Examining Writing Behaviors: An Analysis of Logged Typing Data from Online Essay Submissions

Caleb Walls

INFX 502

11/28/2023

Dataset

A. Dataset Description and Purpose

The selected dataset is comprised of action logs collected from an online writing assignment conducted by Vanderbilt University. It was uploaded to *kaggle.com* as part of a competition that Vanderbilt is currently hosting called "Linking Writing Process to Writing Quality"[1]. The dataset originally contained about 8.4 million observations of 11 variables logged from the essays of roughly 2500 individuals.

The objective of the competition is to develop a machine learning model that can accurately predict student grades on an essay based solely on the features of their writing process that can be observed from a log of their actions and keystrokes. To achieve this, the hosts provided a dataset containing information about each action such as the type of key pressed, the time, the duration, the location of the cursor within the textbox, and the overall word count. To ensure that models can only train on writing behavior rather than the resulting essay itself, all specific letter keys were anonymized in the logs by replacing them with the letter "q".

The organizers of the competition hope that the submitted models will lead to insights that can be used in the development of writing instruction, automated writing evaluation systems, and AI-based tutoring software. As a former teacher, my own interests in these topics have motivated me to analyze this dataset more closely and participate in discussions with model builders involved in this competition.

B. Dataset Collection Procedures

According to the information provided by Vanderbilt University, the typing logs were collected through a simple website built specifically for this project. Participants were recruited through a platform known as "Amazon Mechanical Turk" and were paid to complete this writing task.

Once on the project page, participants were given a prompt and asked to write an argumentative essay in response. The participants were also given success criteria stating that their essay should be a minimum of 200 words in at least 3 paragraphs, and they were directed not to leave the page during the task. In addition, they were told to complete the task within 30 minutes and a timer was provided at the top of the page to let them know the amount of time remaining. A warning at the bottom of the page also let them know that they would not receive payment if they engaged in plagiarism or failed to follow the instructions provided.

A keystroke logging program was embedded in the JavaScript of the project webpage. It used event listeners to log information every time a key was pressed, or the mouse was clicked. The specific data points collected from each event are described in the following sections.

C. Loading the Dataset

The dataset consisted of two main files, which were downloaded directly from *kaggle.com* as a zipped folder. I extracted the contents then used the following R commands to set that folder as my working directory, read the csv files, and store each file as a data frame.

```
> setwd("C:\\Users\\caleb\\OneDrive\\Desktop\\linking-writing-processes-to-writing-quality")
> train_logs <- read.csv("train_logs.csv")
> train_scores <- read.csv("train_scores.csv")
```

The **head()** function was then used to examine the first twenty rows of the *train_logs* data frame. It contained action logs of recorded data about each of the participants' keystrokes as they typed their essays. Afterwords, I used the **str()** function to better understand the variables that were recorded in each column.

```
> head(train_logs, n = 20)
         id event_id down_time up_time action_time      activity down_event  up_event text_change cursor_position word_count
1  001519c8        1      4526    4557          31 Nonproduction  Leftclick Leftclick    NoChange               0          0
2  001519c8        2      4558    4962         404 Nonproduction  Leftclick Leftclick    NoChange               0          0
3  001519c8        3    106571  106571           0 Nonproduction      Shift     Shift    NoChange               0          0
4  001519c8        4    106686  106777          91         Input          q         q           q               1          1
5  001519c8        5    107196  107323         127         Input          q         q           q               2          1
6  001519c8        6    107296  107400         104         Input          q         q           q               3          1
7  001519c8        7    107469  107596         127         Input          q         q           q               4          1
8  001519c8        8    107659  107766         107         Input          q         q           q               5          1
9  001519c8        9    107743  107852         109         Input          q         q           q               6          1
10 001519c8       10    107840  107978         138         Input      Space     Space                           7          1
11 001519c8       11    108008  108195         187         Input          q         q           q               8          2
12 001519c8       12    108104  108259         155         Input          q         q           q               9          2
13 001519c8       13    108229  108370         141         Input          q         q           q              10          2
14 001519c8       14    108341  108486         145         Input      Space     Space                          11          2
15 001519c8       15    109296  109438         142         Input          q         q           q              12          3
16 001519c8       16    109423  109559         136         Input          q         q           q              13          3
17 001519c8       17    109560  109729         169         Input          q         q           q              14          3
18 001519c8       18    109826  109994         168         Input          q         q           q              15          3
19 001519c8       19    110398  110516         118    Remove/Cut  Backspace Backspace           q              14          3
20 001519c8       20    110595  110751         156         Input          q         q           q              15          3
> str(train_logs)
'data.frame':   8405898 obs. of  11 variables:
 $ id             : chr  "001519c8" "001519c8" "001519c8" "001519c8" ...
 $ event_id       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ down_time      : int  4526 4558 106571 106686 107196 107296 107469 107659 107743 107840 ...
 $ up_time        : int  4557 4962 106571 106777 107323 107400 107596 107766 107852 107978 ...
 $ action_time    : int  31 404 0 91 127 104 127 107 109 138 ...
 $ activity       : chr  "Nonproduction" "Nonproduction" "Nonproduction" "Input" ...
 $ down_event     : chr  "Leftclick" "Leftclick" "Shift" "q" ...
 $ up_event       : chr  "Leftclick" "Leftclick" "Shift" "q" ...
 $ text_change    : chr  "NoChange" "NoChange" "NoChange" "q" ...
 $ cursor_position: int  0 0 0 1 2 3 4 5 6 7 ...
 $ word_count     : int  0 0 0 1 1 1 1 1 1 1 ...
```

I also examined the *train_scores* data frame with the **head()** and **str()** functions, which are displayed on the next page. This file was much smaller, because it only contained the final score that each participant received on their essay. As you can see, the individual essays are listed in both data frames by a unique identifier listed in the *id* column.

```
head(train_scores, n = 10)
       id score
  001519c8   3.5
  0022f953   3.5
  0042269b   6.0
  0059420b   2.0
  0075873a   4.0
  0081af50   2.0
  0093f095   4.5
  009e23ab   4.0
  00e048f1   3.5
0 00e1f05a   4.5
```

```
> str(train_scores)
'data.frame':   2471 obs. of  2 variables:
 $ id    : chr  "001519c8" "0022f953" "0042269b" "0059420b" ..
 $ score: num  3.5 3.5 6 2 4 2 4.5 4 3.5 4.5 ...
```

D. Variables Names and Descriptions

The charts below provide a more detailed description of each variable based on the information provided by Vanderbilt University in the competition description on *kaggle.com*. The **str()** outputs showed that some variable data types will also need to be changed before beginning the analysis. Additionally, if the *up_event* and *down_event* variables contain the exact same information as expected, then these will be combined into a single *event* variable during the preprocessing. Those changes have been highlighted in yellow within the charts below.

Column Name	Data Type	Variable Description
id	Character	A unique identifier for each essay submission (Corresponds to *id* in the *train_scores* data frame)
event_id	Numeric	A chronological index of each event that occurred during the essay writing process
down_time	Numeric	A timestamp recording the initiation of a down event caused by pressing a key / mouse button (in milliseconds)
up_time	Numeric	A timestamp recording the completion of an up event caused by releasing a key / mouse button (in milliseconds)
action_time	Numeric	The duration of the event from *down_time* to *up_time* (in milliseconds)
activity	Factor	The type or category of activity accomplished by the event
event	Factor	The name of the recorded keystroke / mouse event
text_change	Character	The alteration that occurred in the text as a result of the recorded event
cursor_position	Numeric	A character index used to record the location of the text cursor within the essay at the time of event completion
word_count	Numeric	A count of words in the essay at the time of event completion.

As stated previously, the *train_scores* data frame contains only two columns and lists a score for each participant based on their unique identifiers from the *id* column of the first data frame.

Column Name	Data Type	Variable Description
id	Character	A unique identifier for each essay submission (Corresponds to *id* in the *train_logs* data frame)
score	Factor	The score assigned to the essay (on a scale of 0 to 6)

E. Expectations

This dataset was compiled by Vanderbilt University specifically for the purpose of understanding the correlation between typing behavior and writing quality. Insights derived from this dataset could prove very important for the development of more effective writing instruction, but it could also be used by developers to build better tools for real-time assessment and intervention. To support the production of valid predictive models that can yield those sorts of insights, *score* will be considered the dependent variable throughout the following analysis.

The goal of this analysis is to determine which variables in the dataset are most useful for predicting a participant's score. Univariate analysis will allow us to better understand the distributions of each variable, and bivariate analysis will examine the degree of correlation between two or more variables in the dataset. Correlation matrices and other visualizations will also be used to understand the relationship between variables.

The amount of raw data in this dataset may pose certain challenges when analyzing the action logs. Even though only 2471 participants were involved in the study, their logged actions totaled up to almost 8.5 million rows of observations. While this is ultimately good for building generalizable models, it does make cleaning, analysis, and visualization more difficult. In addition, the nature of these action logs as time-series data also makes some variables more challenging to analyze. More advanced methods of analysis and modeling might produce better results, but that is beyond the scope of this report.

Despite the complexity of the dataset, it may be possible to derive some simple metrics from the raw data that can be used as predictors. To support the effectiveness of more simple modeling methods, this analysis will focus on single values that can be calculated from the action logs and added to the *train_scores* data frame. While values like final word count and completion time can be easily derived from the raw data and will be simple to understand, other metrics like kurtosis of cursor position, though less straightforward, could capture insights into complex behaviors like high-level editing occurring throughout the paragraph. Ultimately, examination of simple metrics like these may have more explanatory power than complex models for explaining the connections between typing behavior and writing quality, even if those models are more accurate.

Analyses

A. Data Preprocessing

Before beginning a full analysis, the data will need to be cleaned to resolve any data entry errors, missing values, and unnecessary noise. During this process we will also change the data type of several variables to "Factor" as indicated in Section D.

To begin, both data frames were checked for missing values. This was accomplished by viewing the data frames in RStudio's built-in viewer and sorting each row individually to see the highest and lowest values. Initially, one column seemed to have missing values since many cells in the *text_change* column appeared blank. On closer inspection, however, they were found to contain spaces, which showed up quite frequently in the keystroke logs. No actual cells in that column were completely blank. This was confirmed for both data frames by calling the following functions:

```
> any(is.na(train_logs))
[1] FALSE
> any(train_logs == "")
[1] FALSE
```

```
> any(is.na(train_scores))
[1] FALSE
> any(train_scores == "")
[1] FALSE
```

A.1 Identifying Data Issues

Sorting each column as described above also allowed me to check if the numeric columns contained any obvious errors. While some columns had larger ranges than expected, no other major issues were observed. Once each numeric variable is analyzed more thoroughly, then a determination will be made about how to deal with outliers. This data is intended to be used for predictive modeling, however, so its distribution likely reflects the variability that exists within the remaining observations of the test set.

The **unique()** function was then used to examine the potential values of each categorical variable more closely. This process revealed issues with the *activity* column and inconsistencies between the *down_event* and *up_event* columns that would need to be resolved before they could be converted into factors. Before proceeding with those changes, the data type of the *score* column was updated to "Factor" since it did not contain any unexpected values. The **as.factor()** function was used to accomplish this as shown below:

```
> train_scores$score ← as.factor(train_scores$score)
> str(train_scores$score)
 Factor w/ 12 levels "0.5","1","1.5",..: 7 7 12 4 8 4 9 8 7 9 ...
```

A.2 Addressing "Activity" Variable Issues

As seen from the output below, the *activity* variable contained multiple variations of the activity type "move". Each observation of this activity included index values that represented the starting and ending positions of the text before and after it was moved. This activity was recorded very rarely and only occurred when a text was highlighted and dragged to a new position within the

essay. Removing the extra location information would simplify the "move" activity and allow it to be stored similarly to the other values in that column.

```
> unique(train_logs$activity)
 [1] "Nonproduction"                          "Input"                                  "Remove/Cut"
 [4] "Replace"                                "Move From [284, 292] To [282, 290]"     "Move From [287, 289] To [285, 287]"
 [7] "Move From [460, 461] To [465, 466]"     "Paste"                                  "Move From [905, 1314] To [907, 1316]"
[10] "Move From [565, 743] To [669, 847]"     "Move From [669, 847] To [565, 743]"     "Move From [1041, 1121] To [1496, 1576]"
[13] "Move From [1455, 1557] To [1323, 1425]" "Move From [2268, 2275] To [2247, 2254]" "Move From [213, 302] To [902, 991]"
[16] "Move From [0, 158] To [234, 392]"       "Move From [460, 465] To [925, 930]"     "Move From [810, 906] To [816, 912]"
[19] "Move From [186, 187] To [184, 185]"     "Move From [140, 272] To [299, 431]"     "Move From [114, 140] To [272, 298]"
[22] "Move From [1386, 1450] To [1445, 1509]" "Move From [442, 524] To [296, 378]"     "Move From [408, 414] To [390, 396]"
[25] "Move From [1144, 1147] To [1142, 1145]" "Move From [218, 220] To [206, 208]"     "Move From [164, 165] To [153, 154]"
[28] "Move From [623, 632] To [624, 633]"     "Move From [747, 960] To [1041, 1254]"   "Move From [274, 314] To [299, 339]"
[31] "Move From [624, 625] To [845, 846]"     "Move From [1861, 2063] To [1766, 1968]" "Move From [1766, 1968] To [1861, 2063]"
[34] "Move From [2091, 2179] To [252, 340]"   "Move From [923, 1077] To [340, 494]"    "Move From [0, 1] To [590, 591]"
[37] "Move From [999, 1000] To [1000, 1001]"  "Move From [13, 65] To [9, 61]"          "Move From [1651, 1769] To [1565, 1683]"
[40] "Move From [61, 136] To [0, 75]"         "Move From [0, 75] To [1, 76]"           "Move From [75, 134] To [304, 363]"
[43] "Move From [289, 355] To [562, 628]"     "Move From [944, 1102] To [1050, 1208]"  "Move From [1306, 1371] To [1061, 1126]"
[46] "Move From [1061, 1126] To [1306, 1371]" "Move From [1361, 1362] To [1358, 1359]" "Move From [51, 86] To [109, 144]"
[49] "Move From [134, 169] To [122, 157]"     "Move From [382, 437] To [458, 513]"
```

After considering the order of events that would lead to a "move" activity, it was determined that each "move" event was preceded by a separately recorded click that highlighted a portion of text. Since the first set of indices represented the original position of the text, it would be more appropriate to store that information as a highlighting change resulting from the initial click. The loop below was written to change the initial click's *text_change* value from "NoChange" to "Highlighted [x, y]" and to remove the remaining "move" activity information since the final location of the text can be inferred from the resulting *cursor_position*.

```
i ← 1
while (i ≤ nrow(train_logs)) {
  if (substr(train_logs$activity[i+1], 1, 1) == "M" & train_logs$down_event[i] == "Leftclick"
& train_logs$text_change[i] == "NoChange") {

    new_text ← sub(".*?(\\[.*?\\]).*", "\\1", train_logs$activity[i + 1])
    train_logs$text_change[i] = paste0("Highlighted ", new_text)
    train_logs$activity[i+1] ← "Move"
    rm(new_text)

  }
  i ← i + 1
}
```

An example of the changes to the data frame can be seen below. The **unique()** function was also used again to confirm that the loop worked as intended. Since it had, the **as.factor()** function was used to change the data type of the *activity* column to "Factor".

activity	down_event	up_event	text_change
Nonproduction	Leftclick	Leftclick	Highlighted [284, 292]
Move	Leftclick	Leftclick	qqqqqqq

```
> unique(train_logs$activity)
[1] "Nonproduction" "Input"          "Remove/Cut"    "Replace"        "Move"
[6] "Paste"
> train_logs$activity ← as.factor(train_logs$activity)
> str(train_logs$activity)
 Factor w/ 6 levels "Input","Move",..: 3 3 3 1 1 1 1 1 1 1 ...
```

A.3 Addressing "Event" Variable Issues

Before combining the *down_event* and *up_event* columns, any inconsistencies would need to be resolved. Since these two variables refer to the same event, they should always have the same value. The subsetting function below determined that 534 rows did contain inconsistencies between these two values. The subsetted data frame was then viewed to determine why these inconsistencies might have occurred.

```
> nrow(train_logs[train_logs$down_event ≠ train_logs$up_event, ])
[1] 534
> View(train_logs[train_logs$down_event ≠ train_logs$up_event, ])
```

Most rows that contained inconsistencies were labelled as "Replace" activities. The *text_change* column for these activities showed that a group of highlighted text was replaced by a single letter. The anonymizing function used by the host to relabel all typed letters as "q" converted the *down_event* values but skipped over the *up_event* values in error. A handful of rows were for "Input" activities, but these records also showed that the *down_event* values had been correctly anonymized and the *up_event* column contained errors. Since all inconsistencies could be accounted for by errors in the *up_event* column, that column was removed from the data frame and the *down_event* column was renamed *event*. This was achieved using the functions below.

```
> train_logs ← train_logs[, names(train_logs) ≠ "up_event"]
> names(train_logs)[names(train_logs) == "down_event"] ← "event"
```

