Solution for the assignment of the fifth class

Kovacs Marton

9/29/2021

Importing data

```
processed <- read_tsv("data/boldog_processed.tsv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_double(),
##   neme = col_character(),
##   isk = col_character()
## )
## i Use `spec()` for the full column specifications.
```

Data exploration

```
skimr::skim(processed) %>%
  kable()
```

skim_type	skim_variable	n_missing	complete_rate	character.min	character.max	character.empty	char
character	neme	0	1.000	2	5	0	
character	isk	0	1.000	7	11	0	
numeric	index	0	1.000	NA	NA	NA	
numeric	eletkora	0	1.000	NA	NA	NA	
numeric	gyermeke	0	1.000	NA	NA	NA	
numeric	anyagi	0	1.000	NA	NA	NA	
numeric	p_elmeny_percent	0	1.000	NA	NA	NA	
numeric	testi_fi	3	0.994	NA	NA	NA	
numeric	alt_lelki	4	0.992	NA	NA	NA	
numeric	alt_eg_all	7	0.986	NA	NA	NA	
numeric	fizero	6	0.988	NA	NA	NA	
numeric	arcocska	0	1.000	NA	NA	NA	
numeric	aggodalo	0	1.000	NA	NA	NA	
numeric	ideges	0	1.000	NA	NA	NA	
numeric	feszult	0	1.000	NA	NA	NA	
numeric	nyugtala	0	1.000	NA	NA	NA	
numeric	diener1	0	1.000	NA	NA	NA	
numeric	diener2	0	1.000	NA	NA	NA	
numeric	diener3	0	1.000	NA	NA	NA	
numeric	diener4	0	1.000	NA	NA	NA	
numeric	diener5	0	1.000	NA	NA	NA	
numeric	diener6	0	1.000	NA	NA	NA	
numeric	diener7	0	1.000	NA	NA	NA	
numeric	diener8	0	1.000	NA	NA	NA	
numeric	jollet	0	1.000	NA	NA	NA	
numeric	savor	0	1.000	NA	NA	NA	
numeric	a_vhat	0	1.000	NA	NA	NA	
numeric	onreg	0	1.000	NA	NA	NA	
numeric	rezil	0	1.000	NA	NA	NA	
numeric	m_flow	0	1.000	NA	NA	NA	
numeric	g_jerz	0	1.000	NA	NA	NA	
numeric	g_jpszi	0	1.000	NA	NA	NA	
numeric	g_jszoc	0	1.000	NA	NA	NA	
numeric	g_jspir	0	1.000	NA	NA	NA	
numeric	g_jerzpsz	0	1.000	NA	NA	NA	
numeric	g_jspszoc	0	1.000	NA	NA	NA	
numeric	pik_mm	0	1.000	NA	NA	NA	
numeric	pik_av	0	1.000	NA	NA	NA	
numeric	pik_onr	0	1.000	NA	NA	NA	
numeric	pik_rez	0	1.000	NA	NA	NA	
numeric	p_poz_erz	0	1.000	NA	NA	NA	
numeric	p_elmely	0	1.000	NA	NA	NA	
numeric	p_poz_kapc	0	1.000	NA	NA	NA	
numeric	p_ert_cel	0	1.000	NA	NA	NA	
numeric	p_telj	0	1.000	NA	NA	NA	
numeric	p_boldog	0	1.000	NA	NA	NA	
numeric	p_egeszs	0	1.000	NA	NA	NA	
numeric	p_neg_erz	0	1.000	NA	NA	NA	
numeric	p_magany	0	1.000	NA	NA	NA	
numeric	perma	0	1.000	NA	NA	NA	

1. Which scales differ the most along the four levels of education if we treat them as ordinal scales? How big is the eta-square for each comparison?

For these comparisons I will keep only the ordinal scales, and not the interval scales. I will label a scale as ordinal if there is a low number of levels (less than 10) and the scale only consists of whole numbers.

Therefore, the scales that I will include in this analysis are the following:

```
ordinal_vars <- select(processed, 8:24) %>% names()
cat(paste('-', ordinal_vars), sep = '\n')
```

- testi_fi
- alt_lelki
- alt_eg_all
- fizero
- arcocska
- aggodalo
- ideges
- feszult
- nyugtala
- diener1
- diener2
- diener3
- diener4
- diener5
- diener6
- diener7
- diener8

As education has 4 different levels, I will run a Kruskall-Wallis test to investigate the differences between the groups on each scale.

First, I select only the before mentioned variables and convert the educational level variable to a factor variable.

I will also drop all the participants where there are at least a missing value in any of the investigated scales. The number of participants with any missing data 17.

```
ordinal_data <-
  processed %>%
  select(index, 8:24, isk, neme) %>%
  mutate(isk = as.factor(isk),
         index = as.factor(index),
         neme = as.factor(neme)) %>%
  filter_all(all_vars(!is.na(.)))

# Checking levels for isk as a factor variable
# The order does not matter in this case as this is an omnibus test
levels(ordinal_data$isk)
```

```
## [1] "altalanos"   "egyetem"     "foiskola"    "kozepiskola"
```

Then, I will run the analysis iteratively on each variable. I will also compute the summary statistics for each scale as well.

I will also calculate the eta squared for the Kruskall Wallis test. The eta squared will be based on the H statistics (sometimes called epsilon squared). I will also add the possibility calculate the CIs by using a boostrapping method.

```
# Function to run the Kruskall Wallis test
kruskall_iterate <- function(data, scale_var) {
      kruskal.test(data[[scale_var]] ~ data[["isk"]])
}

# Summary stats function
ordinal_summary_stats <- function(data, scale_var) {
  data %>%
    group_by(isk) %>%
    summarise(median = median(.data[[scale_var]], na.rm = TRUE),
              iqr = IQR(.data[[scale_var]], na.rm = TRUE),
              n = n()
              )
}

# Function to calculate the effect size
h_eta_square <- function(data, scale_var, ci) {
      rstatix::kruskal_effsize(data, data[[scale_var]] ~ data[["isk"]], ci = ci)
}

# Run the analysis
ordinal_res <-
  tibble::tibble(
    variable = ordinal_vars,
    kruskal_res = map(variable,
                      ~ kruskall_iterate(data = ordinal_data, scale_var = .x)),
    statistic = map_dbl(kruskal_res,  ~ pluck(.x, "statistic", 1)),
    p_value = map_dbl(kruskal_res,  ~ pluck(.x, "p.value")),
    df = map_dbl(kruskal_res,  ~ pluck(.x, "parameter", 1)),
    summary_stats = map(variable,
                        ~ ordinal_summary_stats(
                          data = ordinal_data,
                          scale_var = .x
                          )),
    effect_size_res = map(variable,
                          ~ h_eta_square(
                              data = ordinal_data,
                              scale_var = .x,
                              FALSE)
                          ),
    eta_squared = map_dbl(effect_size_res, ~pluck(.x, "effsize"))
  )
```

```
ordinal_summary <-
  ordinal_res %>%
  select(variable, summary_stats) %>%
  mutate(summary_stats = map2(variable, summary_stats, ~mutate(.y, variable = .x) %>% select(variable,
```

```
map_df(ordinal_summary$summary_stats, bind_rows) %>%
  kable(
    format = "latex",
    col.names = c("Variable", "Level of Education", "Median", "IQR", "N"),
    align = c("l", "c", "c", "c", "c"),
    caption = "Summary Statistics for the Investigated Scales"
  ) %>%
  row_spec(row = 0, align = "c") %>%
  kable_styling(full_width = TRUE)
```

```
ordinal_res %>%
  select(variable, statistic, p_value, eta_squared) %>%
  arrange(desc(eta_squared)) %>%
  kable(
    format = "latex",
    col.names = c("Variable", "Test statistic", "P value", "Rank Eta Squared"),
    align = c("l", "c", "c"),
    caption = "Results of the Kruskall Wallis Tests"
  ) %>%
  row_spec(row = 0, align = "c") %>%
  kable_styling(full_width = TRUE) %>%
  add_footnote("Results are arranged in descending order by the size of the effec size.")
```

The difference was the biggest for the *diener1* scale with an effect size of 0.0363684.

