Quantitative Analysis

Michael Sun

January 14, 2024

Quantitative Data

In conducting your controlled usability study, you may have collected quantitative data, such as the time taken to complete a task. This guide will show you how to analyze quantitative data using R.

Note that the statistical tests done below assume that the data is parametric. That is, the data meets the following three criteria:

- **Independence**: each subject is sampled independently of every other subject, and measures from one subject are independent of measures on any other subject.
- **Normality**: data is normally distributed (technically, the error terms are normally distributed).
- **Homoskedasticity**: the variance is similar across all levels of factors.

If your data is not normally distributed (e.g., likert scale data), you should use non-parametric tests instead.

> **BEFORE YOU BEGIN**
> This guide uses R to read the data from an Excel file, generate plots, summary statistics, and conduct the analysis. Please install an IDE that supports R programming, such as RStudio or PyCharm with the R plugin.

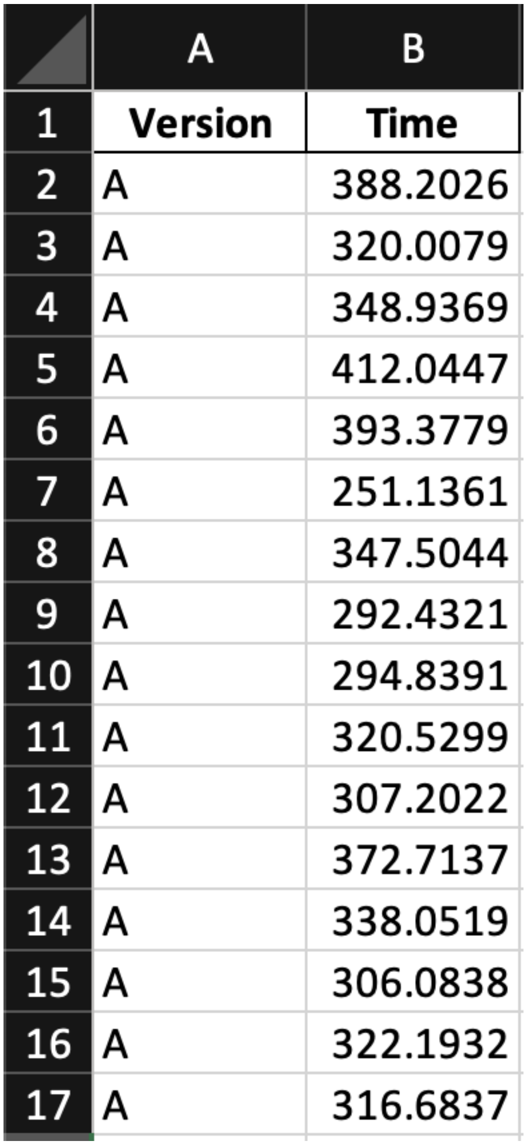
Step 1: Preparing Data

At this point, you should have a set of quantitative data for each participant.

The responses should be stored in an Excel file, with each row representing a respondent. Each dependent variable gets its own column, though how the design prototype is incorporated is dependent on whether you ran a between or within-subjects design experiment.