With the events now consolidated into one column, the **unique()** function was used to determine how many unique values existed within that column. A frequency table was also generated using the **table()** function to better understand the distribution of these values. As seen from the output below, many different values were recorded within this column.

```
> length(unique(train_logs$event))
[1] 131
> print(unique(train_logs$event))
  [1] "Leftclick"          "Shift"          "q"              "Space"
  [5] "Backspace"          "."              "'"              "Enter"
  [9] "ArrowLeft"          "-"              ","              "ArrowRight"
 [13] "-"                  "?"              "Tab"            "\""
 [17] "ArrowUp"            "ArrowDown"      "Rightclick"     "="
 [21] "CapsLock"           "Control"        "c"              "v"
 [25] "/"                  "Delete"         ":"              "z"
 [29] "["                  "$"              "("              ")"
 [33] "+"                  "Home"           "End"            "\\"
```

```
 [37] "Meta"            "*"              "&"              "AudioVolumeMute"
 [41] "x"               "!"              "Insert"         "MediaPlayPause"
 [45] "NumLock"         "%"              "V"              ">"
 [49] "Alt"             "AudioVolumeUp"  "ContextMenu"    "AudioVolumeDown"
 [53] "a"               "<"              "PageDown"       "]"
 [57] "Middleclick"     "@"              "F12"            "j"
 [61] "\u0096"          "Dead"           "t"              "s"
 [65] "n"               "y"              "{"              "ScrollLock"
 [69] "¿"               "Process"        "}"              "MediaTrackPrevious"
 [73] "MediaTrackNext"  "F3"             "^"              "Unidentified"
 [77] "Cancel"          "2"              "i"              "d"
 [81] "r"               "e"              "`"              "\u009b"
 [85] "m"               "#"              "~"              "PageUp"
 [89] "T"               "A"              "b"              "S"
 [93] "ModeChange"      "_"              "Escape"         "F11"
 [97] "Unknownclick"    "AltGraph"       "F10"            "h"
[101] "F15"             "Clear"          "OS"             "F"
[105] "C"               "o"              "Ä±"             "f"
[109] "u"               "w"              "p"              "g"
[113] "M"               "l"              "|"              "å\u0080\u0093"
[117] "I"               "0"              "1"              "5"
[121] "\u0097"          "Ë\u0086"        "¡"              "\u0080"
[125] "Å'"              "Å\u009f"        "F2"             "ä"
[129] "F1"              "Pause"          "F6"
> View(table(train_logs$event))
```

The frequency table showed that many of those values only occurred once, with about half of them occurring 20 times or less across the entire 8.4 million rows. To reduce the number of values, several categories of related keys were determined to have similar functionality and were combined. The **gsub()** functions below used regular expressions to find variations of different keys and combine them under a new label assigned to each category.

```
> train_logs$event <- gsub("(?i)(Clear|Cancel)", "Escape", train_logs$event)
> train_logs$event <- gsub("^[0-9]$", "DigitKey", train_logs$event)
> train_logs$event <- gsub("(?i).*Arrow.*", "ArrowKey", train_logs$event)
> train_logs$event <- gsub("(?i)(Home|End|^Page.*)", "TextJumpKey", train_logs$event)
> train_logs$event <- gsub("(?i)(F[0-9]{1,2}|.*Media.*|.*Audio.*|.*Pause.*).*", "DeviceFeatureChangeKey",
train_logs$event)
> train_logs$event <- gsub("(?i)(Scroll.*|Meta|Dead|Process|OS|AltGraph|Mode.*)", "SpecialProcessKey",
train_logs$event)
```

In addition to combining similar keys, several uninterpretable values needed to be removed from the data altogether. Those keystrokes were relabeled as “Unidentified”, which is a value that already occurs a few thousand times throughout the column. It was important to use regular expressions to accomplish these changes so that these scripts would generalize to accommodate additional variations of keystroke labels that might show up in the test set. In this case, expected terms were combined with regular expressions in a new vector variable called *event_labels* which could be used to check each event and determine if it is valid.

```
> event_labels <- c("^[A-Za-z]$", "^[[:punct:]]$", "click", "Shift", "Space", "Backspace",
"Enter", "Tab", "Caps", "Control", "Delete", "Insert", "Escape", "Print", "RareKey", "NumLoc
k", "Alt", "ContextMenu", "ArrowKey", "TextJumpKey", "DeviceFeatureChangeKey", "SpecialProce
ssKey", "DigitKey", "Unidentified")
```

The **sapply()** function was used to plug each value from the *event_labels* vector into a **grepl()** function that checked for matches against the unique values of the *event* column. The **apply()** function combined with the **any()** function returned a final vector of logical values that corresponded to whether any matches were found. This logical vector was then used within the

subsetting syntax to exclude any unique values from the event column that did not find a match. The results were stored in a new vector variable named *valid_events.*

```
> valid_events ← unique(train_logs$event)[apply(sapply(event_labels, function(x) grepl(x
unique(train_logs$event))), 1, any)]
```

To confirm that this function worked as intended, a frequency table was printed for unique *events* using the exclude parameter to remove all *valid_events.* Those values were then all relabeled as "Unidentified" using an **ifelse()** function.

```
> print(table(train_logs$event, exclude = valid_events))

      \u0080       \u0096       \u0097       \u009b            ã            Â´
           1            2           14            1            1            1
          Ã±      Å\u009f â\u0080\u0093      Ë\u0086
           3            1            4            1
> train_logs$event ← ifelse(train_logs$event %in% valid_events, train_logs$event, "Unidentified")
```

After all of the relabeling described above, there were still 90 unique values in the *event* column. As mentioned, many of these values occurred very rarely. Given the large number of rows in the dataset any rare keystroke would be useless individually for predictive modeling. For this reason, all values that occurred less than 25 times in the column were relabeled as "RareKey". This may be more useful anyway to determine if rare keystrokes in general correlate with any specific outcomes. It also simplifies the *event* column by removing a lot of noise. This was achieved using the function below. The remaining values from the *event* column were also printed to examine more closely.

```
> train_logs$event[train_logs$event %in% names(which(table(train_logs$event) < 25))] ← "RareKey"
> print(unique(train_logs$event))
 [1] "Leftclick"           "Shift"               "q"
 [4] "Space"               "Backspace"           "."
 [7] ","                   "Enter"               "ArrowKey"
[10] "'"                   ";"                   "-"
[13] "?"                   "Tab"                 "\""
[16] "Rightclick"          "="                   "CapsLock"
[19] "Control"             "c"                   "v"
[22] "/"                   "Delete"              ":"
[25] "z"                   "["                   "$"
[28] "("                   ")"                   "+"
[31] "TextJumpKey"         "\\"                  "SpecialProcessKey"
[34] "*"                   "&"                   "DeviceFeatureChangeKey"
[37] "x"                   "!"                   "Insert"
[40] "NumLock"             "%"                   "RareKey"
[43] ">"                   "Alt"                 "ContextMenu"
[46] "a"                   "<"                   "]"
[49] "Unidentified"        "s"                   "Escape"
[52] "DigitKey"
```

Although there are still a large number of potential values, various keys will be more relevant to certain activity types. It is worth noting that the letter keystrokes remaining in the distribution were not labeled as "Input" or "Replace", which means they were likely used as part of a keyboard shortcut, which is why they were not anonymized like the rest of the letter inputs. Since the remaining keystrokes seem relevant and expected, the **as.factor()** function was used to change the data type of the *event* column to "Factor".

```
> train_logs$event ← as.factor(train_logs$event)
> str(train_logs$event)
 Factor w/ 52 levels "'","-","!","\"",..: 38 38 44 40 40 40 40 40 40 45 ...
```

B. Univariate Analysis

Before beginning the Data Analysis it's important to consider the roles that each variable plays within the data set. *Id* is a unique identifier of the participants, so it has no predictive value, but will be used for aggregating metrics from the raw data logs. *Event_id* is an index, so it makes more sense to visualize the final count rather than the raw data that contains all of the sequential values. *Down_time* and *up_time* are timestamps that will be used for deriving other metrics, but they are not easily analyzed as raw data without using advanced techniques like time series analysis or neural networks. *Text_change* is a description of the changes in the text, but it contains too many unique values to analyze easily due to the variation in events like “cut”, “paste”, and “replace”. Since the other categorical variables are simpler representations of those changes in the text, they will receive more focus in this analysis.

As previously mentioned, *score* is the dependent variable, and its distribution will be analyzed first. The remaining variables shown in the blue boxes below will be analyzed in their raw form to help determine the best approach for deriving metrics that can be aggregated by *id* and added to the *train_scores* data frame for use in future predictive modeling efforts.

```
> str(train_logs)
'data.frame':   8405898 obs. of  10 variables:
 $ id             : chr  Unique Identifier
 $ event_id       : int  Index of Events
 $ down_time      : int  Time-Series Data
 $ up_time        : int  Time-Series Data
 $ action_time    : int  Numeric Variable
 $ activity       : Factor w/ 6 levels  Categorical Variable
 $ event          : Factor w/ 52 levels Categorical Variable
 $ text_change    : chr  Description (many unique values)
 $ cursor_position: int  Numeric Variable
 $ word_count     : int  Numeric Variable
> str(train_scores)
'data.frame':   2471 obs. of  2 variables:
 $ id   : chr  Unique Identifier
 $ score: Factor w/ 12 levels Dependent Variable
```

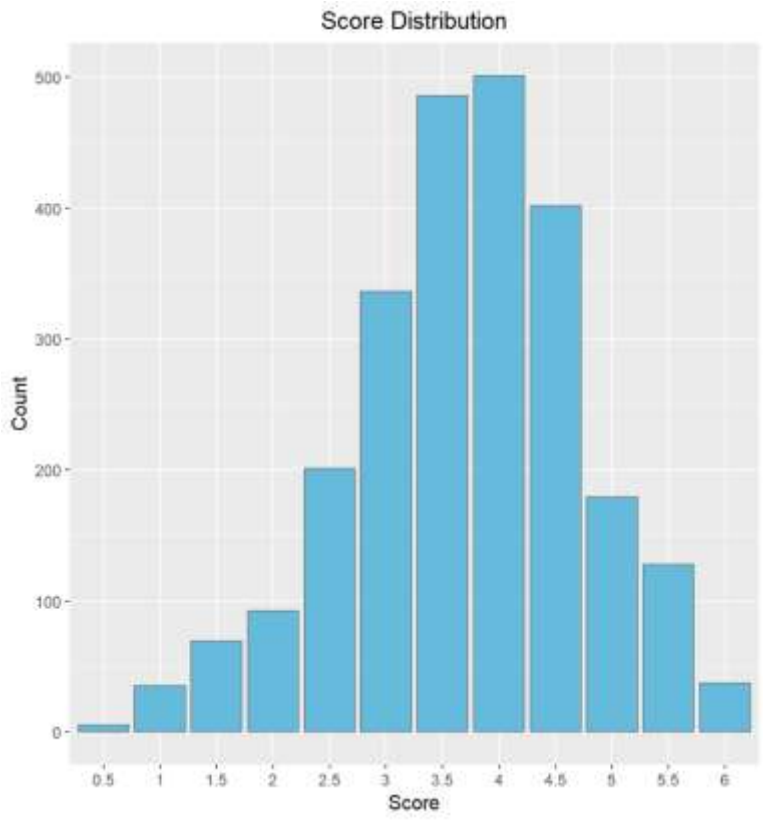
To aid in the analysis and visualization of this dataset, the “ggplot2” and “moments” libraries were added to the global environment in RStudio before beginning. These libraries were used for the creation of visualizations and the calculation of descriptive statistics.

B.1 Dependent Variable: Score

To analyze the distribution of *score*, a bar graph was generated using the ggplot command below. In addition, summary statistics were calculated as well as the count and frequency of each of the 12 possible *score* values, which were compiled into a table below.

```
> ggplot(train_scores, aes(x = score)) +
+   geom_bar(color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "Score", y = "Count", title = "Scores Distribution") +
+   theme(plot.title = element_text(hjust = 0.5))
```

```
> score_distribution ← table(train_scores$score)
> View(score_distribution)
> View(round(prop.table(score_distribution)*100, 2))
```



Mean = 3.7113
Standard Deviation = 1.0249
Skewness = -0.2949
Kurtosis = 3.0681

Score	Count	Frequency
0.5	5	0.2 %
1	35	1.42 %
1.5	69	2.79 %
2	92	3.72 %
2.5	201	8.13 %
3	336	13.6 %
3.5	486	19.67 %
4	501	20.28 %
4.5	402	16.27 %
5	179	7.24 %
5.5	128	5.18 %
6	37	1.5 %

```
> summary(train_scores$score_numeric)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.500   3.000   4.000   3.711   4.500   6.000
> sd(train_scores$score_numeric)
[1] 1.024937
> skewness(train_scores$score_numeric)
[1] -0.2949374
> kurtosis(train_scores$score_numeric)
[1] 3.068182
```

The distribution of *score* is skewed left with a mean of 3.7113, showing wider variance to the left of its median value of 4. This is confirmed by the negative value of skewness, which denotes a slightly longer tail to the left side of the distribution. In addition, the distribution is leptokurtic with a concentration of scores around the mean and much fewer observations of scores at the extreme ends of the range.

In general, the distribution has a bell-shaped curve similar to that of a normal distribution, which may suggest it can be reliably modeled. However, the slight skewness and low occurrence of

scores for extreme values will make it more difficult to predict the occurrence of scores at those values.

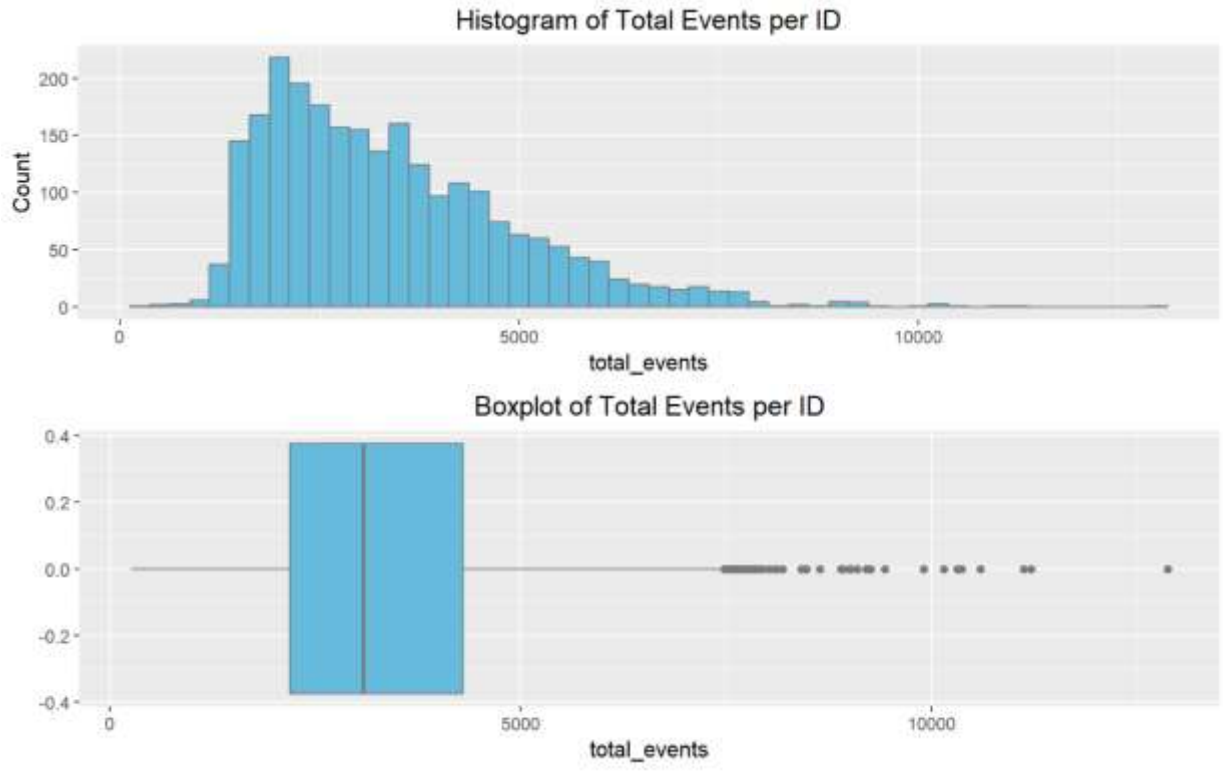
B.2 Independent Variables: Numeric

Total Events (Max *event_id* Per Submission)

```
> train_scores ← merge(aggregate(event_id ~ id, data = train_logs, FUN = max), train_scores,
by = "id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "event_id"] ← "total_events"
```

Using the functions above, a new column was added to the *train_scores* data frame that lists the total events for each participant. This was achieved by using the **aggregate()** function to find the max value of *event_id* for each *id*. Since the *event_id* variable is a sequential index, deriving the total amount of events for each participant will be more useful in this analysis than trying to visualize the raw data. Using the R functions below, a histogram and box plot were generated using the ggplot2 library. By aligning these visualizations, we get a better idea of the distribution of these values and the number of outliers on the upper end of the range.

```
> ggplot(train_scores, aes(x = total_events)) +
+     geom_histogram(binwidth = 250, color = "#767B7D", fill = "#66BBDD") +
+     labs(x = "total_events", y = "Count", title = "Histogram of Total Events per ID") +
+     theme(plot.title = element_text(hjust = 0.5))
> ggplot(train_scores, aes(x = total_events)) +
+     geom_boxplot(color = "#767B7D", fill = "#66BBDD") +
+     labs(x = "total_events", y = "", title = "Boxplot of Total Events per ID") +
+     theme(plot.title = element_text(hjust = 0.5))
```



```
> summary(train_scores$total_events)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    262    2194    3082    3402    4301   12876
> sd(train_scores$total_events)
[1] 1578.85
> skewness(train_scores$total_events)
[1] 1.158578
> kurtosis(train_scores$total_events)
[1] 4.847581
```

As seen above, summary statistics were also calculated for *total_events*. It has a mean value of 3402, which is pulled far from the median value of 3082 due to the presence of extreme values on the right side of the distribution. A skewness of 1.1586 also denotes a distribution that is significantly skewed right. In addition, its high kurtosis value of 4.8476 makes the distribution leptokurtic because of the long tail and high number of extreme values on the upper end of the range.