2. Which scales differ the most along the three levels of education if we treat them as ordinal scales? How big is the eta-square for each comparison?

The N for one level of education (elementary school) is quite low, so it make sense to group participants belonging to elementary and highschool. After relabelling the data we will take a look at the differences on these ordinal scales again.

```
# Merge the levels
ordinal_merged_data <-
  processed %>%
  select(index, 8:24, isk, neme) %>%
  mutate(isk = case_when(isk == "altalanos" ~ "altalanos_es_kozepiskola",
                         isk == "kozepiskola" ~ "altalanos_es_kozepiskola",
                         TRUE ~ isk),
         isk = as.factor(isk),
         index = as.factor(index),
         neme = as.factor(neme)) %>%
  filter_all(all_vars(!is.na(.)))

# Run the analysis
ordinal_merged_res <-
  tibble::tibble(
```

Table 1: Summary Statistics for the Investigated Scales

Variable	Level of Education	Median	IQR	N
testi_fi	altalanos	4.0	1.00	10
testi_fi	egyetem	4.0	1.00	117
testi_fi	foiskola	4.0	1.00	156
testi_fi	kozepiskola	4.0	1.00	200
alt_lelki	altalanos	4.0	1.75	10
alt_lelki	egyetem	4.0	1.00	117
alt_lelki	foiskola	4.0	1.00	156
alt_lelki	kozepiskola	4.0	2.00	200
alt_eg_all	altalanos	4.0	2.75	10
alt_eg_all	egyetem	4.0	2.00	117
alt_eg_all	foiskola	4.0	1.00	156
alt_eg_all	kozepiskola	4.0	2.00	200
fizero	altalanos	4.5	2.75	10
fizero	egyetem	4.0	2.00	117
fizero	foiskola	4.0	2.00	156
fizero	kozepiskola	4.0	2.00	200
arcocska	altalanos	3.0	1.75	10
arcocska	egyetem	2.0	1.00	117
arcocska	foiskola	2.0	1.00	156
arcocska	kozepiskola	3.0	1.00	200
aggodalo	altalanos	4.0	2.75	10
aggodalo	egyetem	2.0	1.00	117
aggodalo	foiskola	2.0	2.00	156
aggodalo	kozepiskola	3.0	2.00	200
ideges	altalanos	3.0	2.50	10
ideges	egyetem	2.0	2.00	117
ideges	foiskola	2.0	2.25	156
ideges	kozepiskola	2.0	2.25	200
feszult	altalanos	3.0	2.50	10
feszult	egyetem	2.0	1.00	117
feszult	foiskola	2.0	2.00	156
feszult	kozepiskola	3.0	2.00	200
nyugtala	altalanos	2.0	1.75	10
nyugtala	egyetem	2.0	2.00	117
nyugtala	foiskola	2.0	2.25	156
nyugtala	kozepiskola	2.0	2.25	200
diener1	altalanos	4.0	3.00	10
diener1	egyetem	6.0	2.00	117
diener1	foiskola	6.0	1.25	156
diener1	kozepiskola	6.0	2.00	200
diener2	altalanos	4.0	0.75	10
diener2	egyetem	6.0	2.00	117
diener2	foiskola	6.0	1.00	156
diener2	kozepiskola	5.0	2.00	200
diener3	altalanos	5.0	2.00	10
diener3	egyetem	6.0	1.00	117
diener3	foiskola	6.0	2.00	156
diener3	kozepiskola	6.0	1.00	200
diener4	altalanos	4.5	2.00	10
diener4	egyetem	6.0	1.00	117
diener4	foiskola	6.0	1.00	156
diener4	kozepiskola	6.0	2.00	200
diener5	altalanos	5.5	2.75	10
diener5	egyetem	6.0	1.00	117
diener5	foiskola	6.0	2.00	156
diener5	kozepiskola	6.0	2.00	200

Table 2: Results of the Kruskall Wallis Tests

Variable	Test statistic	P value	Rank Eta Squared
diener1	20.4204711	0.0001389	0.0363684
diener3	13.8364731	0.0031364	0.0226231
diener2	10.4302177	0.0152418	0.0155119
alt_lelki	9.0422632	0.0287342	0.0126143
feszult	7.4587522	0.0586278	0.0093085
ideges	5.8165889	0.1208826	0.0058801
diener8	5.8061269	0.1214331	0.0058583
alt_eg_all	5.7411903	0.1249037	0.0057227
testi_fi	4.9624243	0.1745694	0.0040969
diener6	4.8123828	0.1860623	0.0037837
diener4	4.1545514	0.2452517	0.0024103
diener7	3.9081014	0.2715600	0.0018958
diener5	3.6501346	0.3018058	0.0013573
nyugtala	3.5806032	0.3104576	0.0012121
aggodalo	3.3188090	0.3450333	0.0006656
arcocska	1.2631723	0.7378962	-0.0036259
fizero	0.6287284	0.8898253	-0.0049505

[a] Results are arranged in descending order by the size of the effec size.

```
    variable = ordinal_vars,
    kruskal_res = map(variable,
                      ~ kruskall_iterate(data = ordinal_merged_data, scale_var = .x)),
    statistic = map_dbl(kruskal_res,  ~ pluck(.x, "statistic", 1)),
    p_value = map_dbl(kruskal_res,  ~ pluck(.x, "p.value")),
    df = map_dbl(kruskal_res,  ~ pluck(.x, "parameter", 1)),
    summary_stats = map(variable, ~
                          ordinal_summary_stats(
                            data = ordinal_merged_data,
                            scale_var = .x
                          )),
    effect_size_res = map(variable,
                          ~ h_eta_square(
                            data = ordinal_merged_data,
                            scale_var = .x,
                            FALSE)),
    eta_squared = map_dbl(effect_size_res, ~pluck(.x, "effsize"))
  )
```

```
ordinal_merged_res %>%
  select(variable, statistic, p_value, eta_squared) %>%
  arrange(desc(eta_squared)) %>%
  kable(
    format = "latex",
    col.names = c("Variable", "Test statistic", "P value", "Rank Eta Squared"),
    align = c("l", "c", "c"),
    caption = "Results of the Kruskall Wallis Tests Comparing the Three Levels of Education on Each Inv
  ) %>%
  row_spec(row = 0, align = "c") %>%
  kable_styling(full_width = TRUE) %>%
```

Table 3: Results of the Kruskall Wallis Tests Comparing the Three Levels of Education on Each Investigated Ordinal Scale

Variable	Test statistic	P value	Rank Eta Squared
diener1	16.0478127	0.0003275	0.0292663
diener3	12.4855516	0.0019445	0.0218449
alt_lelki	8.6981290	0.0129189	0.0139544
feszult	7.4553790	0.0240483	0.0113654
diener2	5.8848006	0.0527390	0.0080933
ideges	5.6772051	0.0585074	0.0076608
alt_eg_all	4.6780349	0.0964223	0.0055792
diener8	3.7551022	0.1529642	0.0036565
diener6	3.6030753	0.1650449	0.0033397
diener7	3.5755755	0.1673299	0.0032824
nyugtala	3.3742334	0.1850523	0.0028630
diener4	3.0574740	0.2168093	0.0022031
aggodalo	3.0193066	0.2209866	0.0021236
diener5	2.8155450	0.2446877	0.0016991
testi_fi	2.5516560	0.2791997	0.0011493
arcocska	0.9784639	0.6130971	-0.0021282
fizero	0.0668622	0.9671216	-0.0040274

[a] Results are arranged in descending order by the size of the effec size.

```
  add_footnote("Results are arranged in descending order by the size of the effec size.")
```

Still the *diener1* scale has the biggest effect size with 0.0363684 eta squared.

