Final Project-M.Viveros

2023-11-1

```
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). There are 192 individual files, so we need to do some data wrangling and combine them into one dataframe for analysis.

```
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 p
roblems? 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

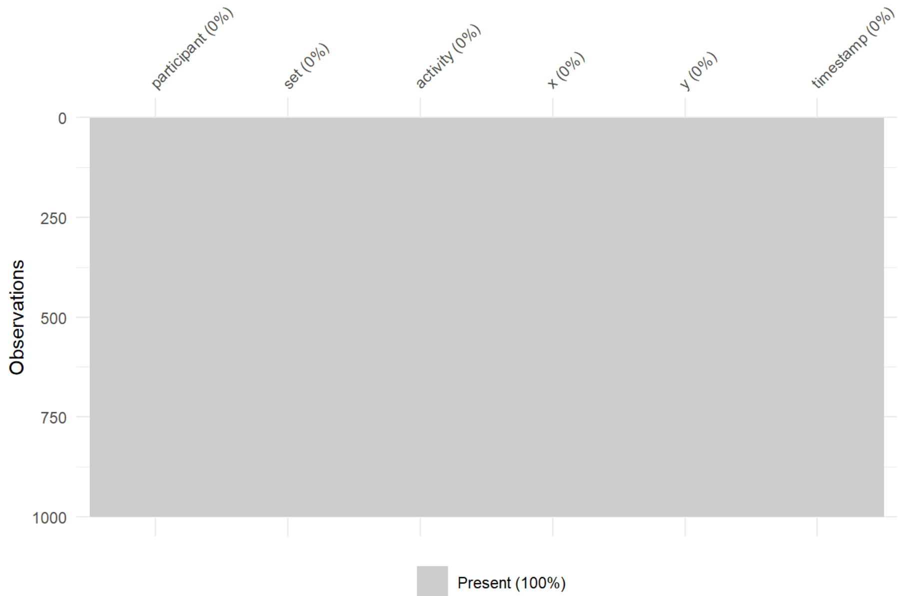
Potentially worrisome: complicated data in x, y and timestamp, dimensions of desktop not clear as we are only provided the desktop monitor size, we presume 1080 resolution.

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_Level_Gaze_Features_for_Desktop_Activity_Recognition (https://www.researchgate.net/publication/329955224_Combining_Low_and 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 d
escending 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.
```

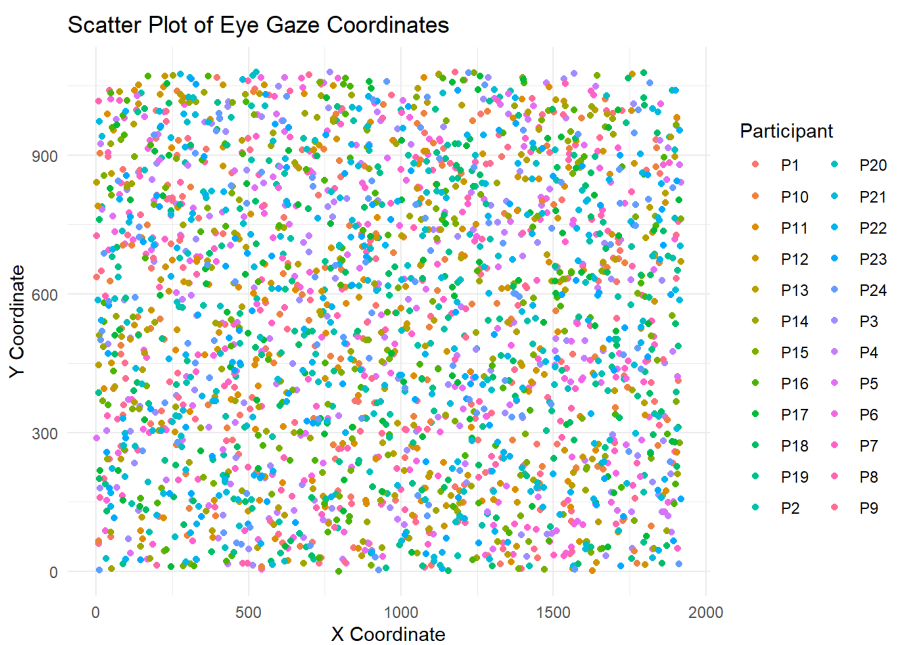
```
## # A tibble: 192 × 5
## # Groups:   activity [8]
##    activity participant max_duration min_x min_y
##    <chr>    <chr>              <dbl> <dbl> <dbl>
##  1 WATCH    P24               399184   195  1092
##  2 WATCH    P07               396323   538   954
##  3 WATCH    P17               394897   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
```

Create scatter plot with x, y, and timestamp-(Item 7-Make a plot)

```
# Sample data creation
set.seed(123) # for reproducibility
n_points <- 100 # Number of data points per participant
n_participants <- 24

# Generate sample data for 24 participants
data <- tibble(
  participant = rep(paste0("P", 1:n_participants), each = n_points),
  x = runif(n_points * n_participants, min = 0, max = 1920), # Adjust max according to your screen resolution resolution of
desktop used on 24 inch monitor, detailed in methods
  y = runif(n_points * n_participants, min = 0, max = 1080), # Adjust max according to your screen resolution, resolution of
desktop used on 24 inch monitor
  timestamp = runif(n_points * n_participants, min = 0, max = 399184) # Random timestamps within the last 30 days
)