The boxplot confirms the existence of outliers with values above the 3rd quartile line by more than 1.5 times the IQR. As seen from the calculation below, 2% of the total submissions (50 out of 2471) would be considered outliers in this distribution. While participants were instructed to stay on the page while completing the assignment, it is unclear whether that rule was enforced. If not, some values may be inaccurate due to off-task activity. Regardless, the test set can be expected to have a similar distribution since it was collected at the same time. Analysis that considers the counts of particular events or activity types may provide more insight into the distribution of events.

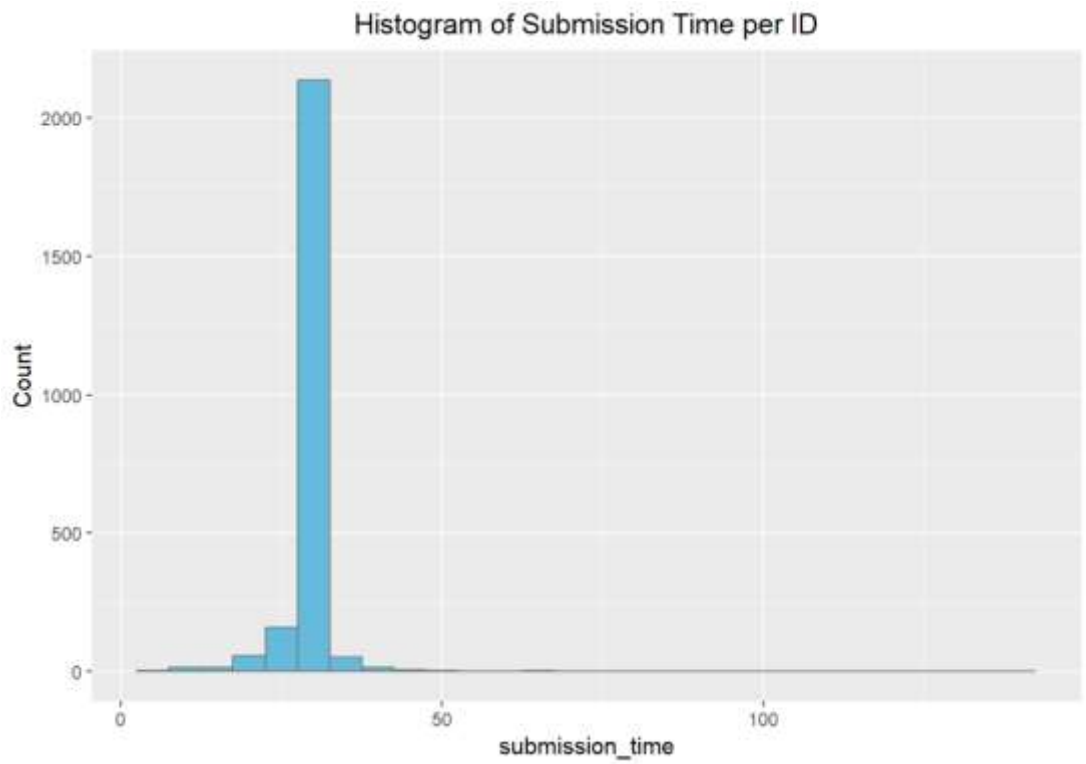
```
> nrow(train_scores[train_scores$total_events > quantile(train_scores
$total_events, 0.75) + (1.5 * IQR(train_scores$total_events)), ])
[1] 50
```

Submission Time (Max *up_time* Per Submission)

```
> train_scores ← merge(aggregate(up_time ~ id, data = train_logs, FUN = max), train_scores,
by = "id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "up_time"] ← "submission_time"
> train_scores$submission_time ← train_scores$submission_time / 60000
```

The submission time for each participant was determined using the functions above. Since *up_time* is sequential, the largest value for this variable would be the completion of the final event that ultimately submitted the essay. The **aggregate()** function was applied again using the "max" parameter to find this value for each unique *id*. Afterwards, a histogram was generated using the R command below to understand the distribution of submission times. Summary statistics were also calculated to better understand the distribution, but a boxplot could not be visualized well due to the tight clustering of values around its central range and the long tail of extreme values extending to the right.

```
> ggplot(train_scores, aes(x = submission_time)) +
+     geom_histogram(binwidth = 5, color = "#767B7D", fill = "#66BBDD") +
+     labs(x = "submission_time", y = "Count", title = "Histogram of Submission Time per ID") +
+     theme(plot.title = element_text(hjust = 0.5))
```



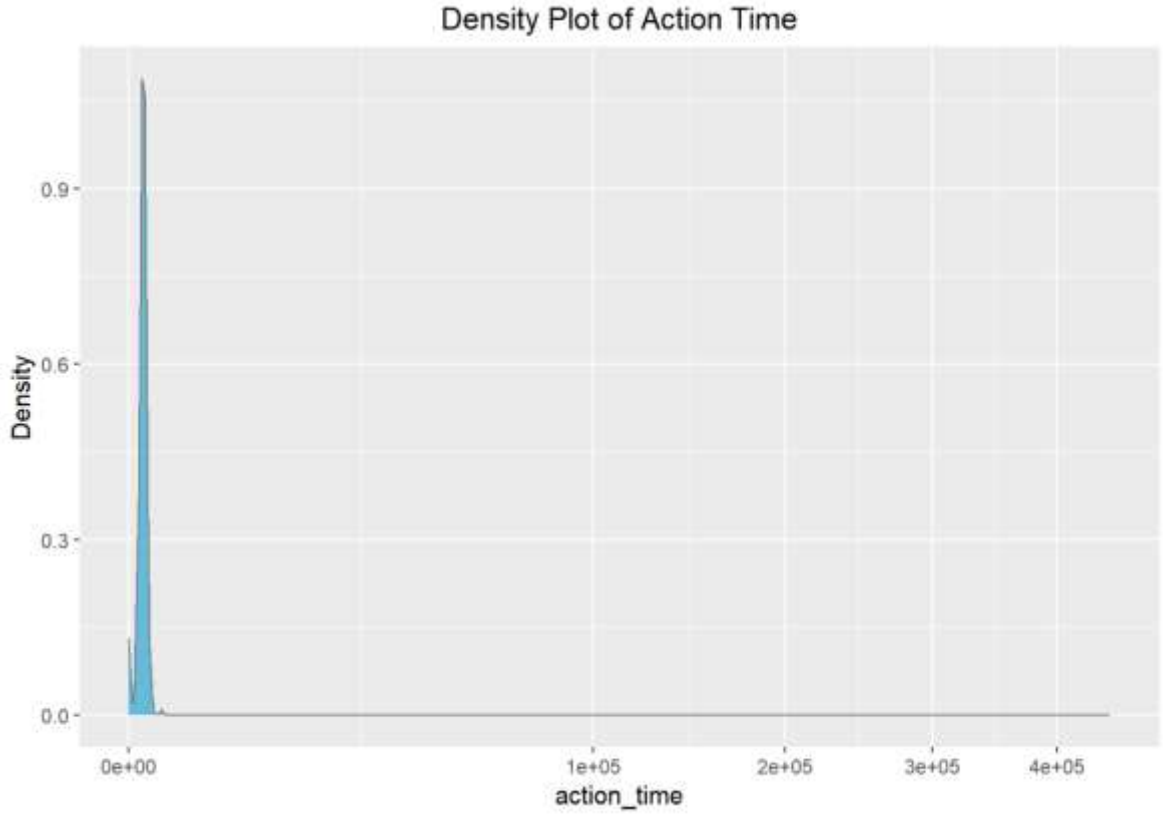

```
> summary(train_scores$submission_time)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.826  29.248  29.872  29.492  30.130 138.562
> sd(train_scores$submission_time)
[1] 4.600471
> skewness(train_scores$submission_time)
[1] 7.355812
> kurtosis(train_scores$submission_time)
[1] 162.4166
```

As seen from the histogram and summary statistics above, the *submission_time* distribution is skewed heavily to the right and is extremely leptokurtic. According to the directions, participants had a 30-minute time limit to complete the essay, but the presence of values above this limit suggest that this was not strictly enforced. The vast majority of participants submitted the essay within a tight window of the 30-minute time limit, but some values were much higher. For this reason, *submission_time* may be less useful for predictive modeling than other variables. Hypothetically, other interesting metrics could be derived from the time stamps, but this may require more advanced algorithms that are beyond the scope of this initial report.

Action Time

The univariate analysis of *action_time* reveals a distribution with extreme skewness and a significant number of outliers. Because of this, the data was very difficult to visualize. The density plot below provided the best view of its distribution. To generate this visualization, I used the ggplot function below with square root scaling on the x-axis to minimize the distortion caused by the extremely long tail of extreme values.

```
> ggplot(train_logs, aes(x = action_time)) +
+   geom_density(color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "action_time", y = "", title = "Density Plot of Action Time") +
+   theme(plot.title = element_text(hjust = 0.5)) +
+   scale_x_sqrt()
```




```
> summary(train_logs$action_time)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
     0.0     66.0     93.0     98.1    122.0 447470.0
> sd(train_logs$action_time)
[1] 253.3985
> skewness(train_logs$action_time)
[1] 1040.334
> kurtosis(train_logs$action_time)
[1] 1557892
```

As seen from the density plot and summary statistics above, the distribution for this variable has a very long tail to the right. This was expected since most keystrokes would be quick while typing, while other less common events, like clicks and highlights, may take much longer. That being said, there are some surprisingly long values at the far end of this distribution, with one click supposedly lasting almost 7.5 minutes.

Using the current numbers for summary statistics, 176,720 observations would be considered outliers for being above the 3rd quartile line by more than 1.5 times the IQR. (2% of the total observations.)

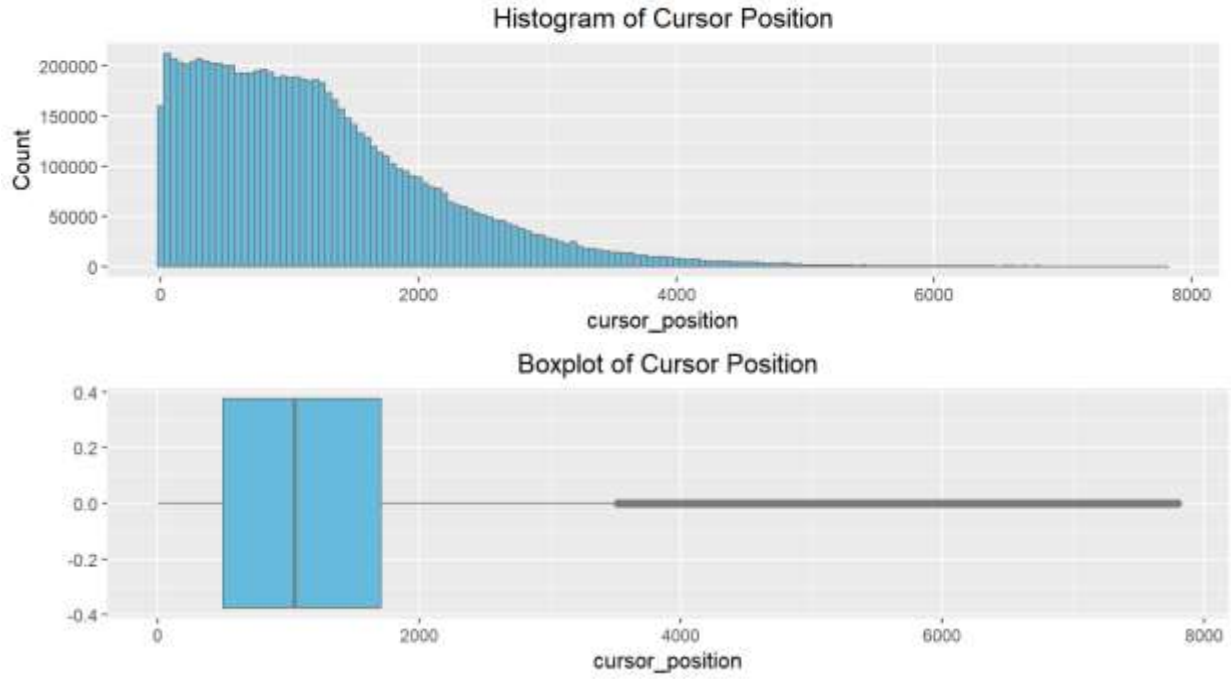
```
> nrow(train_logs[train_logs$action_time > quantile(train_logs$action_time, 0.75) + (1.5 *
IQR(train_logs$action_time)), ])
[1] 176720
> nrow(train_logs[train_logs$action_time > quantile(train_logs$action_time, 0.75) + (1.5 *
IQR(train_logs$action_time)), ]) / nrow(train_logs)
[1] 0.02102333
```

Due to the large number of outliers and the extreme skew of the distribution, it was determined to wait for bivariate analysis to understand if certain activity types or events contribute to these issues so that the best way of handling them can be determined.