3. Run a two-way ANOVA with educational level and gender on the scale type variables. Is there a scale where the interaction significant? How big is the eta-squared here?

Kruskal-Wallis cannot be used to calculate interactions. One option would be using ordinal logistic regression. While every ANOVA is a regression behind the scenes, I will try to stick to an ANOVA. I will try out an Aligned Rank Transformation ANOVA.

```
# For running the analysis we have to first transform the data
ordinal_interaction_res <-
  tibble::tibble(
    variable = ordinal_vars,
    art_trans = map(variable,
                    ~ art(ordinal_merged_data[[.x]] ~ neme*isk, data = ordinal_merged_data)))
```

First, we transformed the data. We have to see 0 for the variables of interest to be able to apply ART ANOVA.

The ART tool automatically runs the correct ANOVA on the transformed data which is nice.

To calculate the partial eta squared effect size measure we will use the sums of squares.

```
# Running the analysis
ordinal_interaction_res <-
  ordinal_interaction_res %>%
  mutate(
    art_re = map(art_trans, ~ anova(.x)),
    f_value_int = map_dbl(art_re,
                          ~filter(.x, Term == "neme:isk") %>% pull(`F value`)),
    p_value_int = map_dbl(art_re,
                          ~filter(.x, Term == "neme:isk") %>% pull(`Pr(>F)`)),
    df_int = map_dbl(art_re,
                          ~filter(.x, Term == "neme:isk") %>% pull(`Df`)),
    df_res_int = map_dbl(art_re,
                          ~filter(.x, Term == "neme:isk") %>% pull(`Df.res`)),
    sum_square_int = map_dbl(art_re,
                          ~filter(.x, Term == "neme:isk") %>% pull(`Sum Sq`)),
    sum_square_res_int = map_dbl(art_re,
                          ~filter(.x, Term == "neme:isk") %>% pull(`Sum Sq.res`)),
    partial_eta_square_int = sum_square_int / (sum_square_int + sum_square_res_int)
    )

ordinal_interaction_res %>%
  select(variable, f_value_int, p_value_int, df_int, df_res_int, partial_eta_square_int) %>%
  arrange(desc(partial_eta_square_int)) %>%
  kable(
    format = "latex",
    col.names = c("Variable", "F Value", "P Value", "Df", "Df Residuals", "Partial Eta Squared"),
    align = c("l", "c", "c", "c", "c", "c"),
    caption = "Results of the Two-way ART ANOVA with Comparing the Interaction Between Three Levels of I
  ) %>%
  row_spec(row = 0, align = "c") %>%
  kable_styling(full_width = TRUE) %>%
  add_footnote("Results are arranged in descending order by the size of the effec size.")
```

4. Is there a difference between the means of the subscales of the Globalis Jollet scale?

To answer this question first I select the needed variables and transform them to long format.

```
global_comparison_data <-
  processed %>%
  select(index, g_jpszi, g_jszoc, g_jspir, g_jerzpsz) %>%
  gather(key = "subscale", value = "value", -index) %>%
  mutate(index = as.factor(index))
```

We will use the Kruskal-Wallis test for testing the ordinal scales.

```
# Run the analysis
kruskal.test(global_comparison_data$value ~ global_comparison_data$subscale)
```

```
##
```

Table 4: Results of the Two-way ART ANOVA with Comparing the Interaction Between Three Levels of Education and Gender on Each Investigated Ordinal Scale

Variable	F Value	P Value	Df	Df Residuals	Partial Eta Squared
ideges	2.9837992	0.0515456	2	477	0.0123561
diener7	2.9647944	0.0525223	2	477	0.0122784
diener8	2.8106506	0.0611627	2	477	0.0116474
diener1	2.7357875	0.0658606	2	477	0.0113407
diener6	2.6394185	0.0724455	2	477	0.0109456
arcocska	2.6220978	0.0736973	2	477	0.0108746
nyugtala	2.2811807	0.1032770	2	477	0.0094741
alt_lelki	2.2268336	0.1089897	2	477	0.0092505
aggodalo	1.9256947	0.1469060	2	477	0.0080095
diener5	1.8944289	0.1515341	2	477	0.0078805
alt_eg_all	1.4863343	0.2272460	2	477	0.0061934
diener2	1.2955317	0.2747137	2	477	0.0054027
feszult	1.1913768	0.3047050	2	477	0.0049705
testi_fi	0.7446720	0.4754413	2	477	0.0031126
fizero	0.5465181	0.5793243	2	477	0.0022862
diener4	0.5094534	0.6011504	2	477	0.0021315
diener3	0.4787743	0.6198397	2	477	0.0020034

[a] Results are arranged in descending order by the size of the effec size.

```
##  Kruskal-Wallis rank sum test
##
## data:  global_comparison_data$value by global_comparison_data$subscale
## Kruskal-Wallis chi-squared = 26.432, df = 3, p-value = 7.745e-06
```

The comparison between the subscales of the Globalis Jollet scale is significant indicating a difference between the rank means of the subscales.

Therefore, we can run post hoc tests. We are going to perform a Dunn test with Holm method adjusting for family error rate.

```
dunnTest(global_comparison_data$value ~ global_comparison_data$subscale, data = global_comparison_data)
```

```
## Warning: global_comparison_data$subscale was coerced to a factor.
```

```
## Dunn (1964) Kruskal-Wallis multiple comparison
```

```
##   p-values adjusted with the Holm method.
```

```
##            Comparison           Z      P.unadj        P.adj
## 1 g_jerzpsz - g_jpszi -1.04711750 2.950454e-01 8.851362e-01
## 2 g_jerzpsz - g_jspir -1.01293184 3.110927e-01 6.221854e-01
## 3   g_jpszi - g_jspir  0.03418566 9.727291e-01 9.727291e-01
## 4 g_jerzpsz - g_jszoc  3.39710071 6.810390e-04 2.724156e-03
## 5   g_jpszi - g_jszoc  4.44421821 8.821204e-06 5.292722e-05
## 6   g_jspir - g_jszoc  4.41003255 1.033551e-05 5.167754e-05
```

The p values are indicating a significant difference between all of the subscales with the *g_jszoc* subscale with $alpha = 0.05$.

5. On which scales can we find the biggest difference between man and woman in stohastic dominance?

To test for the stochastic dominance we first have to transform the data so that all observations from the male participants are in one vector for the given scale, and all observations for the female participants in another vector for the given scale. We will do this nested, so we can run the analysis for each investigated variable in bulk.

We are comparing only the ordinal variables described at the beginning of this assignment. We also drop all the participants where there is at least one missing observation.

```
transform_scales <- function(var) {
  select(ordinal_data, neme, .data[[var]]) %>%
    pivot_wider(names_from = neme, values_from = .data[[var]])
}

sto_dom_data <-
  tibble::tibble(
    variable = ordinal_vars,
    data = map(variable, transform_scales)
  )
```

```
## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
```

```
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates

## Warning: Values are not uniquely identified; output will contain list-cols.
## * Use `values_fn = list` to suppress this warning.
## * Use `values_fn = length` to identify where the duplicates arise
## * Use `values_fn = {summary_fun}` to summarise duplicates
```

Now, we can run the test for stohastic dominance in two samples. We are using *alpha* = 0.05, and 1e+05 iterations for calculating the p value. The alpha dependent tuning parameter will be 2.2.

```
sto_dom_res <-
  tibble::tibble(
    res = map(sto_dom_data$data,
              ~ ddst.forstochdom.test(.x$ferfi[[1]], .x$no[[1]], t = 2.2, compute.p = TRUE)),
    plot = map(res, plot)
  )
```

The results.

```
sto_dom_res$res
```

```
## [[1]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[2]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 137.12, T = 8
##
##
## [[3]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 92.755, T = 8
##
##
## [[4]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[5]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0.053709, T = 1
##
```

