 396323 538 954  
## 3 WATCH P17 394897 527 325  
## 4 WATCH P01 394598 540 1067  
## 5 WATCH P03 394594 940 974  
## 6 WATCH P19 389977 188 1056  
## 7 WATCH P10 389362 720 1118  
## 8 WATCH P11 388227 691 436  
## 9 WATCH P15 382229 680 1065  
## 10 WATCH P08 379463 1098 56  
## # ℹ 182 more rows