Cursor Position

Cursor Position is measured as a character index of the typing cursor within the essay submission textbox. This could be useful for understanding how participants move through the essay, and the degree to which they backtrack to revise or edit parts of their essay. Deriving these insights is not straightforward, though, so a histogram and boxplot were created using the commands below and aligned with each other to get a better understanding of the distribution of data.

```
> ggplot(train_logs, aes(x = cursor_position)) +
+     geom_histogram(binwidth = 50, color = "#767B7D", fill = "#66BBDD") +
+     labs(x = "cursor_position", y = "Count", title = "Histogram of Cursor Position") +
+     theme(plot.title = element_text(hjust = 0.5))
> ggplot(train_logs, aes(x = cursor_position)) +
+     geom_boxplot(color = "#767B7D", fill = "#66BBDD") +
+     labs(x = "cursor_position", y = "", title = "Boxplot of Cursor Position") +
+     theme(plot.title = element_text(hjust = 0.5))
```

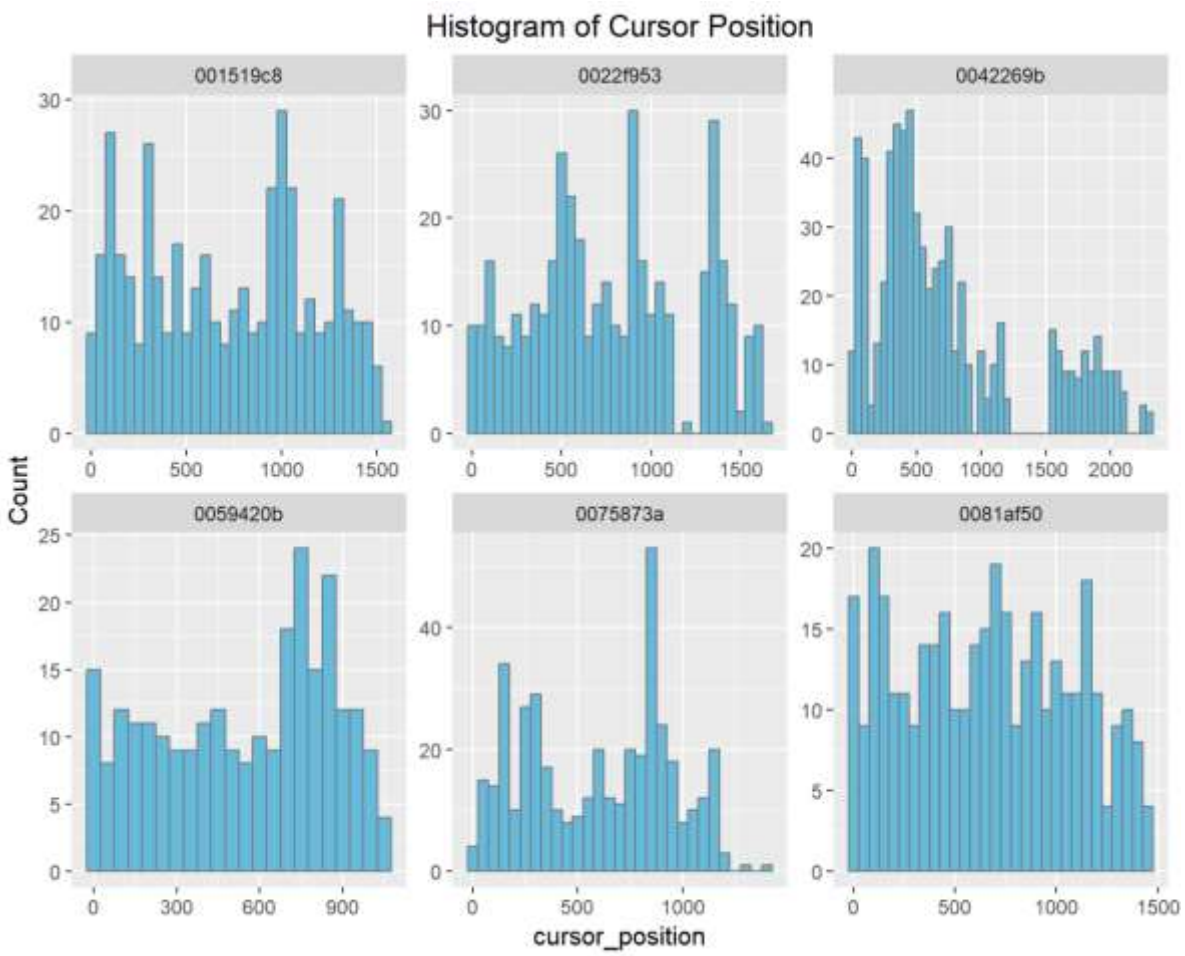


```
> summary(train_logs$cursor_position)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0     499    1043    1223    1706    7802
> sd(train_logs$cursor_position)
[1] 948.5242
> skewness(train_logs$cursor_position)
[1] 1.275548
> kurtosis(train_logs$cursor_position)
[1] 5.331764
```

As seen from the visualizations and summary statistics above, the *cursor_position* variable has a leptokurtic distribution with a long flat tail extending to the right. This is confirmed by its skewness value of 1.2755 and the fact that its mean value of 1223 is greater than its median value of 1043. According to the boxplot, there are also many outliers within this distribution. However, both the skewed distribution and presence of outliers can be explained by the differences in the lengths of essays amongst the participants. The distribution of total events suggests that the lengths of submitted essays are significantly skewed right. This means that there will be significantly less people typing longer essays that would reach the character indexes that are being labeled as outliers in the box plot.

To account for the influence that essay length has on *cursor_position*, a normalized value could be calculated for *cursor_position* by dividing it within each submission by the final word count of that submission. However, normalizing variables in the raw data doesn't help, if the distribution of *cursor_position* is not different enough from one submission to the next to have any predictive value. To check if this is the case, histograms of *cursor_position* for each of the first few submissions were created using the following R command.

```
> ggplot(initial_logs, aes(x = cursor_position)) +
+   geom_histogram(binwidth = 50, color = "#767B7D", fill = "#66B8DD") +
+   labs(x = "cursor_position", y = "Count", title = "Histogram of Cursor Position") +
+   theme(plot.title = element_text(hjust = 0.5)) +
+   facet_wrap(~id, scales = "free")
```



Since the distribution of *cursor_position* looks fairly different for each submission, the differences in the data may have predictive value if they represent actual differences in the patterns of behavior of the participants. Conceivably, more kurtosis or a higher IQR may result from participants who jump around in the text to edit the overall flow of their ideas. It is unclear which metrics would be most correlated with specific behaviors so the following summary statistics will be aggregated by id and added to the train scores data frame: Mean, median, IQR, standard deviation, skewness and kurtosis. Normalized values were calculated within the raw data, and the following code was run to visualize the new distributions of these metrics.

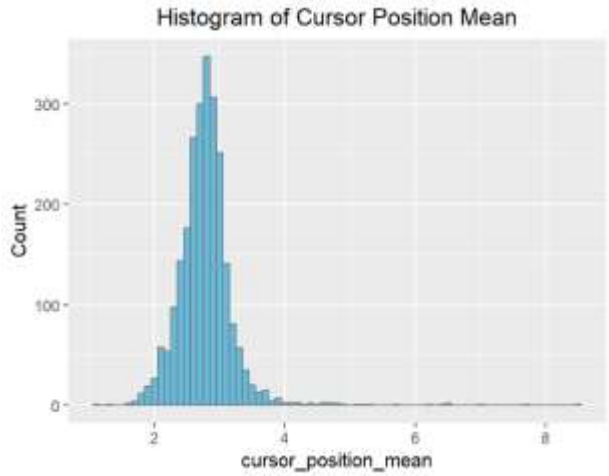
```
train_logs ← merge(train_scores[ , c("id", "submitted_words")], train_logs, by = "id", all.x = TRUE)
train_logs$normalized_cursor_position ← train_logs$cursor_position / train_logs$submitted_words
```

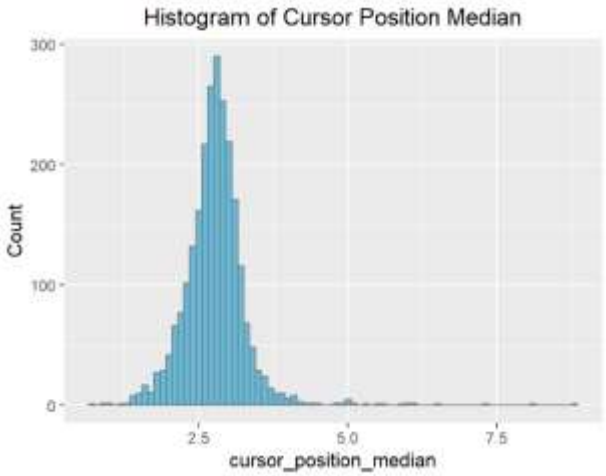
```
> train_scores ← merge(aggregate(normalized_cursor_position ~ id, data = train_logs, FUN = mean), train_scores, by
= "id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "normalized_cursor_position"] ← "cursor_position_mean"
> train_scores ← merge(aggregate(normalized_cursor_position ~ id, data = train_logs, FUN = sd), train_scores, by =
"id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "normalized_cursor_position"] ← "cursor_position_sd"
> train_scores ← merge(aggregate(normalized_cursor_position ~ id, data = train_logs, FUN = median), train_scores, b
y = "id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "normalized_cursor_position"] ← "cursor_position_median"
> train_scores ← merge(aggregate(normalized_cursor_position ~ id, data = train_logs, FUN = skewness), train_scores,
by = "id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "normalized_cursor_position"] ← "cursor_position_skewness"
> train_scores ← merge(aggregate(normalized_cursor_position ~ id, data = train_logs, FUN = kurtosis), train_scores,
by = "id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "normalized_cursor_position"] ← "cursor_position_kurtosis"
> train_scores ← merge(aggregate(normalized_cursor_position ~ id, data = train_logs, FUN = IQR), train_scores, by =
"id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "normalized_cursor_position"] ← "cursor_position_IQR"
```

These metrics were then visualized, to see if they had a distribution that seemed more useful than the raw data:

```
> ggplot(train_scores, aes(x = cursor_position_mean)) +
+   geom_histogram(binwidth = 0.1, color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "cursor_position_mean", y = "Count", title = "Histogram of Cursor Position Mean") +
+   theme(plot.title = element_text(hjust = 0.5))
> summary(train_scores$cursor_position_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.091   2.558   2.769   2.780   2.962   8.503
> sd(train_scores$cursor_position_mean)
[1] 0.4470499
> skewness(train_scores$cursor_position_mean)
[1] 3.186387
> kurtosis(train_scores$cursor_position_mean)
[1] 31.96935
```

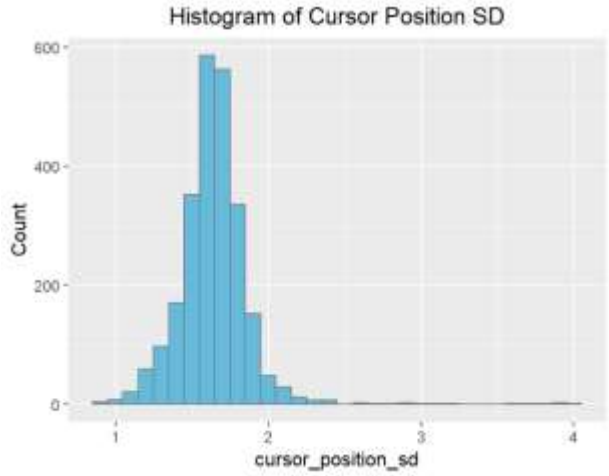
```
> ggplot(train_scores, aes(x = cursor_position_median)) +
+   geom_histogram(binwidth = 0.1, color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "cursor_position_median", y = "Count", title = "Histogram of Cursor Position Median") +
+   theme(plot.title = element_text(hjust = 0.5))
> summary(train_scores$cursor_position_median)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.7419  2.5070  2.7786  2.7711  3.0137  8.7731
> sd(train_scores$cursor_position_median)
[1] 0.5359933
> skewness(train_scores$cursor_position_median)
[1] 2.066285
> kurtosis(train_scores$cursor_position_median)
[1] 20.33637
```

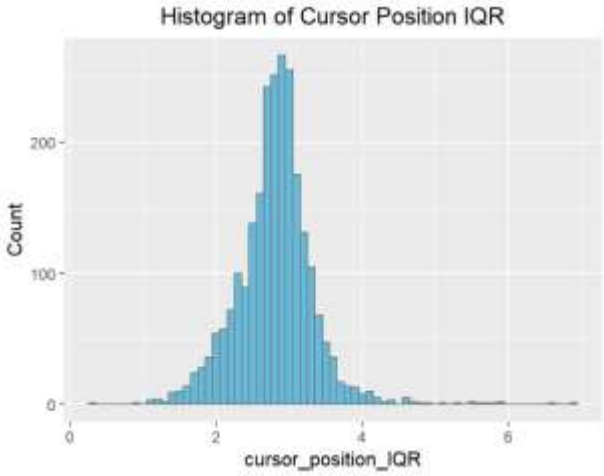




```
> ggplot(train_scores, aes(x = cursor_position_sd)) +
+   geom_histogram(binwidth = 0.1, color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "cursor_position_sd", y = "Count", title = "Histogram of Cursor Position SD") +
+   theme(plot.title = element_text(hjust = 0.5))
> summary(train_scores$cursor_position_sd)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8546  1.5298  1.6387  1.6438  1.7484  3.9513
> sd(train_scores$cursor_position_sd)
[1] 0.2394807
> skewness(train_scores$cursor_position_sd)
[1] 2.533568
> kurtosis(train_scores$cursor_position_sd)
[1] 24.07049
```

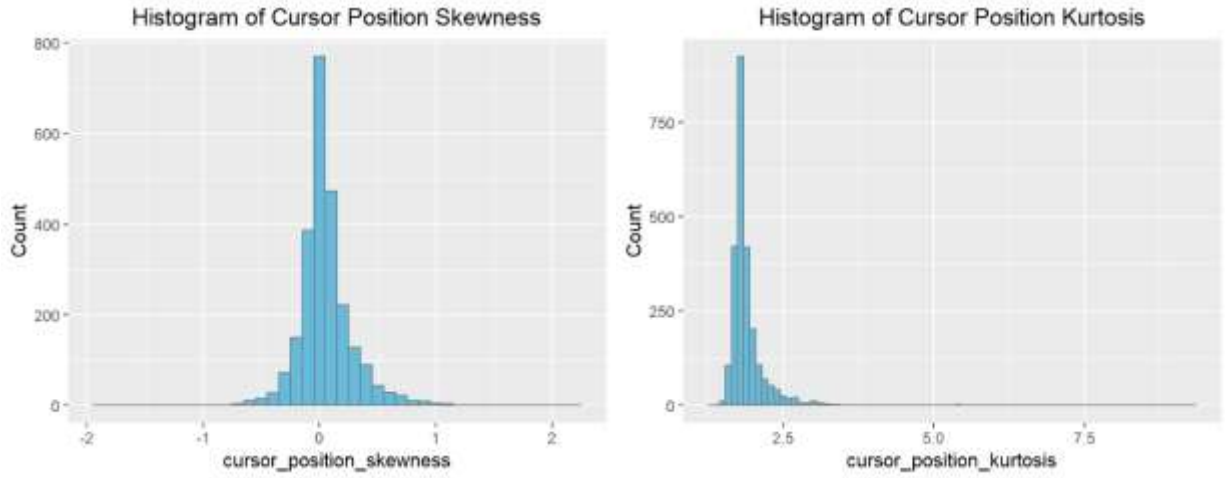
```
> ggplot(train_scores, aes(x = cursor_position_IQR)) +
+   geom_histogram(binwidth = 0.1, color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "cursor_position_IQR", y = "Count", title = "Histogram of Cursor Position IQR") +
+   theme(plot.title = element_text(hjust = 0.5))
> summary(train_scores$cursor_position_IQR)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3235  2.5356  2.8260  2.8043  3.0716  6.8557
> sd(train_scores$cursor_position_IQR)
[1] 0.5296956
> skewness(train_scores$cursor_position_IQR)
[1] 0.7131209
> kurtosis(train_scores$cursor_position_IQR)
[1] 8.691049
```






```
> ggplot(train_scores, aes(x = cursor_position_skewness)) +
+   geom_histogram(binwidth = 0.1, color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "cursor_position_skewness", y = "Count", title = "Histogram of Cursor Position Skewness") +
+   theme(plot.title = element_text(hjust = 0.5))
> summary(train_scores$cursor_position_skewness)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.92141 -0.05713  0.01765  0.05108  0.13263  2.19167
> sd(train_scores$cursor_position_skewness)
[1] 0.2312358
> skewness(train_scores$cursor_position_skewness)
[1] 0.8979508
> kurtosis(train_scores$cursor_position_skewness)
[1] 12.21915
```

```
> ggplot(train_scores, aes(x = cursor_position_kurtosis)) +
+   geom_histogram(binwidth = 0.1, color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "cursor_position_kurtosis", y = "Count", title = "Histogram of Cursor Position Kurtosis") +
+   theme(plot.title = element_text(hjust = 0.5))
> summary(train_scores$cursor_position_kurtosis)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.286   1.758   1.820   1.904   1.940   9.307
> sd(train_scores$cursor_position_kurtosis)
[1] 0.3372109
> skewness(train_scores$cursor_position_kurtosis)
[1] 7.646697
> kurtosis(train_scores$cursor_position_kurtosis)
[1] 120.7951
```



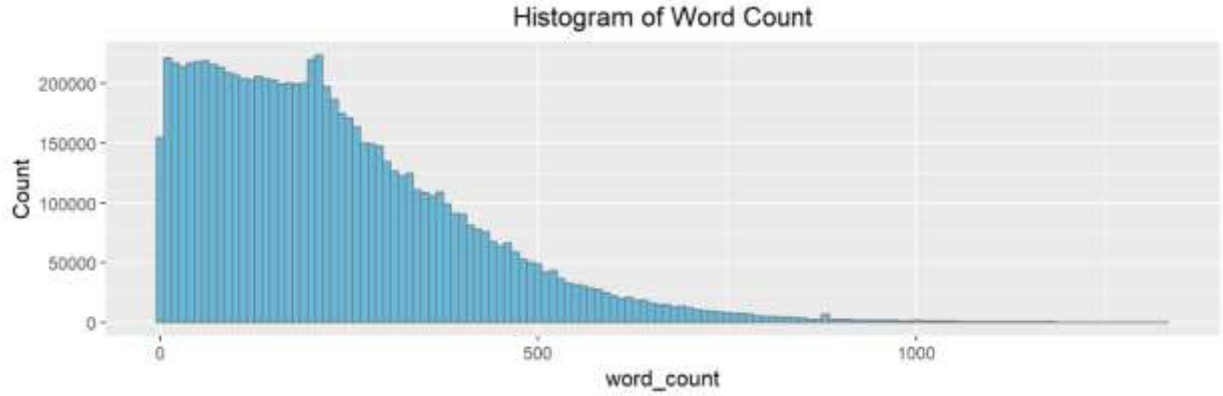
Most of these derived metrics of *cursor_position* still show a significantly right skewed distribution. Long tails to the right indicate that there are still higher values than expected in the normalized values. This may be a result of normalizing with the submitted words metric for each participant, because some participants may have typed to positions further than the ultimate submitted length and revised it back to a shorter length before submitting. If so, this may be the one type of behavior that would show more correlation with higher scores. Bivariate analysis will ultimately need to be conducted to understand the relevance of these metrics for predicting score

