

Likert Scale Analysis

Michael Sun

January 14, 2024

Likert Scales

In the realm of UI/UX research, Likert scales play a pivotal role in quantitatively capturing opinions, attitudes, and responses to various questions. These scales, typically ranging from “strongly disagree” to “strongly agree,” offer a simple yet powerful way for respondents to express their level of agreement with a given statement. The versatility of Likert scales makes them a staple in HCI.

For a list of Likert scale examples, see [here](#).

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 survey responses to one or more Likert scale questions.

The responses should be stored in an Excel file, with each row representing a respondent. Each question 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 Likert scale question.

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

	A	B	C	D	E	F	G	H
1	Version	Question_1	Question_2	Question_3	Question_4	Question_5	Question_6	Question_7
2	A	5	4	2	5	4	4	5
3	A	3	4	1	4	5	3	2
4	A	4	3	3	3	4	2	3
5	A	5	3	2	3	4	3	3
6	A	5	3	3	2	2	2	5
7	A	2	3	2	3	4	4	3
8	A	4	4	2	3	3	5	2
9	A	3	1	3	3	4	4	2
10	A	3	3	4	2	4	3	3
11	A	3	3	4	4	1	3	5
12	A	3	2	4	3	3	4	3
13	A	5	5	2	3	3	3	4
14	A	4	3	4	3	3	3	3
15	A	3	5	4	2	1	1	3
16	A	3	2	3	2	2	2	5

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

	A	B	C	D	E	F	G	H	I
1	Participant_ID	Version	Question_1	Question_2	Question_3	Question_4	Question_5	Question_6	Question_7
2	1 A	5	4	2	5	4	4	4	5
3	1 B	3	3	1	3	4	2	2	3
4	2 A	3	4	1	4	5	3	2	2
5	2 B	3	3	3	4	3	3	2	2
6	3 A	4	3	3	3	4	2	3	3
7	3 B	4	3	5	3	3	2	4	4
8	4 A	5	3	2	3	4	3	3	3
9	4 B	3	2	2	4	3	3	3	3
10	5 A	5	3	3	2	2	2	5	5
11	5 B	3	2	3	2	3	3	4	4
12	6 A	2	3	2	3	4	4	3	3
13	6 B	2	3	2	3	4	2	2	2
14	7 A	4	4	2	3	3	5	2	2
15	7 B	1	3	4	5	2	3	2	2
16	8 A	3	1	3	3	4	4	2	2
17	8 B	1	3	1	2	1	1	2	2

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.

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

likert_data_multiple_ws <- read_excel("../data/usability_study_likert_within_subjects_data.xlsx")
```

Step 3: Creating Plots

It may be useful to create some bar graphs to show the distribution of Likert scores by version and by question.

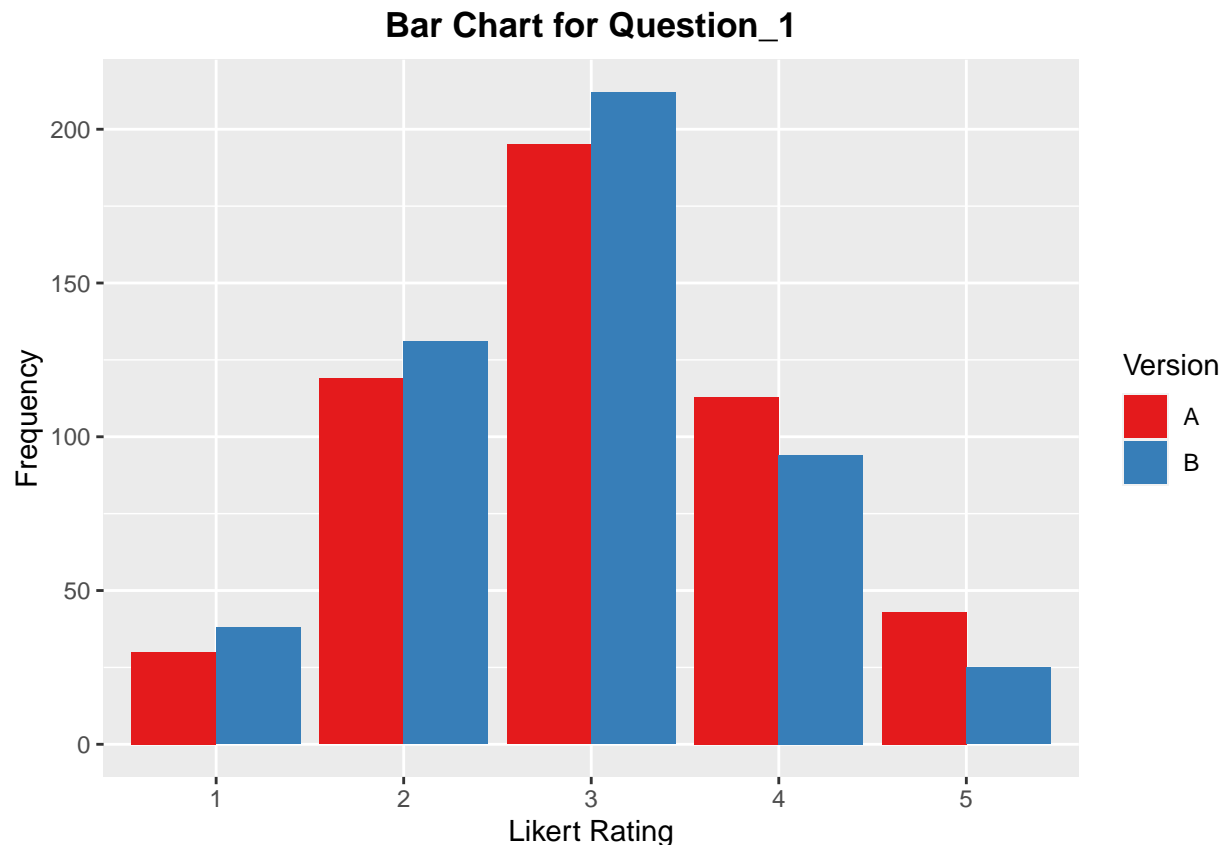
```
# install.packages("ggplot2")
# install.packages("dplyr")
library(ggplot2)
library(tidyr)