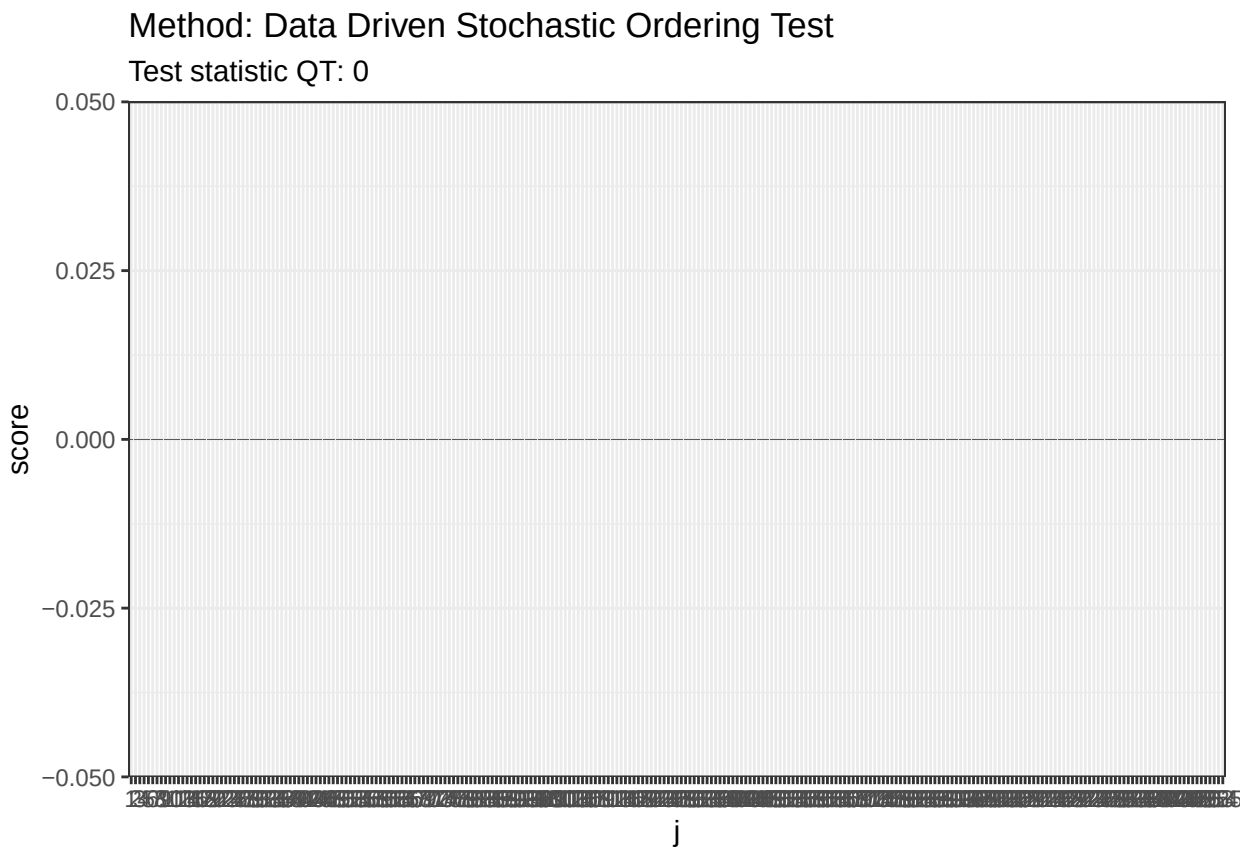
```
##
## [[6]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[7]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[8]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[9]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[10]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[11]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[12]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
```

```
## QT = 0, T = 1
##
##
## [[13]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[14]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[15]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 104.14, T = 8
##
##
## [[16]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
##
##
## [[17]]
##
##  Data Driven Stochastic Ordering Test
##
## data:  [[ [[ .x$ferfi .x$no 1 1, t: 2.2, K.N: 7
## QT = 0, T = 1
```

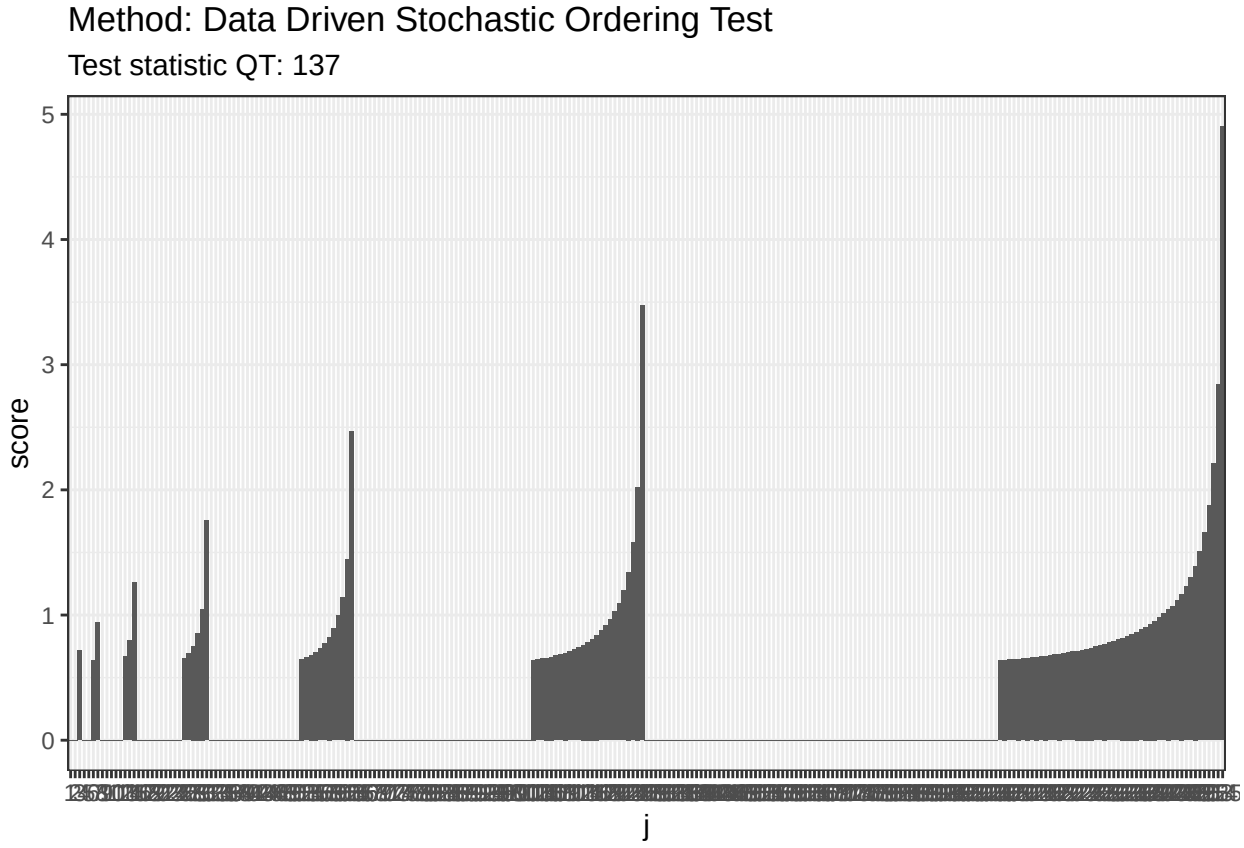
Plotting the results.

```
sto_dom_res$plot
```