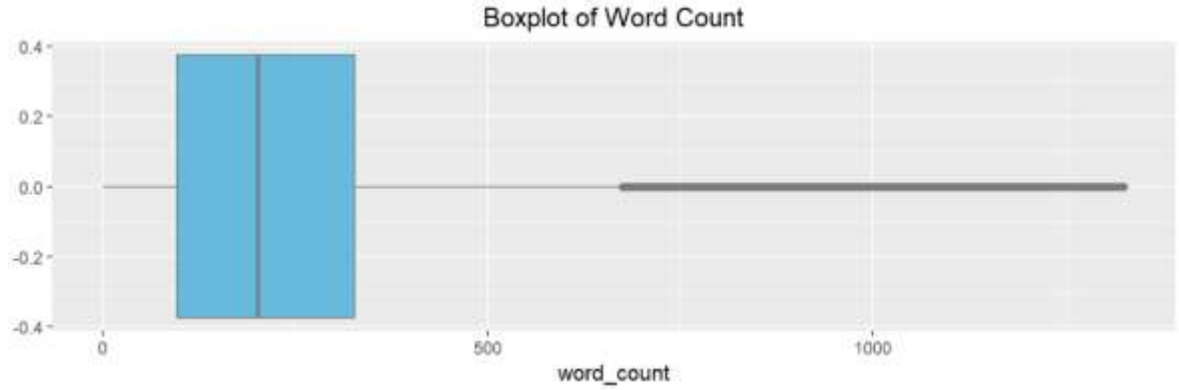
Word Count

The raw data for *word_count* was visualized using the ggplot commands below. Like *cursor_position*, the data will be heavily weighted to the front of the distribution because all submissions will have word counts in that range and much fewer will have longer word counts. The histogram and boxplot below show a similar distribution to *cursor_position* as a result.

```
> ggplot(train_logs, aes(x = word_count)) +
+   geom_histogram(binwidth = 25, color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "word_count", y = "Count", title = "Histogram of Word Count") +
+   theme(plot.title = element_text(hjust = 0.5))
```

```
> ggplot(train_logs, aes(x = word_count)) +
+   geom_boxplot(color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "word_count", y = " ", title = "Boxplot of Word Count") +
+   theme(plot.title = element_text(hjust = 0.5))
```






```
> summary(train_logs$word_count)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    96.0   200.0   231.5   327.0  1326.0
> sd(train_logs$word_count)
[1] 175.9088
> skewness(train_logs$word_count)
[1] 1.191454
> kurtosis(train_logs$word_count)
[1] 4.945565
```

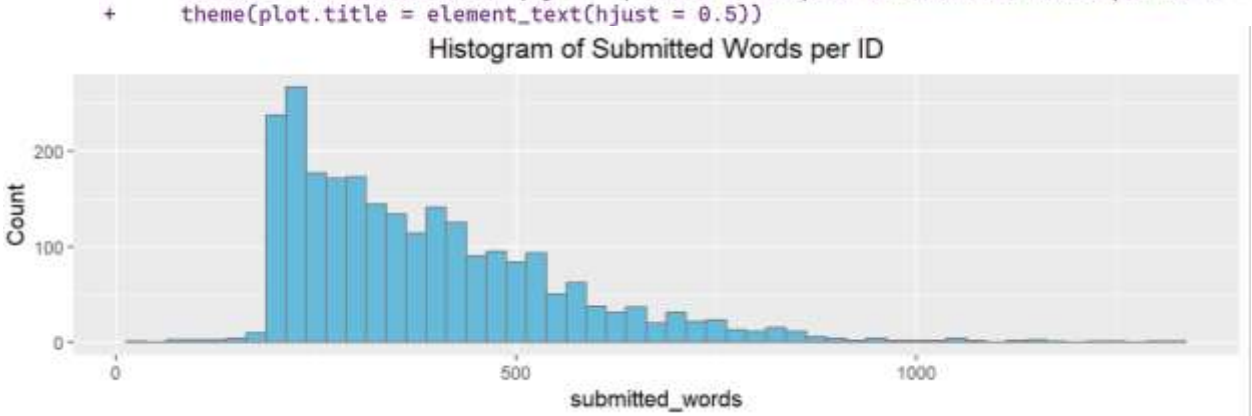
As expected, the raw data shows a significant skew to the right with a positive skewness value of 1.1915 and a Mean value of 231.5, which is considerably higher than the median value of 200. The kurtosis of 4.9456 also denotes the presence of a long tail to the right, and the box plot displays numerous outliers. As previously mentioned, though, this way of visualizing the raw data is less useful due to the large number of occurances of smaller values which will occur within all of the submissions. Ultimately the total number of submitted words will be more useful for understanding the distribution of essay submission lengths. Other metrics could be developed to capture the changes to word count that happen from revision, but this will likely be easier to derive from the activity type data than from descriptive statistics of word count.

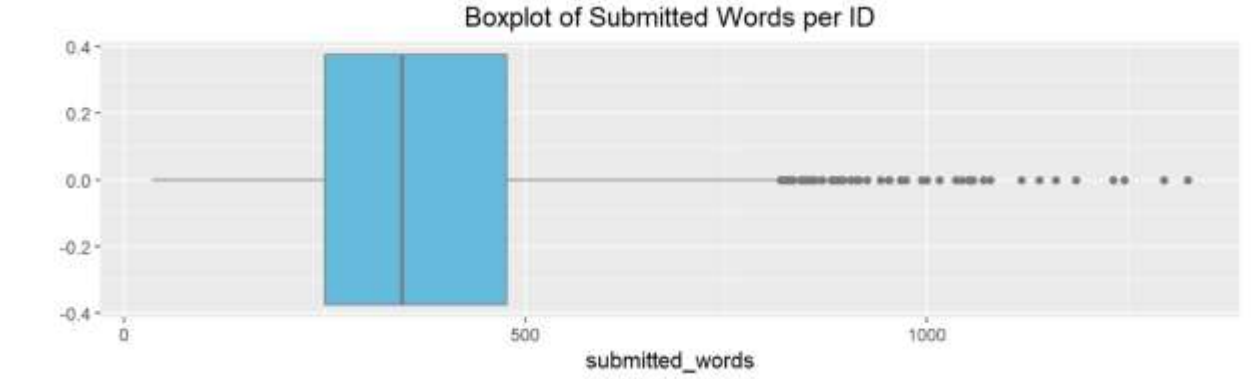
Submitted Words (Final *word_count* Per Submission)

```
> train_scores ← merge(aggregate(word_count ~ id, data = train_logs, FUN = function(x) tail(x, 1)),
train_scores, by = "id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "word_count"] ← "submitted_words"
```

The functions above added another new column to the *train_scores* data frame to list the final word count at submission time for each participant. This was achieved by combining the **aggregate()** and **tail()** functions to grab the last value of *word_count* for each unique *id*. A histogram and box plot were generated using the R commands below, and the visualizations were aligned to get a better idea of the distribution of these values and the number of outliers on the upper end of the range. Summary statistics were also calculated.

```
> ggplot(train_scores, aes(x = submitted_words)) +
+     geom_histogram(binwidth = 25, color = "#767B7D", fill = "#66BBDD") +
+     labs(x = "submitted_words", y = "Count", title = "Histogram of Submitted Words per ID") +
+     theme(plot.title = element_text(hjust = 0.5))
> ggplot(train_scores, aes(x = submitted_words)) +
+     geom_boxplot(color = "#767B7D", fill = "#66BBDD") +
+     labs(x = "submitted_words", y = "", title = "Boxplot of Submitted Words per ID") +
+     theme(plot.title = element_text(hjust = 0.5))
```






```
> summary(train_scores$submitted_words)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   35.0   250.0   346.0   386.1   477.0  1326.0
> sd(train_scores$submitted_words)
[1] 171.7734
> skewness(train_scores$submitted_words)
[1] 1.358784
> kurtosis(train_scores$submitted_words)
[1] 5.493356
```

The distribution of submitted words is skewed to the right significantly. The increased variance on the right side of the chart pulled the mean value to 386.1, which is considerably higher than the median value of 346. The skewness of 1.3588 and the kurtosis values of 5.4934 also denote a long flat tail to the right side of the distribution. The box plot also still shows many outliers beyond the 3rd quartile line by more than 1.5 times the IQR. In total there are 62 submissions that would be considered outliers by our calculations below, which amount to about 2.5% of the total submissions.

```
> nrow(train_scores[train_scores$submitted_words > quantile(train_scores$submitted_words, 0.75) + (1.5 *
IQR(train_scores$submitted_words)), ])
[1] 62
> nrow(train_scores[train_scores$submitted_words > quantile(train_scores$submitted_words, 0.75) + (1.5 *
IQR(train_scores$submitted_words)), ]) / nrow(train_scores)
[1] 0.02509106
```

Ultimately, these descriptive statistics look very similar to those of the raw *word_count* data, but the aggregated values will be easier to incorporate into our bivariate analysis. Despite the high number of outliers, no changes will be made to the data. The distribution of the test set is expected to closely mirror the training set, so a similar number of outliers should be expected. Additionally, since the actual text of the essays are obscured, it is difficult to determine if these outliers resulted from excessive wordiness on the part of enthusiastic participants or through errors, repetition, or some other means. Regardless, some trends may still be derivable from the distribution that can increase the predictive potential of future models.

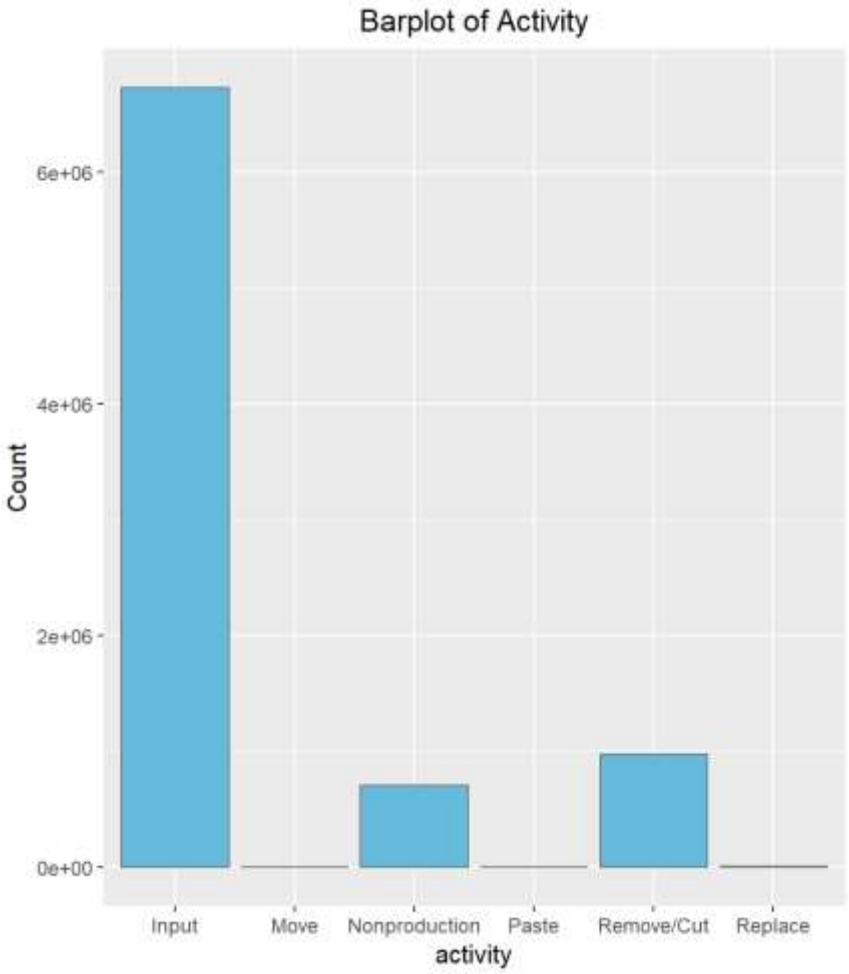
B.3 Independent Variables: Categorical

Activity

The *activity* variable in the *train_logs* data frame records the type of activity that occurs for each logged event. Understanding some of the differences in the rate of activity within different submissions may offer insights into writing behaviors. To understand the distribution of the different activity types, the following R commands were used to create a bar chart as well as a frequency table, which was compiled below.

```
> ggplot(train_logs, aes(x = activity)) +
+   geom_bar(color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "activity", y = "Count", title = "Barplot of Activity") +
+   theme(plot.title = element_text(hjust = 0.5))
> View(table(train_logs$activity))
> print(round(prop.table(table(train_logs$activity))* 100, 4))

      Input          Move Nonproduction         Paste    Remove/Cut       Replace
    80.0247        0.0005        8.3733        0.0071       11.5414        0.0529
```



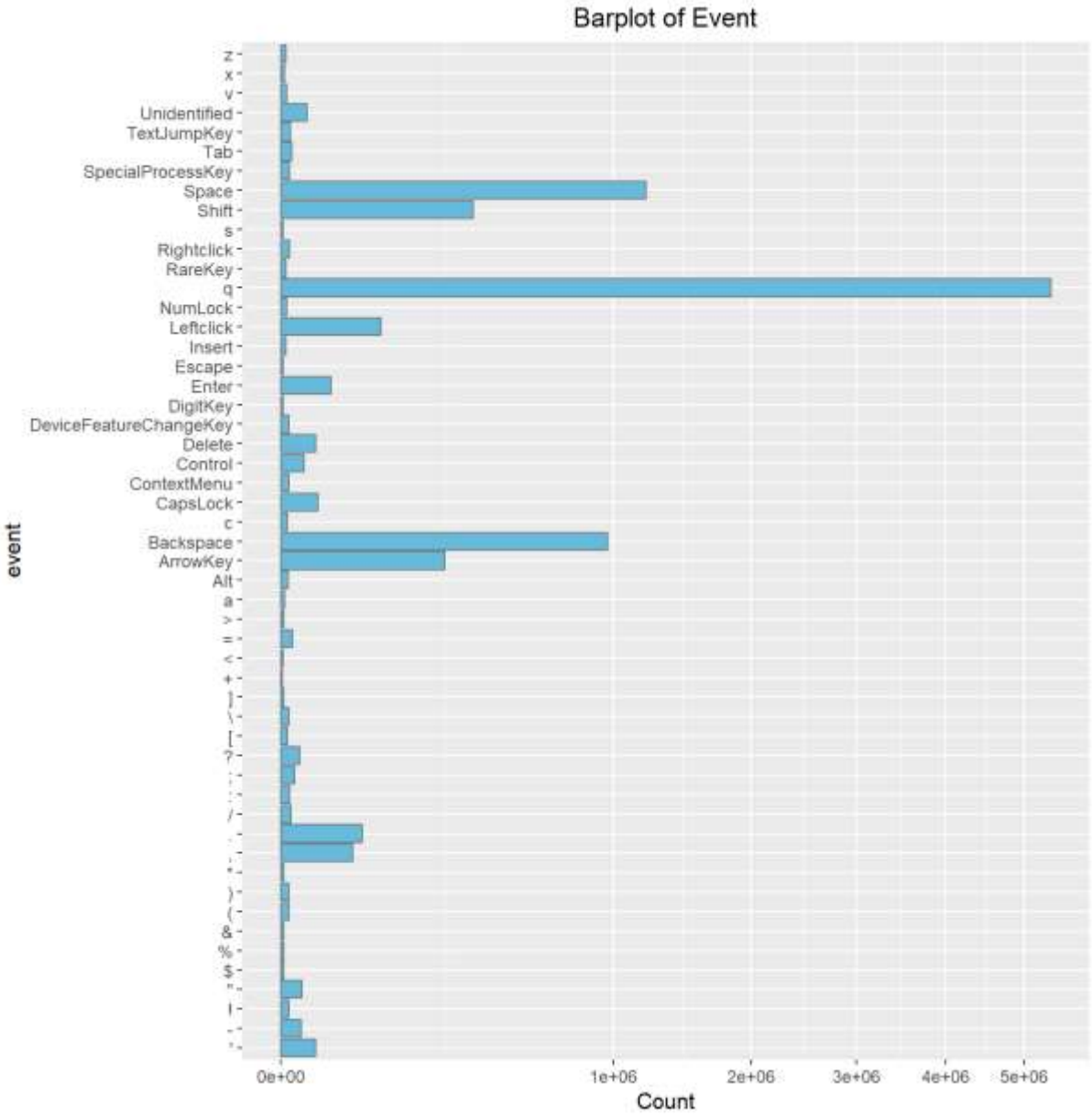
Activity	Count	Frequency
Input	6,726,796	80.02%
Move	46	0.0005%
Nonproduction	703,851	8.37%
Paste	599	0.007%
Remove/Cut	970,158	11.5%
Replace	4448	0.05%

As seen above, “input” is by far the most frequent activity type as expected, with over 80% of the observations falling into that category. While useful metrics may be derivable from other activity types, the rarest activities occurred too infrequently to be considered reliable for predictive modeling. Even if a strong correlation is observed between those activities and score, there is no guarantee that it will be generalizable.