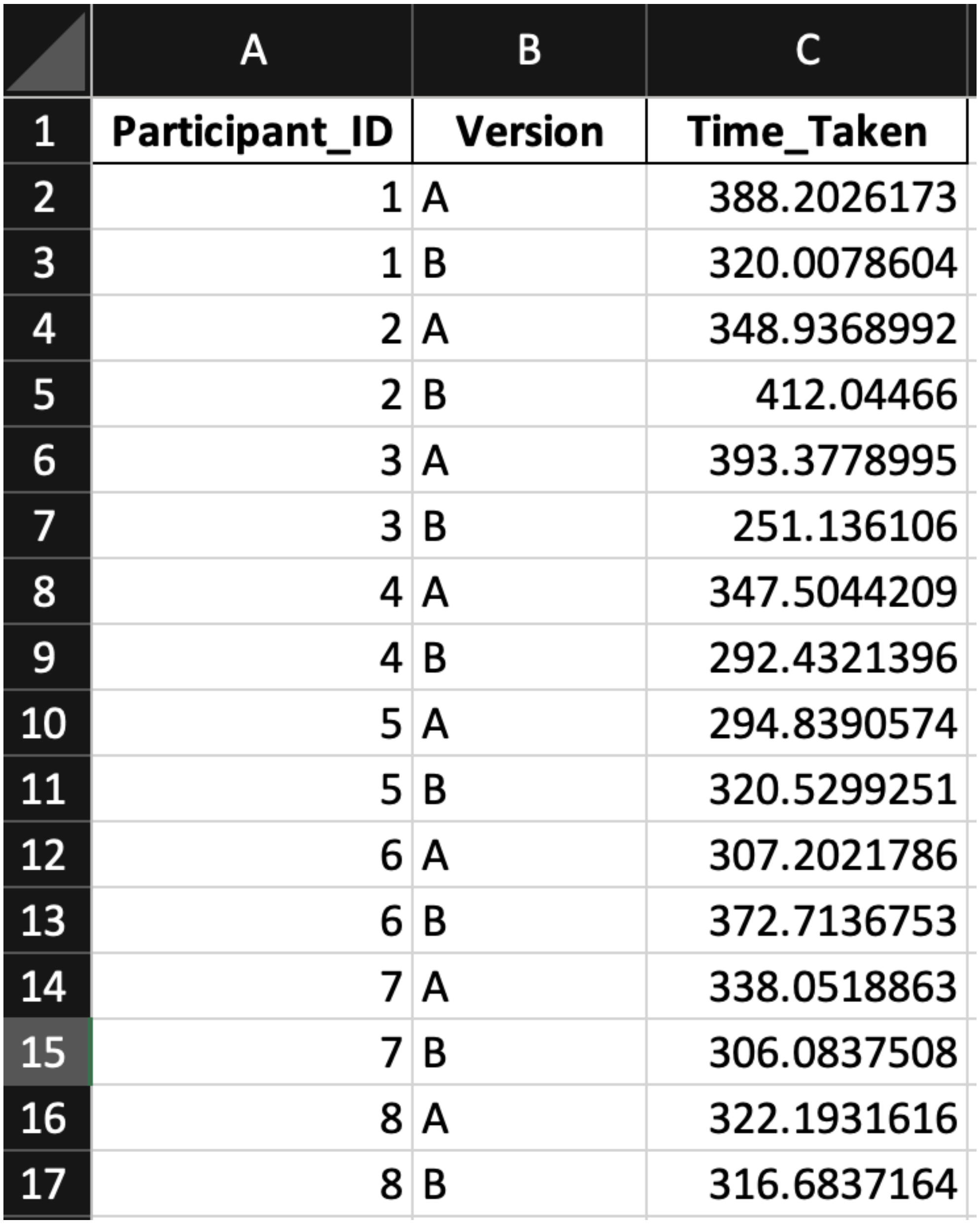
> **Experimental Design**
> * If you ran a **between-subjects** design, you only need a column for the prototype version.
> * If you ran a **within-subjects** design, you should have one column for the prototype version and another column for a user's identifier (e.g., participant ID).
>
> In either case, the remainder of your columns should correspond to each dependent variable that you measured.

This is an example of what your Excel file should look like for a **between-subjects** design:



	A	B
1	**Version**	**Time**
2	A	388.2026
3	A	320.0079
4	A	348.9369
5	A	412.0447
6	A	393.3779
7	A	251.1361
8	A	347.5044
9	A	292.4321
10	A	294.8391
11	A	320.5299
12	A	307.2022
13	A	372.7137
14	A	338.0519
15	A	306.0838
16	A	322.1932
17	A	316.6837

And for a **within-subjects** design:



	A	B	C
1	Participant_ID	Version	Time_Taken
2	1	A	388.2026173
3	1	B	320.0078604
4	2	A	348.9368992
5	2	B	412.04466
6	3	A	393.3778995
7	3	B	251.136106
8	4	A	347.5044209
9	4	B	292.4321396
10	5	A	294.8390574
11	5	B	320.5299251
12	6	A	307.2021786
13	6	B	372.7136753
14	7	A	338.0518863
15	7	B	306.0837508
16	8	A	322.1931616
17	8	B	316.6837164

Step 2: Loading Data into R

Now that you have your data in an Excel file, you can load it into R. The easiest way to do this is to use the `read_excel()` function from the `readxl` package.