# Reshape the data to long format
data_long <- pivot_longer(likert_data_multiple_ws,
                          cols = starts_with("Question"),
                          names_to = "Question",
                          values_to = "Likert_Rating")

# Function to create a bar chart for a given question
plot_question_versions <- function(question) {
  filtered_data <- filter(data_long, Question == question)

  ggplot(filtered_data, aes(x = as.factor(Likert_Rating), fill = Version)) +
    geom_bar(position = position_dodge()) +
    scale_fill_brewer(palette = "Set1", name = "Version") +
    labs(title = paste("Bar Chart for", question),
         x = "Likert Rating",
         y = "Frequency") +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"))
}

# Get unique questions
unique_questions <- unique(data_long$Question)
print(plot_question_versions(unique_questions[1]))
```



If you want to get an individual plot for each question, you can use a for loop to iterate through each question and call the `plot_question_versions()` function.

```
# Loop through each question and plot
for (question in unique_questions) {
  print(plot_question_versions(question))
}
```

Step 4: Creating Summary Statistics

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

```
# Compute summary statistics for each version and question
summary_stats_likert_multiple <- data_long %>%
  group_by(Question, Version) %>%
  summarize(
    Count = n(),
    Mean = mean(Likert_Rating),
    Median = median(Likert_Rating),
    Standard_Deviation = sd(Likert_Rating),
    Min = min(Likert_Rating),
    Max = max(Likert_Rating)
  )
```

'summarise()' has grouped output by 'Question'. You can override using the

```
## '.groups' argument.
```

```
# Print the summary statistics
options(dplyr.print_max = Inf)
print(summary_stats_likert_multiple)
```