Event

As discussed during the data cleaning process, the *event* variable in this data set originally had even more unique values, but many with similar functionality were combined to reduce the amount. A bar chart was generated by the function below to analyze the distribution more closely. Square root scaling had to be used on the x-axis to reduce the effect of the large amount of "q" events for easier visualization. A frequency table was also compiled on the next page.

```
> ggplot(train_logs, aes(x = event)) +
+   geom_bar(color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "event", y = "Count", title = "Barplot of Event") +
+   theme(plot.title = element_text(hjust = 0.5)) +
+   coord_flip() +
+   scale_y_sqrt()
```



```
> event_frequency ← merge(table(train_logs$event), round(prop.table(table(train_logs$event))*100, 4),
by = "Var1")
> View(event_frequency)
```

EVENT	COUNT	FREQUENCY	EVENT	COUNT	FREQUENCY
q	5365319	63.8280 %	\	554	0.0066 %
Space	1207128	14.3605 %	**ContextMenu**	552	0.0066 %
Backspace	964089	11.4692 %	**DeviceFeature ChangeKey**	551	0.0066 %
Shift	334227	3.9761 %	**!**	546	0.0065 %
ArrowKey	243618	2.8982 %	**Alt**	466	0.0055 %
Leftclick	91126	1.0841 %	**[**	369	0.0044 %
.	59586	0.7089 %	**c**	359	0.0043 %
,	46806	0.5568 %	**v**	311	0.0037 %
Enter	22689	0.2699 %	**NumLock**	273	0.0032 %
CapsLock	12568	0.1495 %	**RareKey**	268	0.0032 %
'	11170	0.1329 %	**Insert**	226	0.0027 %
Delete	10965	0.1304 %	**z**	207	0.0025 %
Unidentified	6180	0.0735 %	**a**	140	0.0017 %
Control	4885	0.0581 %	**x**	127	0.0015 %
"	4102	0.0488 %	**$**	97	0.0012 %
-	3843	0.0457 %	**>**	89	0.0011 %
?	3155	0.0375 %	**]**	81	0.0010 %
;	1785	0.0212 %	*****	69	0.0008 %
=	1155	0.0137 %	**%**	65	0.0008 %
Tab	1081	0.0129 %	**&**	60	0.0007 %
/	874	0.0104 %	**DigitKey**	51	0.0006 %
TextJumpKey	812	0.0097 %	**Escape**	49	0.0006 %
Rightclick	655	0.0078 %	**s**	44	0.0005 %
SpecialProcess Key	639	0.0076 %	**<**	34	0.0004 %
:	634	0.0075 %	**+**	30	0.0004 %
(	611	0.0073 %			
)	578	0.0069 %			

The sheer number of unique “event” types make it difficult to understand the best way to use the *event* data. Rare event types would be useless on their own, but metrics that gauge the diversity of “event” types used may have some usefulness. For this reason, the following functions were used to aggregate this data and add it to the train_scores data frame. In addition, a histogram and

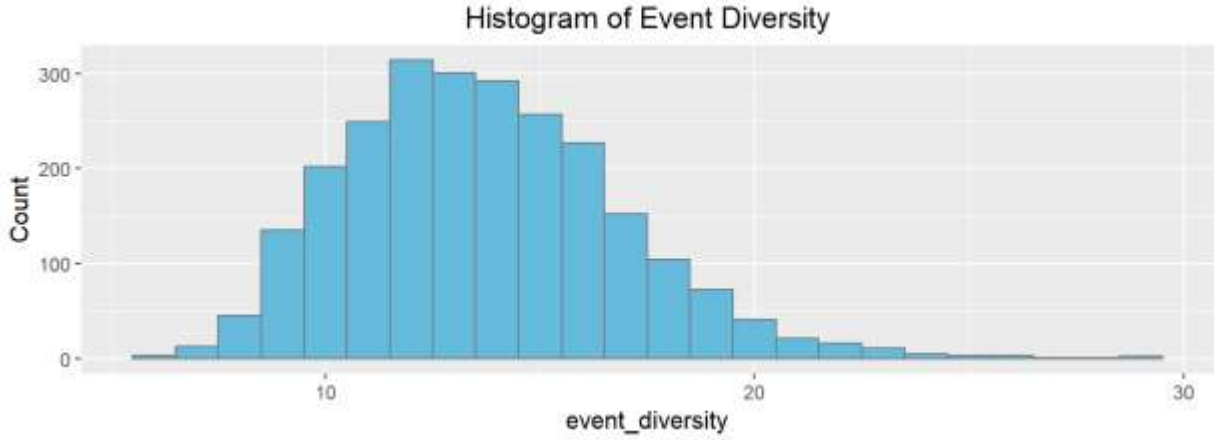
boxplot were created to understand the distribution of this metric, and summary statistics were also calculated.

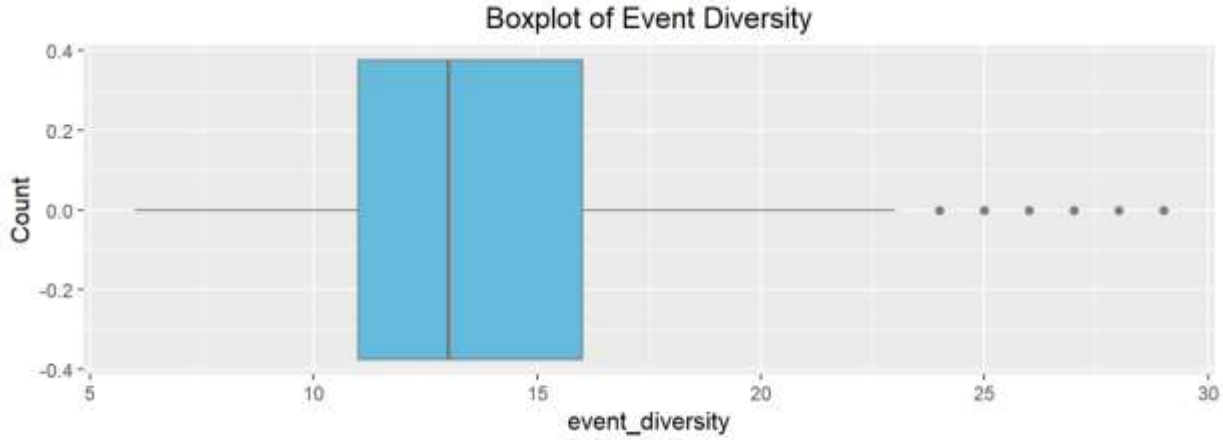
Event Diversity

```
> train_scores ← merge(aggregate(event ~ id, data = train_logs, FUN = function(x) length(unique(x))),
train_scores, by = "id", all.x = TRUE)
> names(train_scores)[names(train_scores) == "event"] ← "event_diversity"
```

```
> ggplot(train_scores, aes(x = event_diversity)) +
+   geom_histogram(binwidth = 1, color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "event_diversity", y = "Count", title = "Histogram of Event Diversity") +
+   theme(plot.title = element_text(hjust = 0.5))
```

```
> ggplot(train_scores, aes(x = event_diversity)) +
+   geom_boxplot(color = "#767B7D", fill = "#66BBDD") +
+   labs(x = "event_diversity", y = "Count", title = "Boxplot of Event Diversity") +
+   theme(plot.title = element_text(hjust = 0.5))
```






```
> summary(train_scores$event_diversity)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   6.00   11.00   13.00   13.68   16.00   29.00
> sd(train_scores$event_diversity)
[1] 3.190763
> skewness(train_scores$event_diversity)
[1] 0.6176656
> kurtosis(train_scores$event_diversity)
[1] 3.773879
```

As seen from the visualizations and summary statistics above, event diversity has a roughly bell-shaped distribution. It is skewed slightly right with a long tail resulting from a hand full of outliers. The kurtosis value of 3.77 also denotes that it is a leptokurtic distribution as a result of the long flat tail to the right.

Bivariate analysis should provide more insights into the ability of using direct event data or event distribution statistics for predictive modeling. As stated previously, rare events would be useless on their own, but a combination of events may have predictive value. Additional metrics could be developed from this data like the frequency of punctuation or potentially the ratio of letters to backspaces. With so many unique values, however, it may take advanced algorithms or neural networks to determine which combination of keys have the most predictive value.

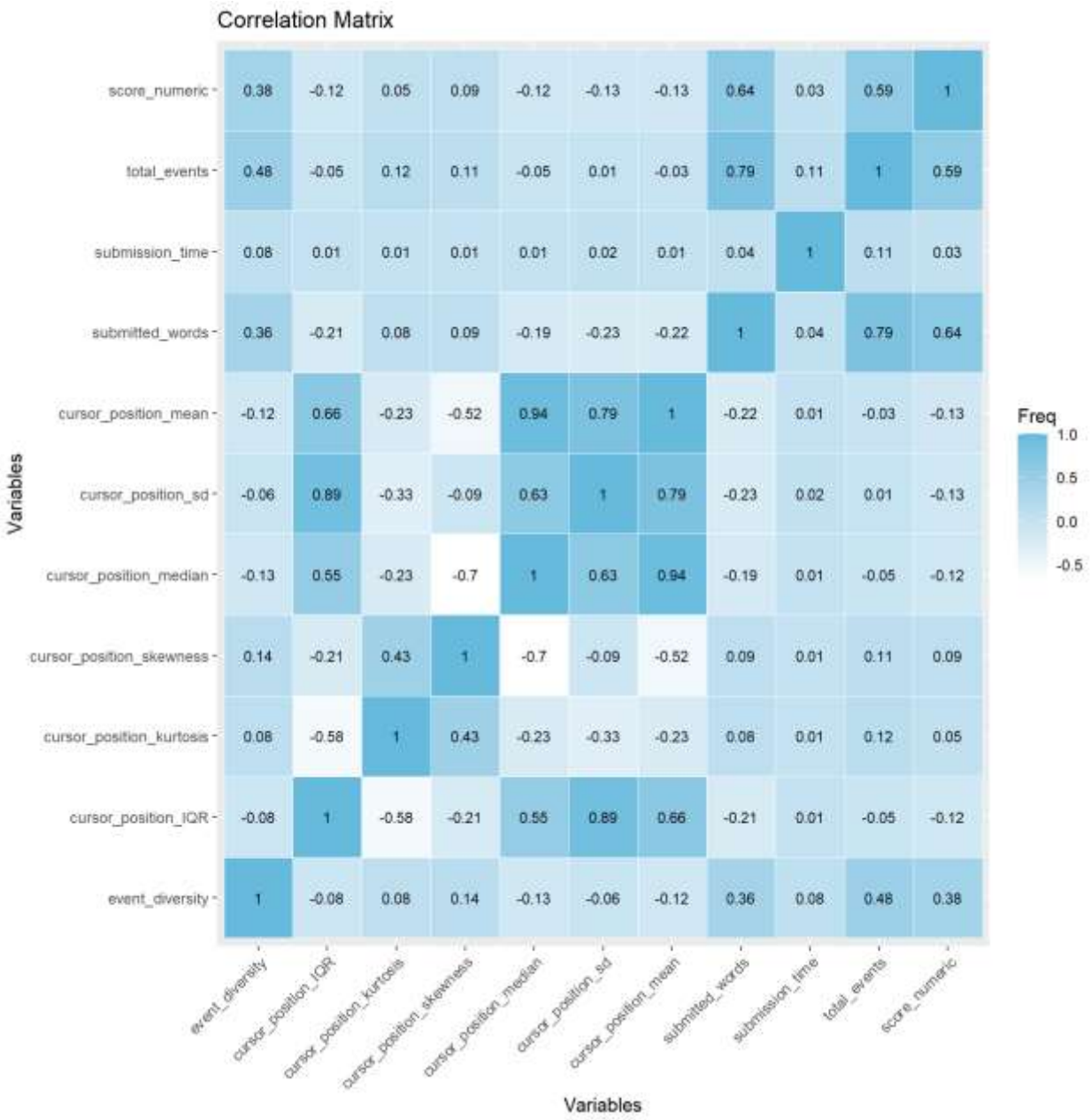
C. Bivariate Analysis

C.1 Numeric Variable Correlations

To begin the bivariate analysis, a correlation matrix was used to determine which numeric values have the highest correlation with each other and ultimately to our dependent variable -- score. I was able to use the ggplot tile geom to visualize the correlation matrix created with the R **cor()** function.

Correlation Matrix

```
correlation_matrix ← as.data.frame(as.table(cor(train_scores[, c(2:11, 13)])))
ggplot(correlation_matrix, aes(Var1, Var2, fill = Freq, label = round(Freq, 2))) +
  geom_tile(color = "white") +
  geom_text(color = "black", size = 3) +
  scale_fill_gradient(low = "#FFFFFF", high = "#66BBDD") +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) +
  labs(title = "Correlation Matrix",
       x = "Variables",
       y = "Variables")
```

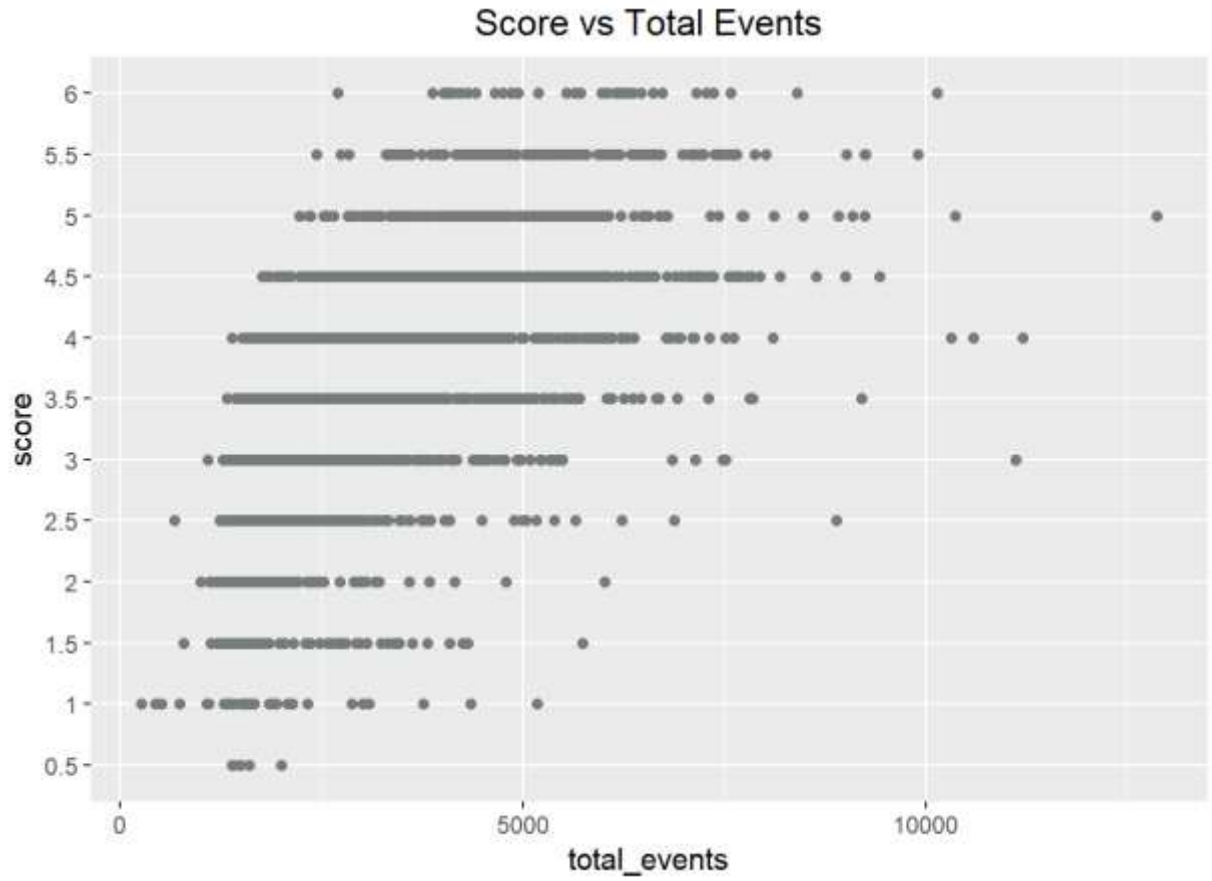


Unfortunately, there are very few numeric variables that show a strong correlation with *score*. Most of the numeric variables that we derived from the raw data seem completely uncorrelated. The two that do show the strongest correlation are *submitted_words* and *total_events*, but even those two variables only have a moderately strong positive correlation. *Event_diversity* shows a mild positive correlation with *score* as well. Our bivariate analysis will visualize those correlations in more depth and explore any other correlations that may exist with the categorical variables.