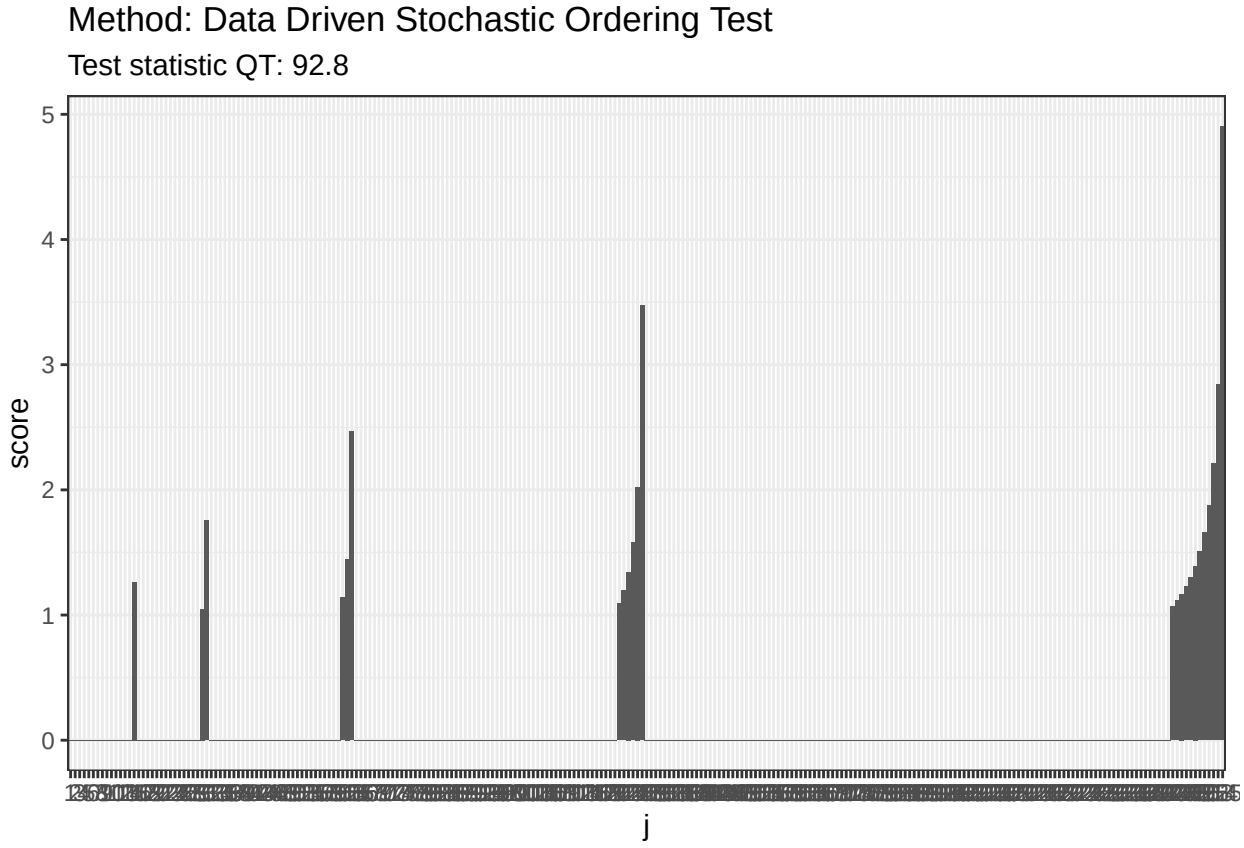
```
## [[1]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
0.050
0.025
0.000
−0.025
−0.050
score
j

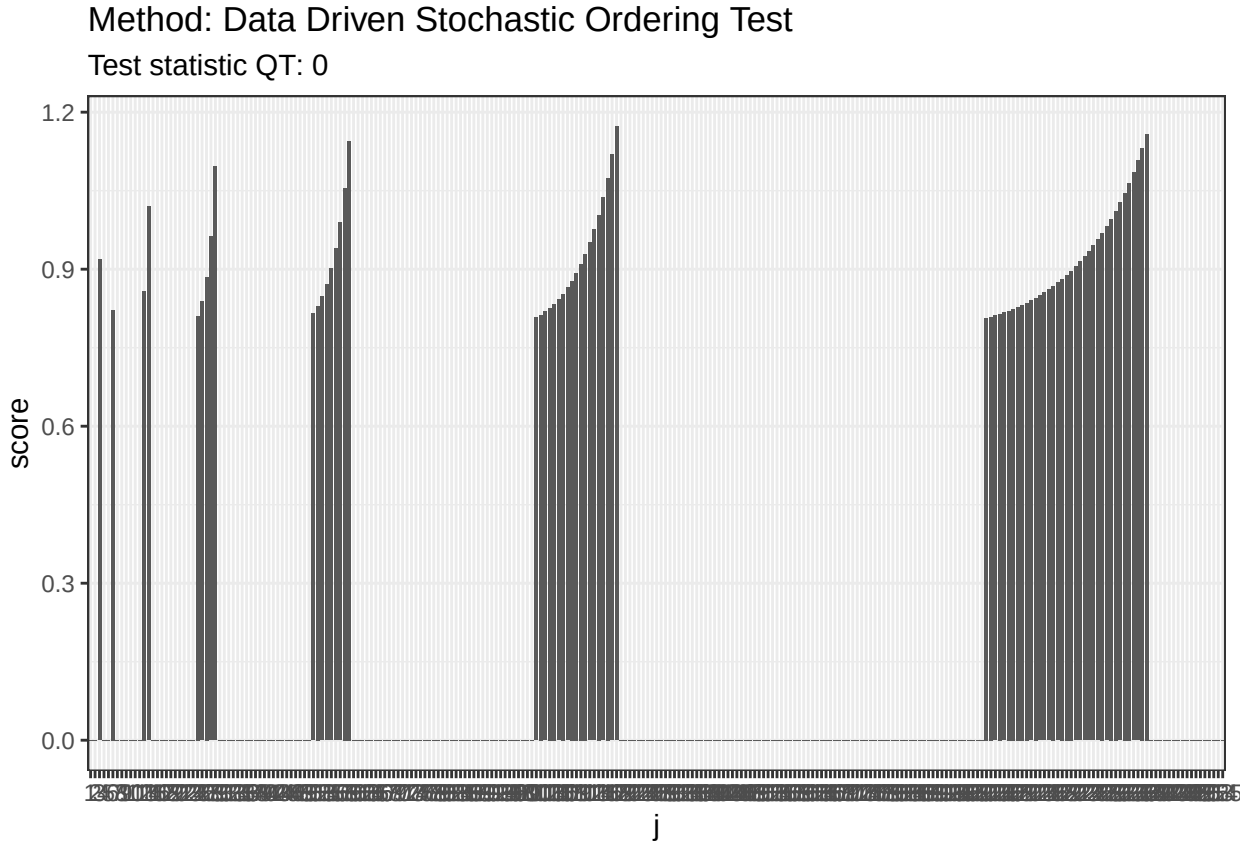
```
##
## [[2]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 137
score
j
0
1
2
3
4
5