```
## # A tibble: 40 x 8
## # Groups:   Question [20]
##   Question   Version Count   Mean Median Standard_Deviation   Min   Max
##   <chr>       <chr>   <int> <dbl>  <dbl>          <dbl> <dbl> <dbl>
## 1 Question_1   A         500  3.04    3          1.02    1    5
## 2 Question_1   B         500  2.87    3          0.970    1    5
## 3 Question_10  A         500  3.40    3          0.991    1    5
## 4 Question_10  B         500  2.54    3          1.00    1    5
## 5 Question_11  A         500  3.56    4          0.967    1    5
## 6 Question_11  B         500  2.54    2          1.03    1    5
## 7 Question_12  A         500  3.49    4          0.980    1    5
## 8 Question_12  B         500  2.43    2          1.00    1    5
## 9 Question_13  A         500  3.51    3          0.984    1    5
## 10 Question_13 B         500  2.39    2          0.967    1    5
## 11 Question_14  A         500  3.66    4          0.935    1    5
## 12 Question_14  B         500  2.38    2          0.952    1    5
## 13 Question_15  A         500  3.76    4          0.898    1    5
## 14 Question_15  B         500  2.30    2          0.926    1    5
## 15 Question_16  A         500  3.73    4          0.978    1    5
## 16 Question_16  B         500  2.28    2          0.931    1    5
## 17 Question_17  A         500  3.94    4          0.901    1    5
## 18 Question_17  B         500  2.23    2          0.954    1    5
## 19 Question_18  A         500  3.75    4          0.971    1    5
## 20 Question_18  B         500  2.19    2          0.943    1    5
## 21 Question_19  A         500  3.89    4          0.903    1    5
## 22 Question_19  B         500  2.08    2          0.898    1    5
## 23 Question_2   A         500  3.15    3          0.978    1    5
## 24 Question_2   B         500  2.89    3          1.00    1    5
## 25 Question_20  A         500  3.99    4          0.878    1    5
## 26 Question_20  B         500  2.1     2          0.919    1    5
## 27 Question_3   A         500  3.14    3          1.01    1    5
## 28 Question_3   B         500  2.8     3          0.966    1    5
## 29 Question_4   A         500  3.12    3          1.00    1    5
## 30 Question_4   B         500  2.78    3          1.03    1    5
## 31 Question_5   A         500  3.22    3          1.02    1    5
## 32 Question_5   B         500  2.8     3          1.00    1    5
## 33 Question_6   A         500  3.26    3          0.990    1    5
## 34 Question_6   B         500  2.70    3          0.960    1    5
## 35 Question_7   A         500  3.31    3          1.04    1    5
## 36 Question_7   B         500  2.70    3          0.970    1    5
## 37 Question_8   A         500  3.33    3          0.967    1    5
## 38 Question_8   B         500  2.57    3          0.994    1    5
## 39 Question_9   A         500  3.39    3          0.975    1    5
## 40 Question_9   B         500  2.57    3          0.977    1    5
```

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 Wilcoxon signed rank test. For between-subjects design, you should use the Wilcoxon rank sum test.

The Wilcoxon signed ranked test is a non-parametric test that compares two related samples. The Wilcoxon rank sum test (a.k.a. the Mean-Whitney U test) is a non-parametric test that compares two independent samples. These tests are non-parametric tests—used when the data is not normally distributed or when the sample size is small. Such is the case with Likert scale data.

Within-subjects:

```
# Initializing a dataframe to store the results
test_results <- data.frame(Question = character(),
                           Wilcoxon_Statistic = numeric(),
                           P_Value = numeric(),
                           stringsAsFactors = FALSE)

data_wide <- data_long %>%
  pivot_wider(names_from = Version, values_from = Likert_Rating)

# Looping through each question to perform the test
for (q in unique(data_long$Question)) {
  # Extracting the paired responses for each version
  responses <- filter(data_wide, Question == q)

  # Performing the Wilcoxon signed-rank test
  test <- wilcox.test(responses$`A`, responses$`B`, paired = TRUE)

  # Storing the results
  test_results <- rbind(test_results,
                        data.frame(Question = q,
                                   Wilcoxon_Statistic = test$statistic,
                                   P_Value = test$p.value))
}

# Displaying the test results
print(test_results)
```

##	Question	Wilcoxon_Statistic	P_Value
## V	Question_1	35867.0	1.757328e-02
## V1	Question_2	39954.5	1.007498e-04
## V2	Question_3	45495.0	7.592614e-08
## V3	Question_4	44448.0	7.418933e-07
## V4	Question_5	41782.0	8.047282e-11
## V5	Question_6	50335.0	1.671821e-16
## V6	Question_7	50244.5	2.395226e-20
## V7	Question_8	58378.0	5.517323e-27
## V8	Question_9	56533.0	1.509075e-30
## V9	Question_10	66626.0	5.275079e-31
## V10	Question_11	69713.5	8.375511e-39
## V11	Question_12	67279.5	2.602166e-41
## V12	Question_13	73528.5	9.147343e-43
## V13	Question_14	76277.5	6.002662e-53

## V14 Question_15	81107.0	8.814582e-60
## V15 Question_16	84973.0	8.156181e-58
## V16 Question_17	92695.0	3.100225e-68
## V17 Question_18	92430.5	2.289302e-62
## V18 Question_19	94250.5	2.055136e-69
## V19 Question_20	98981.0	4.808506e-72

For a **between-subjects** design, you can use the Wilcoxon rank sum test. See the GitHub page for more information.