Score vs. Total Events

According to the correlation matrix, the *total_events* variable was shown to positively correlate to *score* with a correlation coefficient of 0.59. The scatterplot below was generated to better understand how the distribution of these two variables are related.

```
> ggplot(train_scores, aes(x = total_events, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "total_events", y = "score", title = "Score vs Total Events") +
+   theme(plot.title = element_text(hjust = 0.5))
```

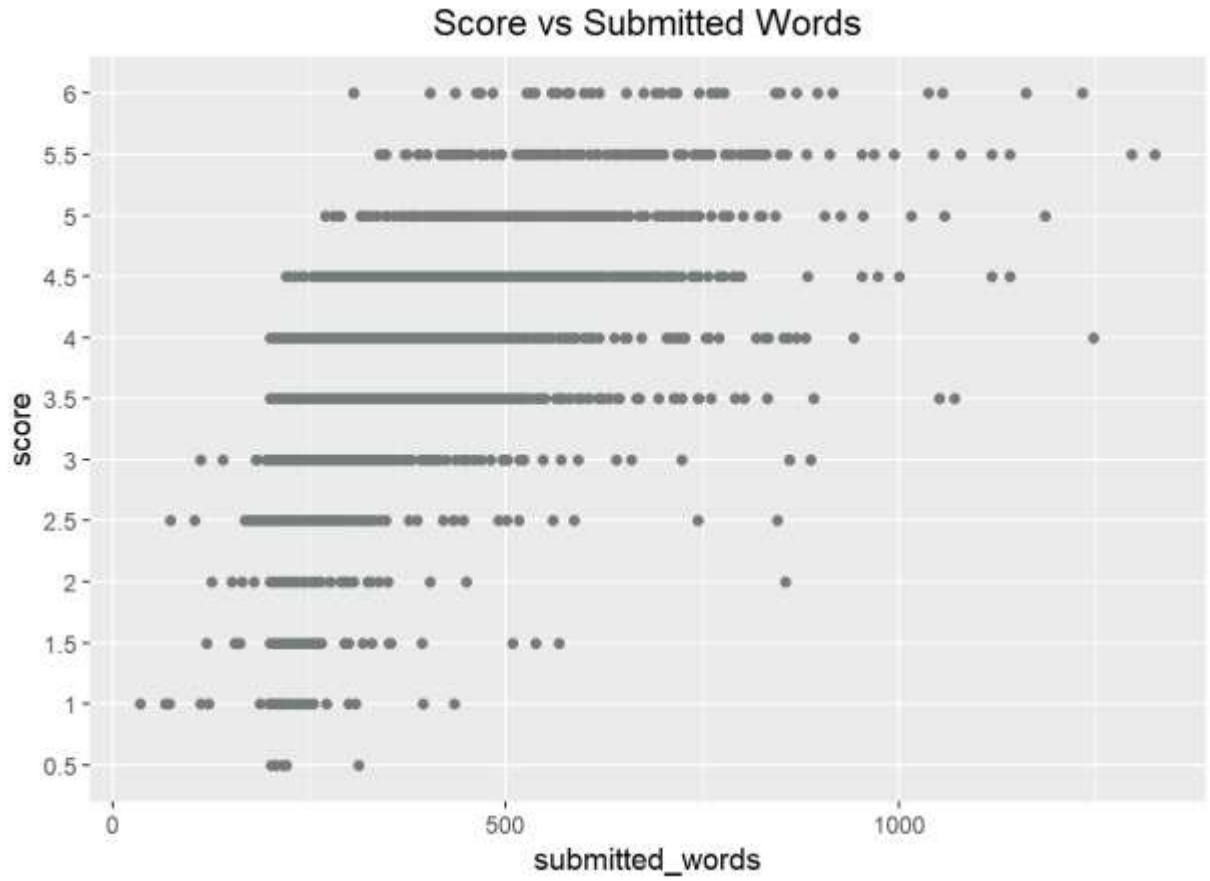


As seen in the scatterplot above, increased number of total events tend to equate to higher scores. There is a lot of overlap in the bands for each score though, which is why these two variables only show a moderate correlation. On its own, *total_events* will likely not have much ability to accurately predict score because most quantities for *total_events* could fit into several different score bands in the distribution.

Score vs. Submitted Words

Score and *submitted_words* are the two variables that show the highest degree of correlation with a coefficient of 0.64. The scatterplot below was generated to closer examine any patterns that exist between the distributions of these variables.

```
> ggplot(train_scores, aes(x = submitted_words, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "submitted_words", y = "score", title = "Score vs Submitted Words") +
+   theme(plot.title = element_text(hjust = 0.5))
```

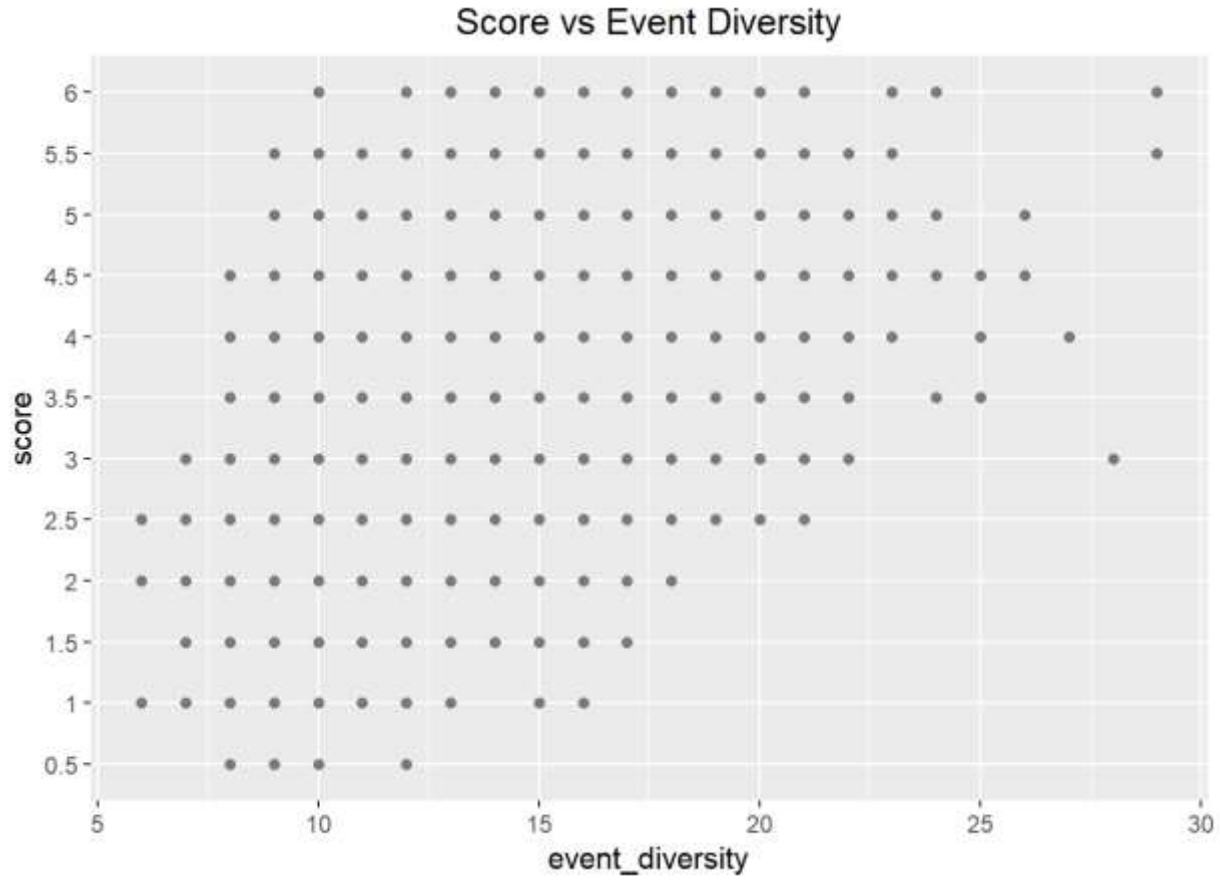


The distributions shown in the scatterplot above reveal that score does frequently increase as the number of submitted words increases. Even with a stronger positive correlation between the two variables, there is still a lot of overlap in the bands for each score. Conceivably, an essay submission 250 words could comfortably fit in the bands from a score of 1 to a score of 4.5. For this reason, this correlation will still not be sufficient on its own to accurately predict scores, especially at the lower end of the range.

Score vs. Event Diversity

The correlation matrix identified that a mild correlation exists between *score* and *event_diversity* with a correlation coefficient of 0.38. The scatterplot below was generated to examine how these variables relate to each other, and what patterns can be observed in the data.

```
> ggplot(train_scores, aes(x = event_diversity, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "event_diversity", y = "score", title = "Score vs Event Diversity") +
+   theme(plot.title = element_text(hjust = 0.5))
```

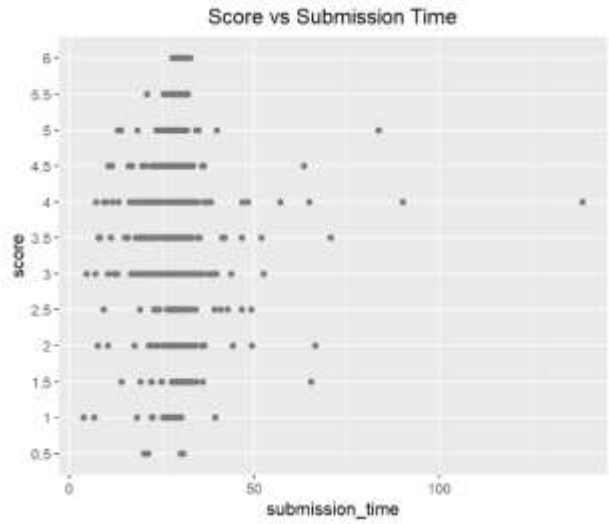


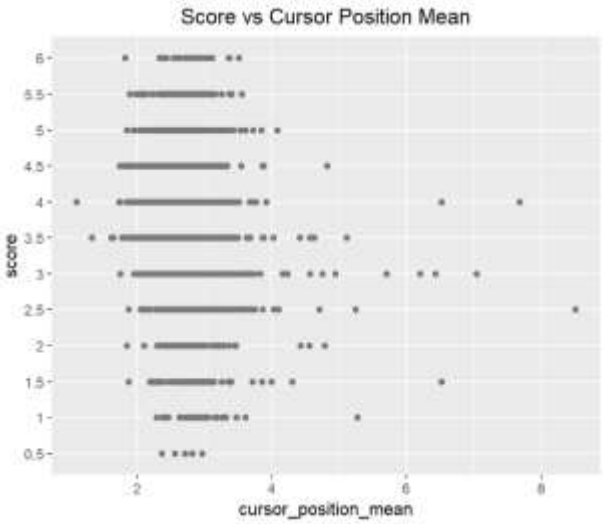
As seen in the scatterplot above, a discernable positive correlation between these two variables can be observed. Increased event diversity often does relate to higher quality writing as evidenced by the increasing scores. Once again, though, there is a lot of overlap in the bands of observations at each score level. There may be some predictive value in observing either very low diversity or very high diversity, but some middle show no discernible correlation as they can be observed in all bands of the score distribution.

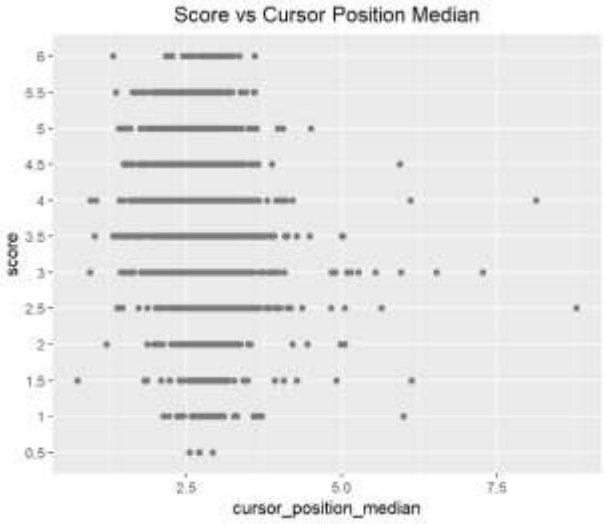
Uncorrelated Numeric Variables

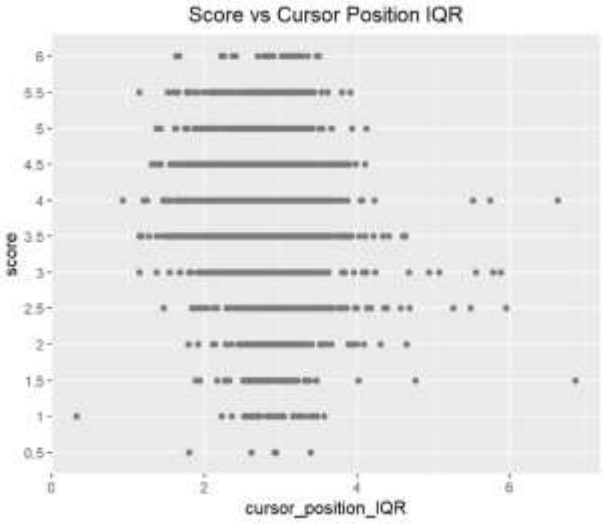
The following variables showed very little correlation on the correlation matrix, but individual scatterplots were generated to see if any patters stand out in the data that could be useful for further analysis

```
> ggplot(train_scores, aes(x = submission_time, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "submission_time", y = "score", title = "Score vs Submission Time") +
+   theme(plot.title = element_text(hjust = 0.5))
> ggplot(train_scores, aes(x = cursor_position_mean, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "cursor_position_mean", y = "score", title = "Score vs Cursor Position Mean") +
+   theme(plot.title = element_text(hjust = 0.5))
> ggplot(train_scores, aes(x = cursor_position_median, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "cursor_position_median", y = "score", title = "Score vs Cursor Position Median") +
+   theme(plot.title = element_text(hjust = 0.5))
> ggplot(train_scores, aes(x = cursor_position_IQR, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "cursor_position_IQR", y = "score", title = "Score vs Cursor Position IQR") +
+   theme(plot.title = element_text(hjust = 0.5))
```