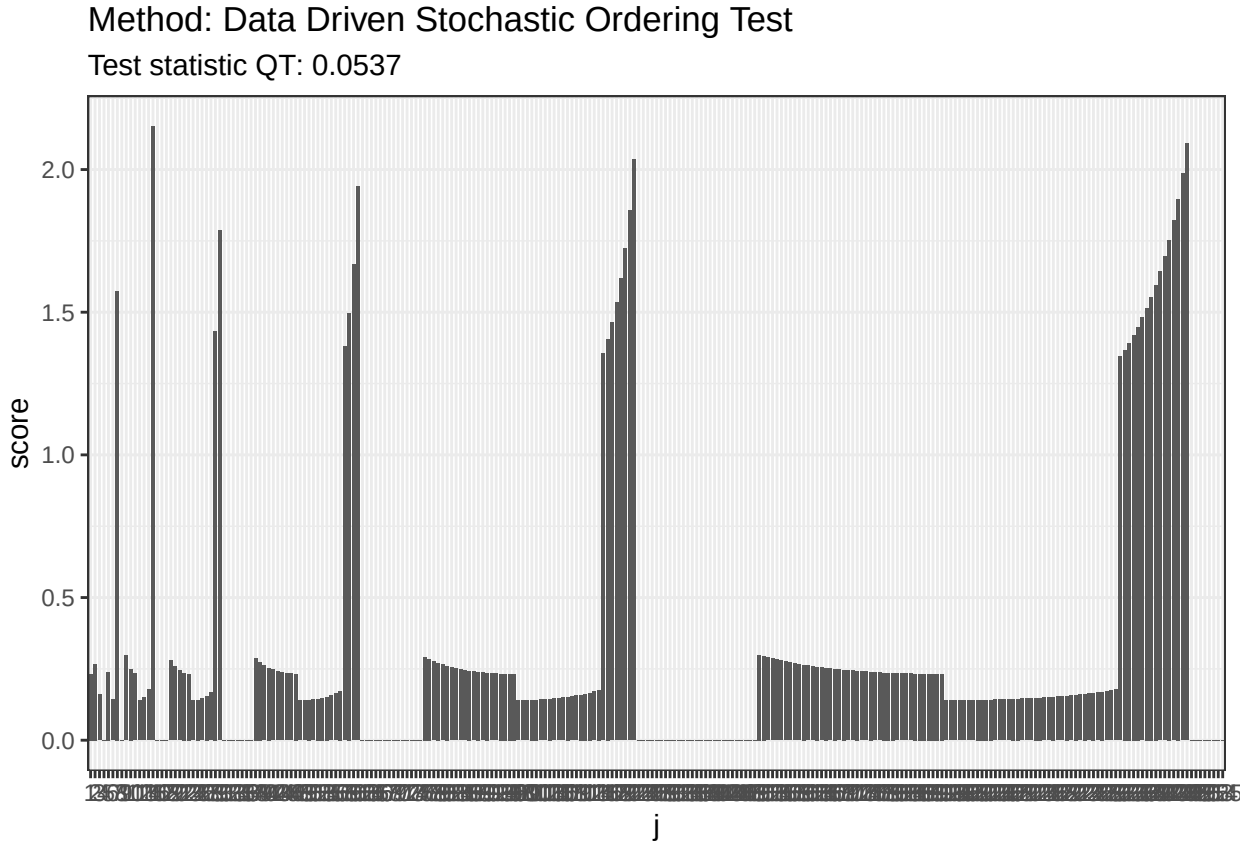
```
##
## [[3]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 92.8
score
j
0
1
2
3
4
5

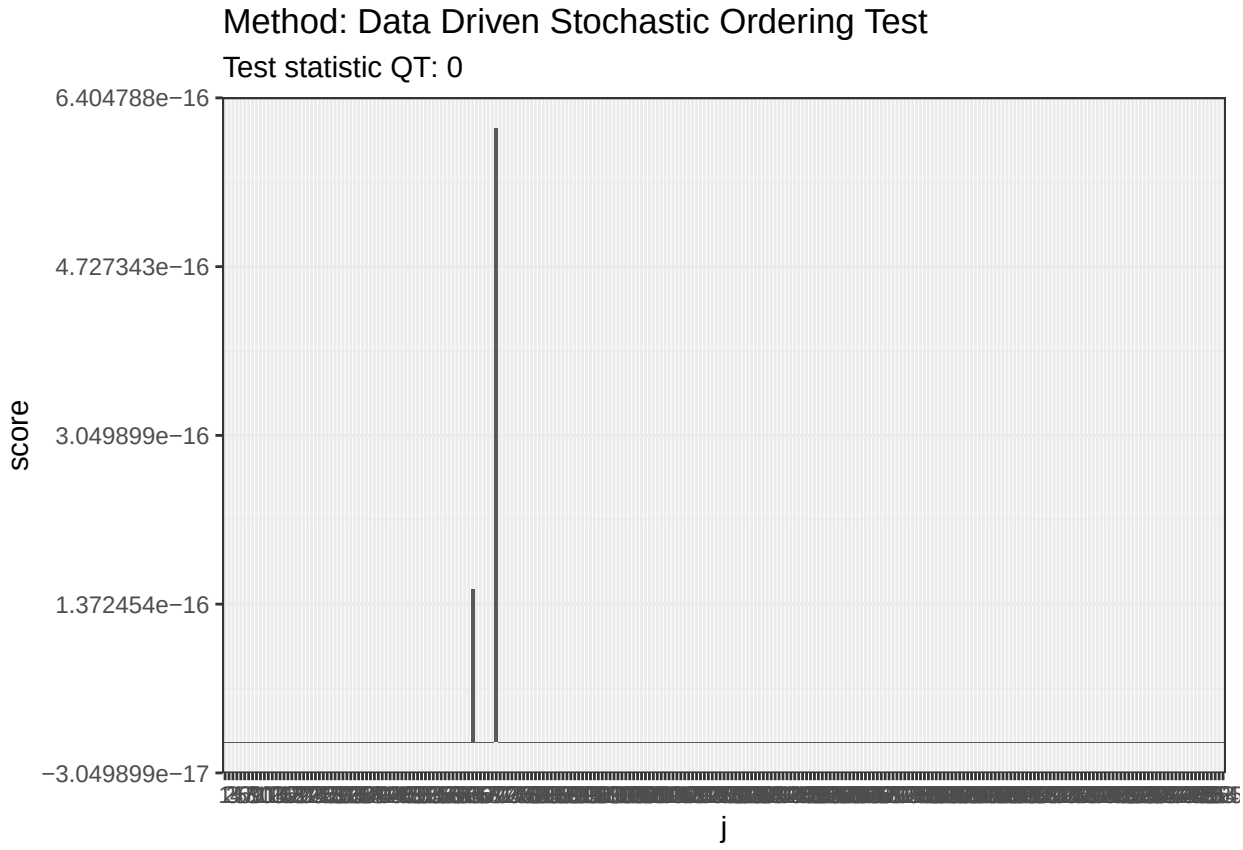
```
##
## [[4]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
score
1.2
0.9
0.6
0.3
0.0
j