If you haven't installed the package yet, be sure to uncomment the `install.packages()` line in the code chunk below.

```r
# install.packages("readxl")
library(readxl)

# Read the Excel file
time_data <- read_excel("../data/usability_study_quantitative_data_bs.xlsx")
```

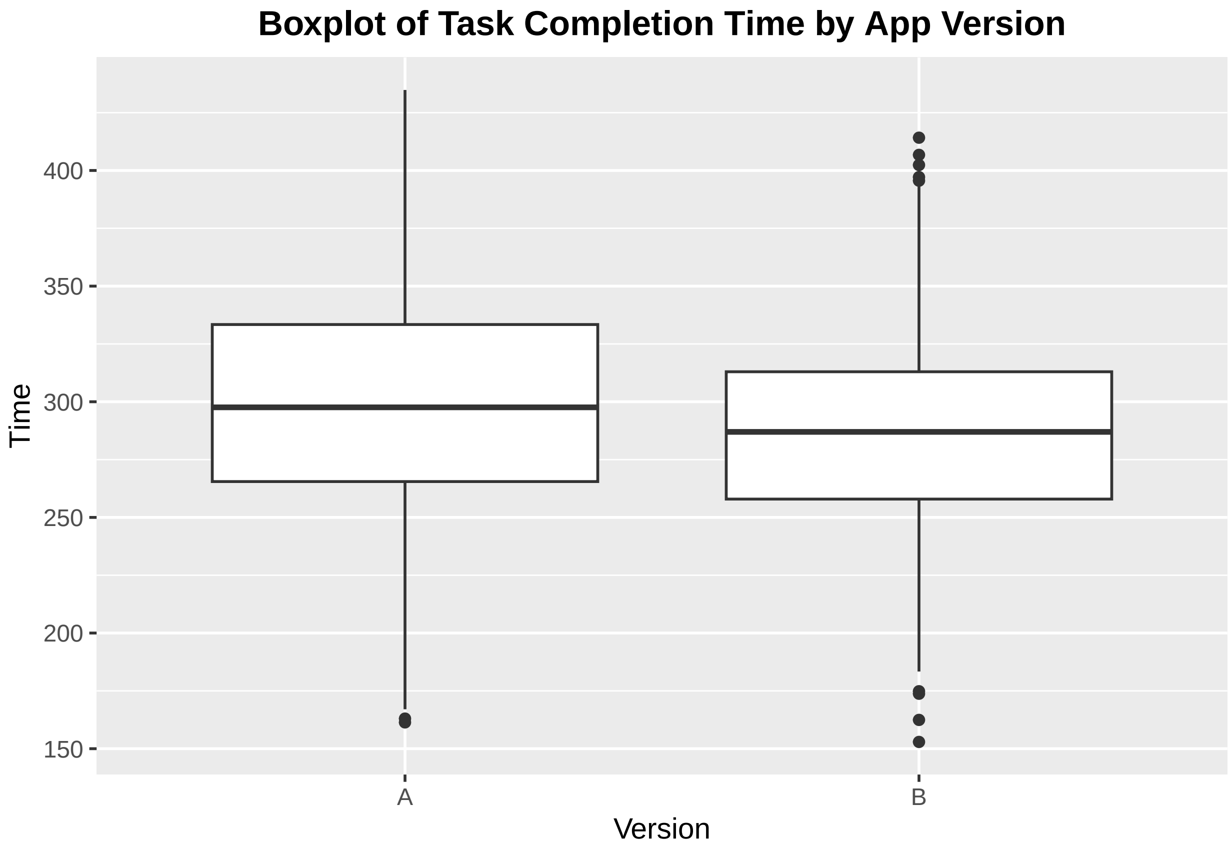
Step 3: Creating Plots

It may be useful to create plots to show the distribution of the completion time by version. You can use the `ggplot2` package to create plots.

Boxplot:

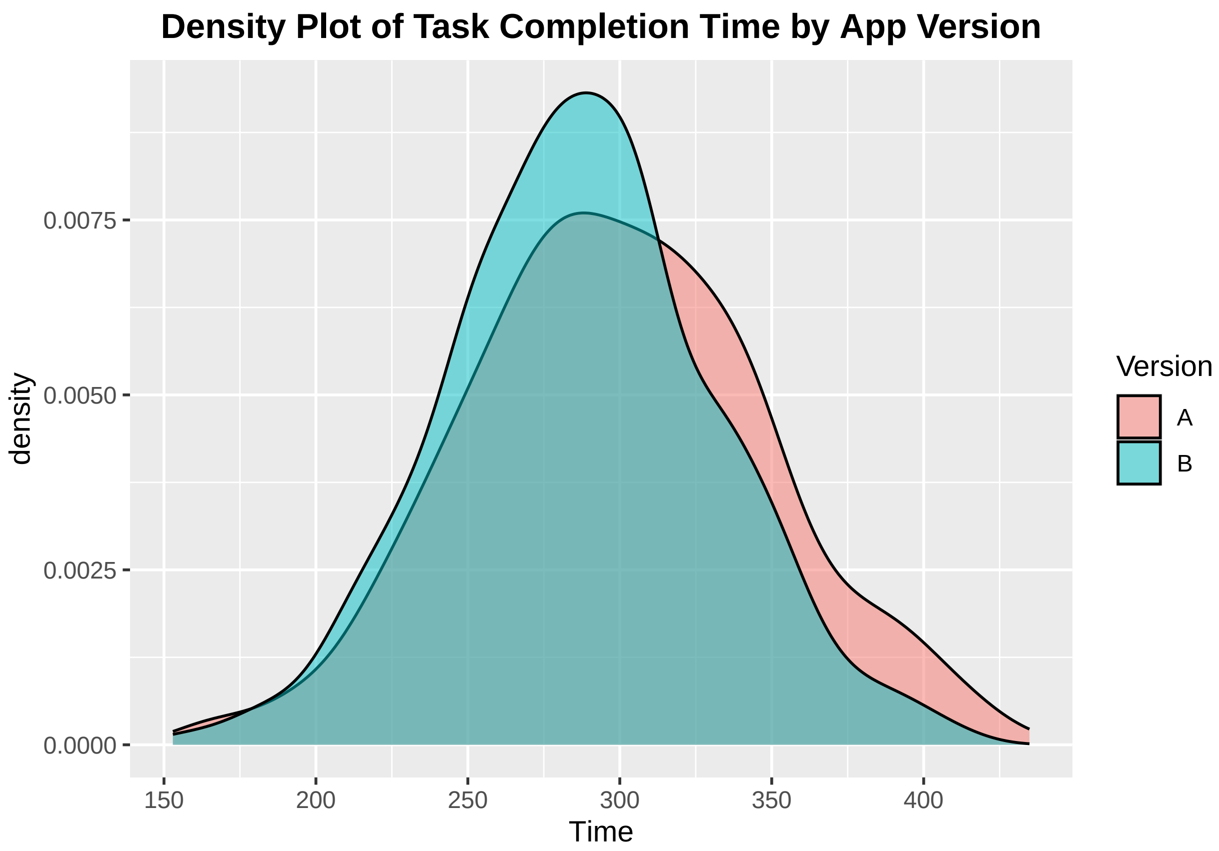
```r
# install.packages("ggplot2")
# install.packages("dplyr")
library(ggplot2)
library(tidyr)

ggplot(time_data, aes(x = Version, y = Time)) +
  geom_boxplot() +
  labs(title = "Boxplot of Task Completion Time by App Version") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```



Density plot:

```r
ggplot(time_data, aes(x = Time, fill = Version)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Task Completion Time by App Version") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```



Step 4: Creating Summary Statistics

You can also create summary statistics for each version, such as the mean, median, standard deviation, minimum, and maximum.

```
# Compute summary statistics for each version
summary_stats_time <- time_data %>%
  group_by(Version) %>%
  summarize(
    Count = n(),
    Mean = mean(Time),
    Median = median(Time),
    Standard_Deviation = sd(Time),
    Min = min(Time),
    Max = max(Time)
  )

# Print the summary statistics
print(summary_stats_time)
```

```
## # A tibble: 2 x 7
##   Version Count  Mean Median Standard_Deviation   Min   Max
##   <chr>   <int> <dbl>  <dbl>              <dbl> <dbl> <dbl>
## 1 A         500  299.   298.               50.0  161.  435.
## 2 B         500  287.   287.               43.9  153.  414.
```

Step 5: Running Statistical Tests

The statistical test you run depends on the experimental design you used. For within-subjects design, you should use the paired-sample t-test. For between-subjects design, you should use the two-sample t-test.

In general, we recommend the two-tailed t-test, which tests whether the means of two groups are statistically different from each other. The one-tailed t-test is used when you have a specific hypothesis about the direction of the difference between the two groups. For example, if you hypothesize that the new version of the app will take less time to complete a task, you would use a one-tailed t-test. But showing that there is a statistically significant difference in the completion time is sufficient for most usability studies, especially as the mean completion time can be determined from the summary statistics.

Within-subjects:

```r
# Perform a two-sample two-tailed t-test
t_test_result <- t.test(Time ~ Version, data = time_data)

# Print the results
print(t_test_result)
```

```
##
##  Welch Two Sample t-test
##
## data:  Time by Version
## t = 3.9204, df = 981.98, p-value = 9.452e-05
## alternative hypothesis: true difference in means between group A and group B is not equal to 0
## 95 percent confidence interval:
##   5.825792 17.503072
## sample estimates:
## mean in group A mean in group B
##        298.7323        287.0678
```

For a **between-subjects** design, you can use the 2-sample t-test to compare the means of two groups. See the usability_quantitative_single_bs.R page on GitHub for more information.