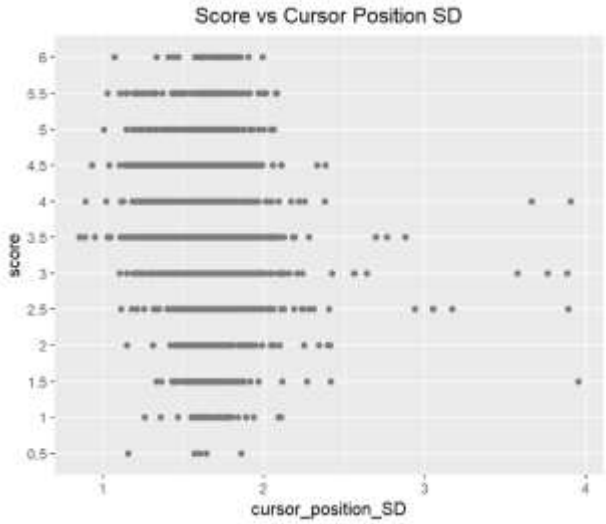


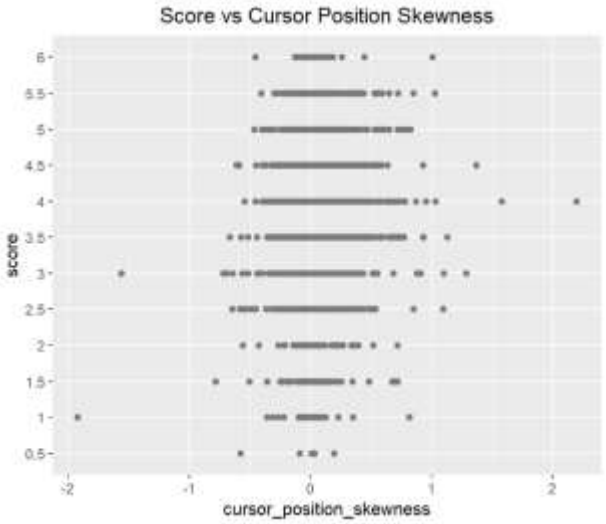


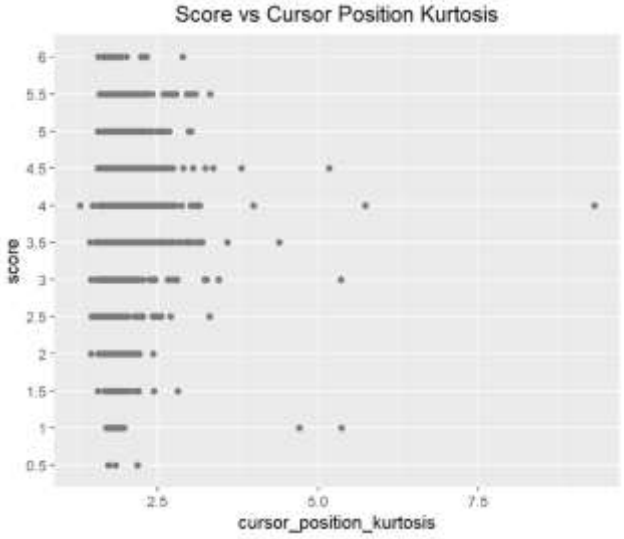
```
> ggplot(train_scores, aes(x = cursor_position_sd, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "cursor_position_SD", y = "score", title = "Score vs Cursor Position SD") +
+   theme(plot.title = element_text(hjust = 0.5))

> ggplot(train_scores, aes(x = cursor_position_skewness, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "cursor_position_skewness", y = "score", title = "Score vs Cursor Position Skewness") +
+   theme(plot.title = element_text(hjust = 0.5))

> ggplot(train_scores, aes(x = cursor_position_kurtosis, y = score)) +
+   geom_point(color = "#767B7D") +
+   labs(x = "cursor_position_kurtosis", y = "score", title = "Score vs Cursor Position Kurtosis") +
+   theme(plot.title = element_text(hjust = 0.5))
```





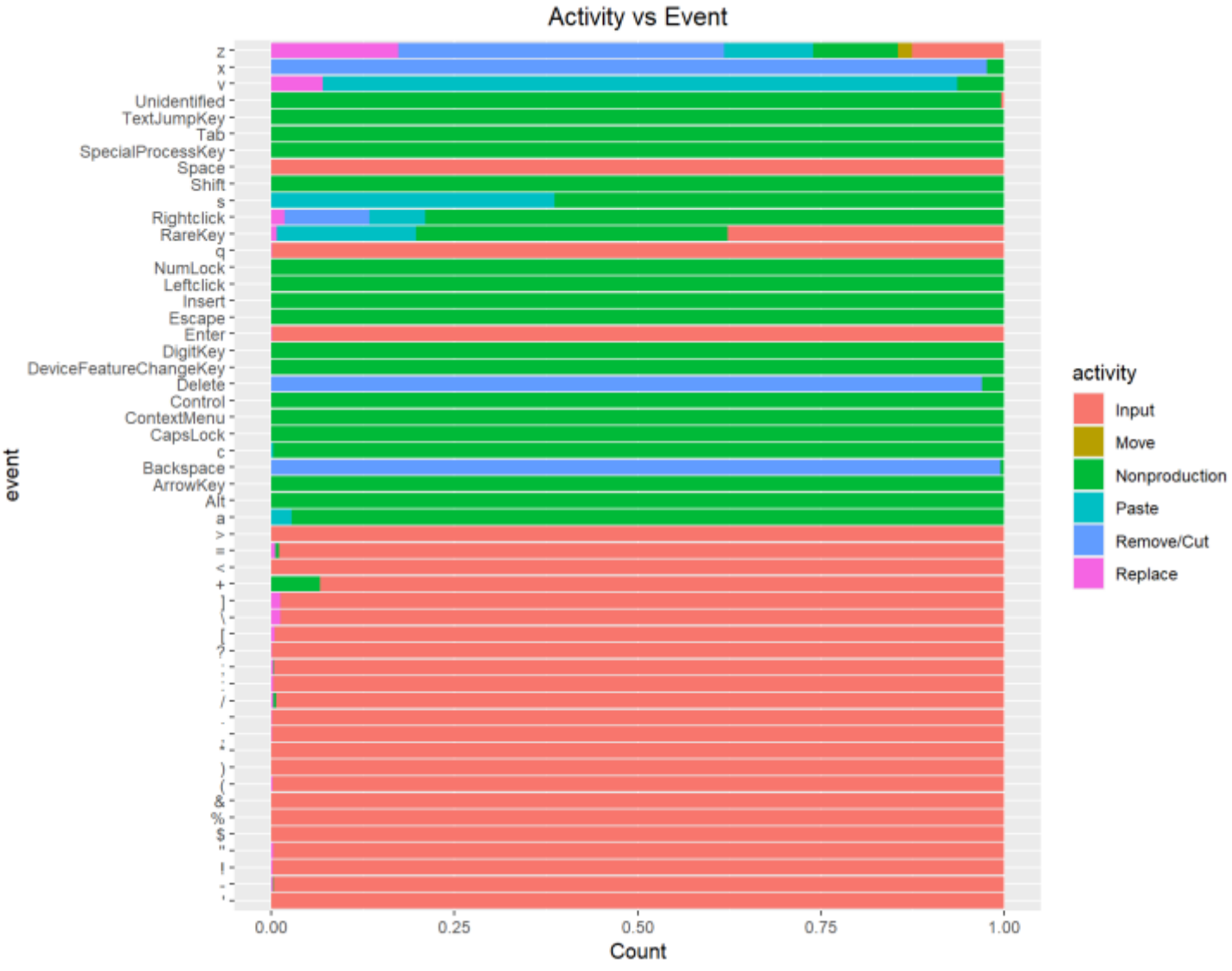


C.2 Categorical Variable Correlations

Activity vs. Event

The following bar graph was made to display the distribution of activity by event type. A heat map was not feasible because the overwhelming number of "q" events prevented a good visualization of the frequency. The following function was used to generate the table, which is useful in determining what keystrokes relate to specific activities.

```
> ggplot(train_logs, aes(x = event, fill = activity)) +
+   geom_bar(position = "fill") +
+   labs(x = "event", y = "Count", title = "Activity vs Event") +
+   theme(plot.title = element_text(hjust = 0.5)) +
+   coord_flip()
```

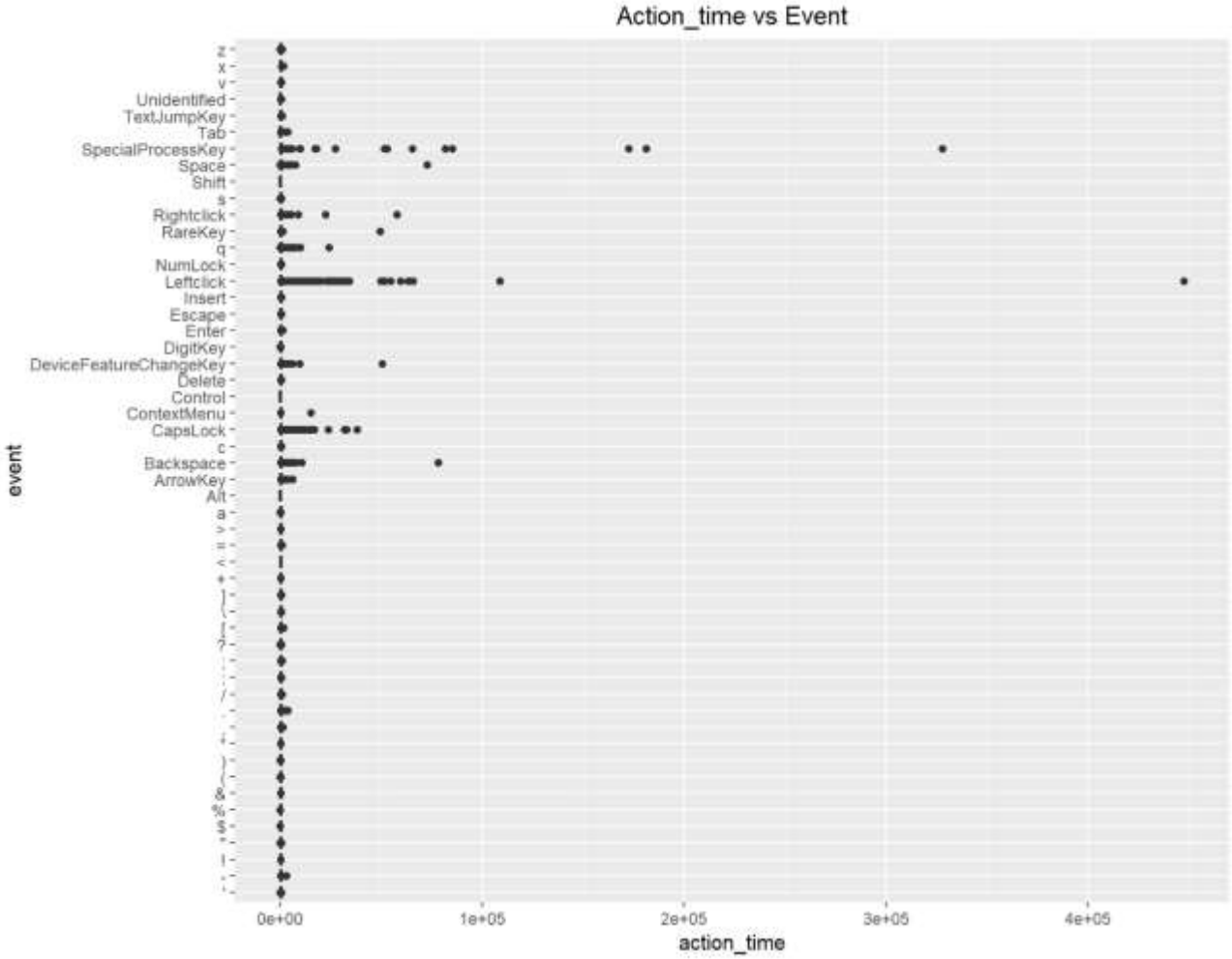


The bar graph on the page above does provide useful information for understanding the distribution of events by activity type. Many specific events are only observed in the context of one specific type of event. This may be useful for determining how the large number of unique events can be mapped to specific behaviors. Due to the complexity of this task, however, more advanced visualization methods will be needed to understand specifically how metrics can be derived from this raw data in a way that shows correlation with the dependent variable of score.

Event vs. Action Time

The *event* variable and the *action_time* variables were also compared to gain further insight into the distribution of those two variables. A graph of boxplots was generated using the ggplot function below.

```
> ggplot(train_logs, aes(x = event, y = action_time)) +
+   geom_boxplot() +
+   labs(x = "event", y = "action_time", title = "Action_time vs Event") +
+   theme(plot.title = element_text(hjust = 0.5)) +
+   coord_flip()
```

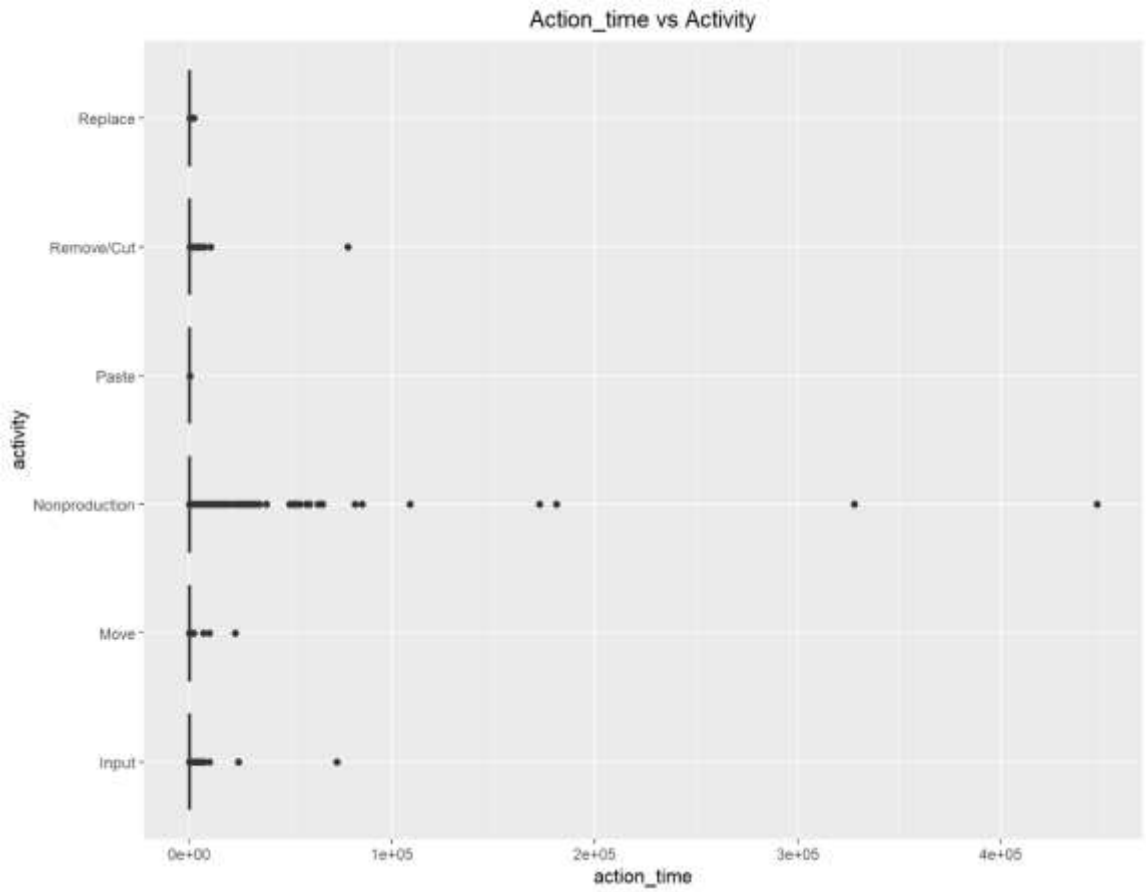


While the number of unique values and the existence of extreme outliers made this comparison difficult to visualize, we can see that a small number of specific events are responsible for the large number of outliers in action time. This information can help us determine how to best understand the presence of outliers.

Activity vs. Action Time

Since the events responsible seem to be mainly events that would be logged as “Nonproduction”, another visualization was made to analyze the distribution of action_time by activity. The ggplot function below was used to compare these variables with more focused boxplots.

```
> ggplot(train_logs, aes(x = activity, y = action_time)) +
+   geom_boxplot() +
+   labs(x = "activity", y = "action_time", title = "Action_time vs Activity") +
+   theme(plot.title = element_text(hjust = 0.5)) +
+   coord_flip()
```



As expected, this comparison does show that most of the extreme outliers represent “Nonproduction” events. While a deeper analysis of the distribution of specific values in the *event* or *activity* columns may provide insight into ways of splitting the data into more useful

groups. This would likely require more advanced machine learning techniques to do this well, and that is outside of the scope of this analysis and my current abilities.

Summary

The goal of this data analysis report was to determine the degree to which simple metrics of typing behavior can be used to predict scores on essay writing tasks. Through a detailed analysis of millions of observations, we derived numerous aggregated metrics in hopes of finding simple correlations between the observations and the scores that each participant received.

While some correlation was observed between these derived metrics and the score, more advanced methods can likely make better use of the data for achieving accurate predictive modeling. Even when using more advanced methods, it may be beneficial to incorporate metrics like *submitted_words*, *total_events*, and *event diversity*, which have been shown to have mild to moderate positive correlations to score.

In addition, observations made during this analysis can determine ways to improve the validity of similar studies in the future by an understanding of the issues that were encountered. A lack of consistent enforcement of the assignment requirements led to a large number of outliers. Specific measures can be taken to ensure that the time requirement is adhered to and that navigation to other windows is not allowed. In addition, consideration should be given to the incentives of the participants. Graded essays in an actual educational setting may produce a higher level of effort than a paid online task.

In conclusion, the data analysis conducted above has allowed us to better understand some of the basic behaviors that contribute to score, but more advanced analysis through the use of time-series analysis and neural networks could provide even more insight.

[1] Alex Franklin, Jules King, Maggie Demkin, Perpetual Baffour, Ryan Holbrook, Scott Crossley. (2023). Linking Writing Processes to Writing Quality. Kaggle. https://kaggle.com/competitions/linking-writing-processes-to-writing-quality