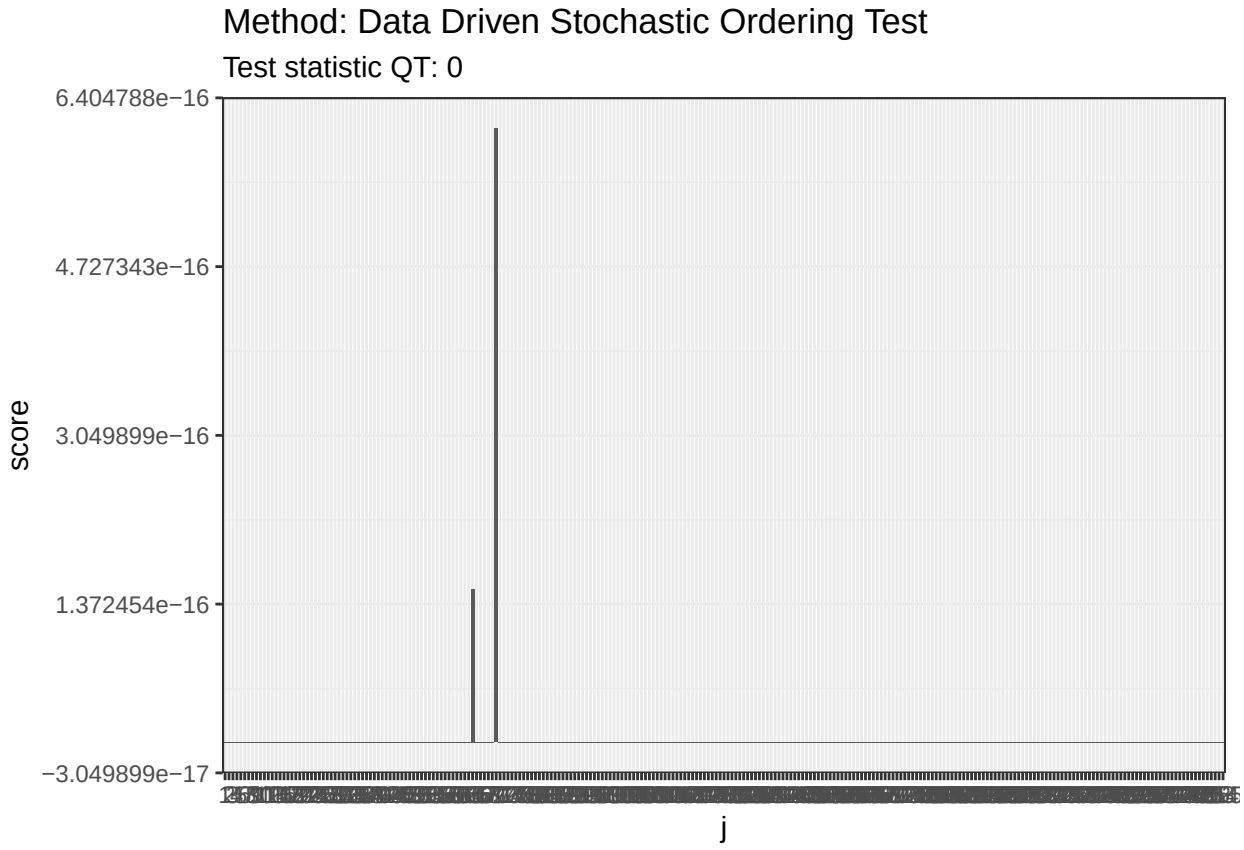
```
##
## [[5]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0.0537
score
2.0
1.5
1.0
0.5
0.0
j

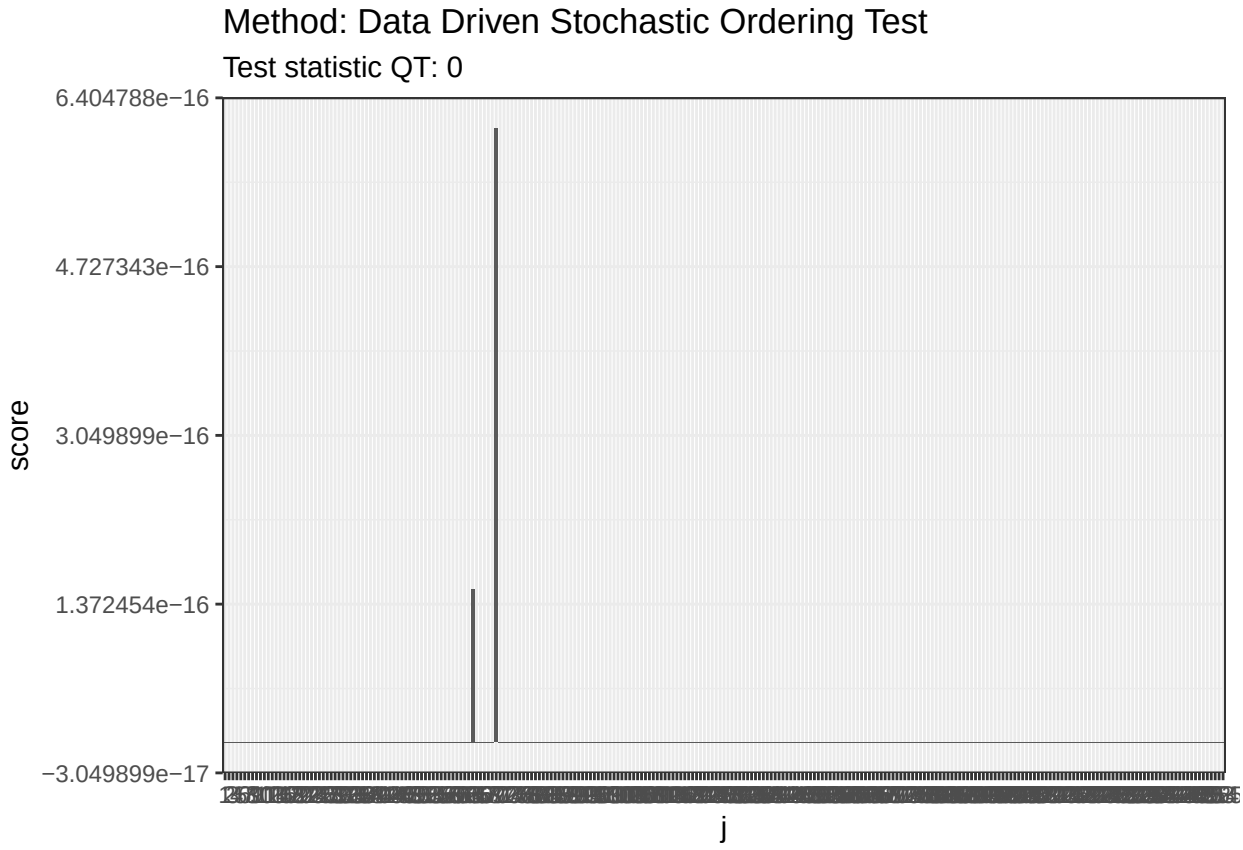
```
##
## [[6]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
6.404788e−16
4.727343e−16
3.049899e−16
1.372454e−16
−3.049899e−17
score
j