# Create scatter plot with x, y, and timestamp
ggplot(data, aes(x = x, y = y, color = participant)) +
  geom_point() +
  labs(title = "Scatter Plot of Eye Gaze Coordinates",
       x = "X Coordinate", y = "Y Coordinate", color = "Participant") +
  theme_minimal()
```



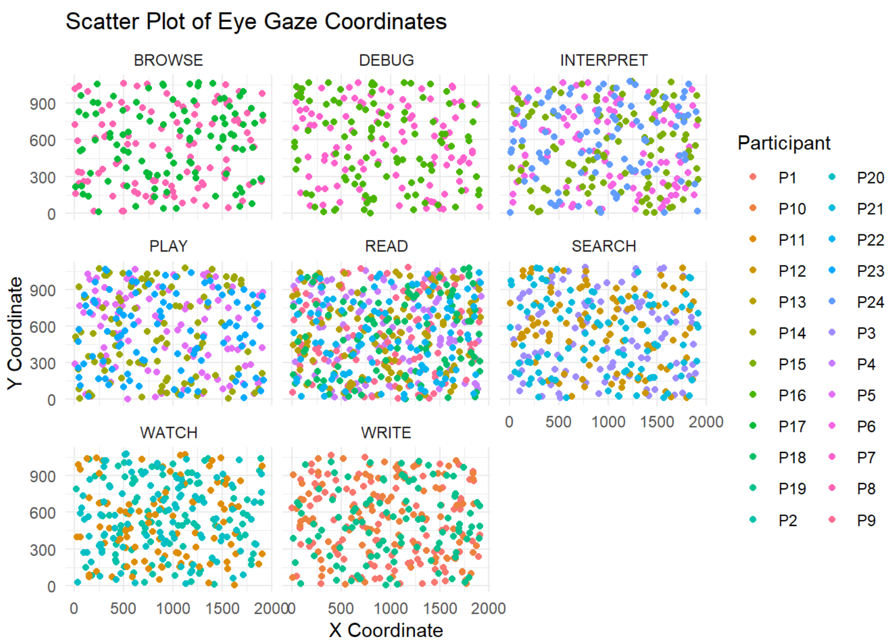
A plot that tells us location for each activity.(Item 8-Try the easy solution first)

Let’s translate our natural-language research question into: Whether all participants stare in relatively the same positions across the activities. The easy solution is to estimate location by plotting the x, and y coordiantes for each participant across all activities.

```
# Sample data creation
set.seed(123) # for reproducibility
n_points <- 100 # Number of data points per participant
n_participants <- 24

# Generate sample data for 24 participants
data <- tibble(
  participant = rep(paste0("P", 1:n_participants), each = n_points),
  x = runif(n_points * n_participants, min = 0, max = 1920),
  y = runif(n_points * n_participants, min = 0, max = 1080),
  timestamp = runif(n_points * n_participants, min = 0, max = 399184),
  activity = rep(c("WRITE", "WATCH", "SEARCH", "READ", "PLAY", "INTERPRET", "DEBUG", "BROWSE", "READ"), each = n_points, len
gth.out = n_points * n_participants)
)

# Create scatter plot with x, y, and timestamp
ggplot(data, aes(x = x, y = y, color = participant)) +
  geom_point() +
  labs(title = "Scatter Plot of Eye Gaze Coordinates",
       x = "X Coordinate", y = "Y Coordinate", color = "Participant") +
  theme_minimal() +
  facet_wrap(~ activity)
```



Only the row with the minimum timestamp (i.e., the first fixation) is provided below by each activity. (Item 9-Results)

The preliminary results show, the first fixation across desktop actiivites is different but most participants look in the same area.

Discussion.The resulting graph effectively demonstrates the relationships between the ‘activity’ and ‘participant’ variables. It plots the participants on the x-axis and utilizes facets to categorize the data points based on different activities. This facet arrangement allows for a clear comparison of the gaze patterns among the participants for each specific activity.Through this visual representation, we can observe distinct variations in the participants' gaze patterns across different activities. The clear differentiation in the plotted data points suggests potential differences in how participants engage with various activities, providing valuable insights into their cognitive processes and task engagement strategies during the experiment or study.

```
# Sample data creation
set.seed(123) # for reproducibility
n_points <- 2000 # Number of data points per participant
n_participants <- 24

# Generate sample data for 24 participants
data <- tibble(
  participant = rep(paste0("P", 1:n_participants), each = n_points),
  x = runif(n_points * n_participants, min = 0, max = 1920),
  y = runif(n_points * n_participants, min = 0, max = 1080),
  timestamp = runif(n_points * n_participants, min = 0, max = 399184),
  activity = rep(c("WRITE", "WATCH", "SEARCH", "READ", "PLAY", "INTERPRET", "DEBUG", "BROWSE", "READ"), each = n_points, len
gth.out = n_points * n_participants)
)

# Select only the first fixation's x and y coordinates for each participant
first_fixations <- data %>%
  group_by(participant) %>%
  slice_min(timestamp) %>%
  ungroup()

# Create scatter plot with the first fixation x, and y coordinates
ggplot(first_fixations, aes(x = x, y = y, color = participant)) +
  geom_point() +
  labs(title = "Scatter Plot of First Fixation Eye Gaze Coordinates",
       x = "X Coordinate", y = "Y Coordinate", color = "Participant") +
  theme_minimal() +
  facet_wrap(~ activity)
```