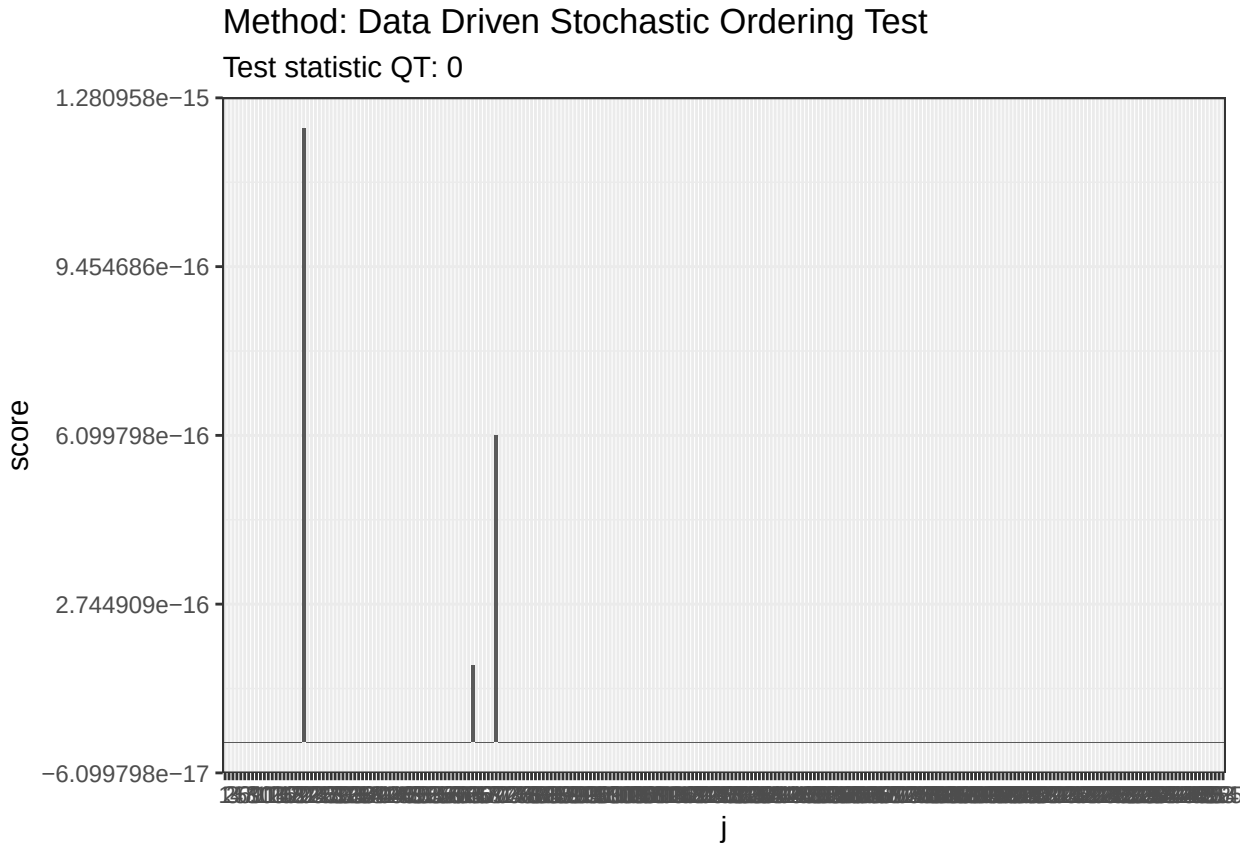
```
##
## [[7]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
6.404788e−16
4.727343e−16
3.049899e−16
1.372454e−16
−3.049899e−17
score
j

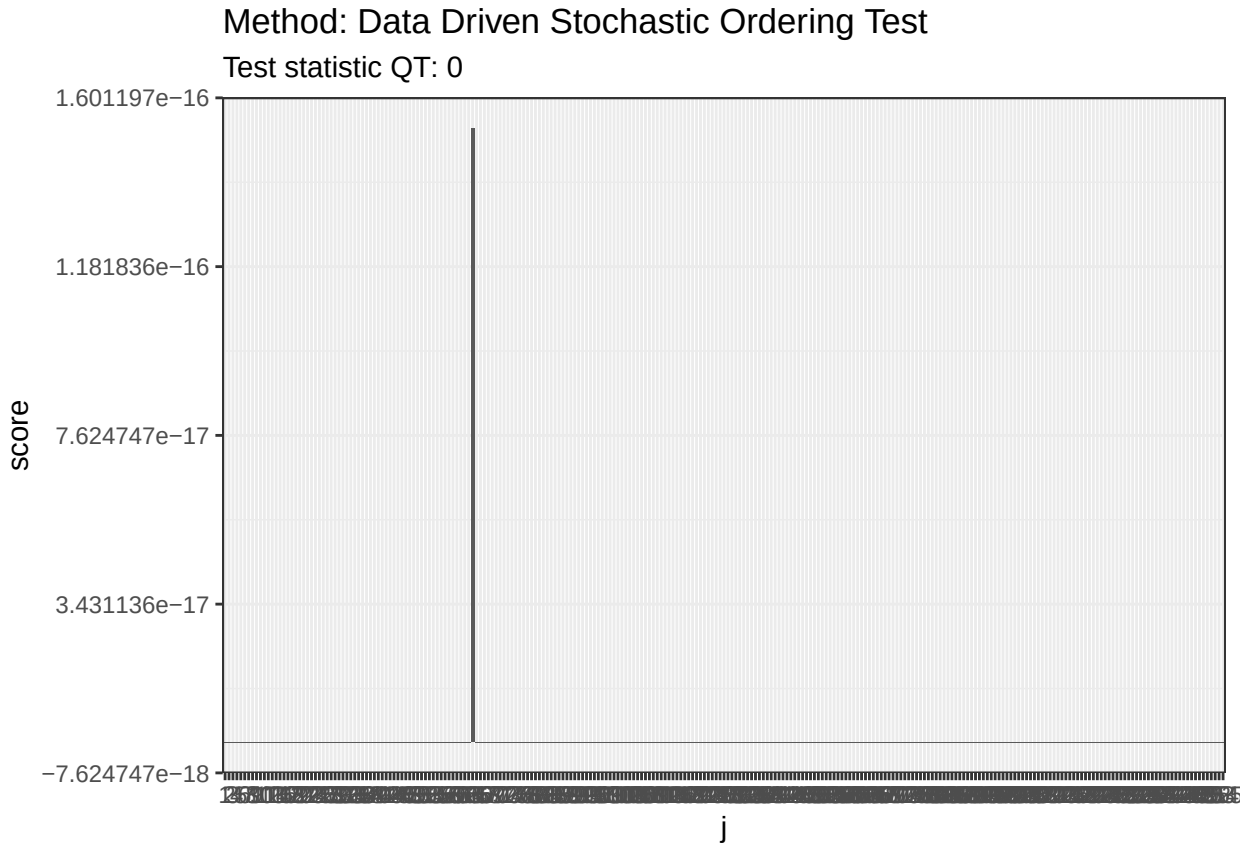
```
##
## [[8]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
6.404788e−16
4.727343e−16
3.049899e−16
1.372454e−16
−3.049899e−17
score
j

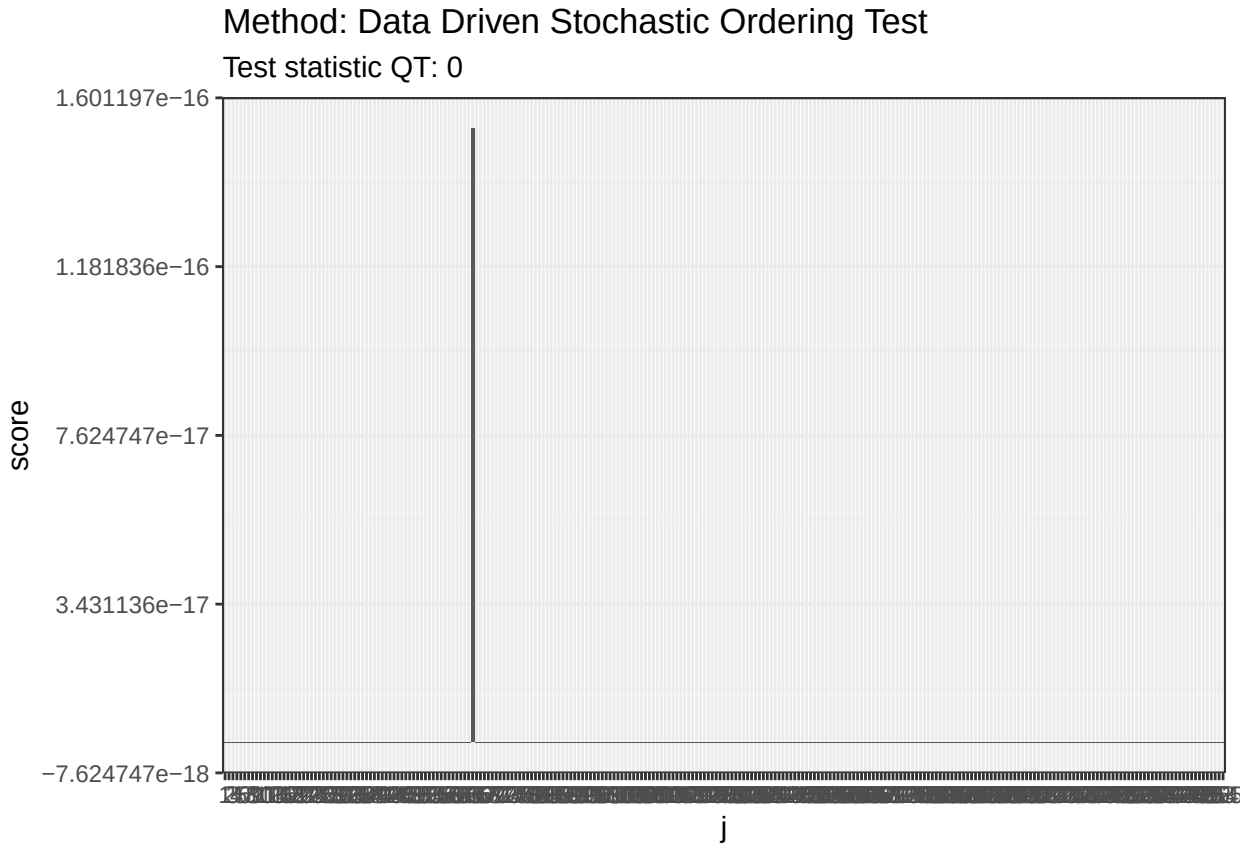
```
##
## [[9]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
1.280958e−15
9.454686e−16
6.099798e−16
2.744909e−16
−6.099798e−17
score
j

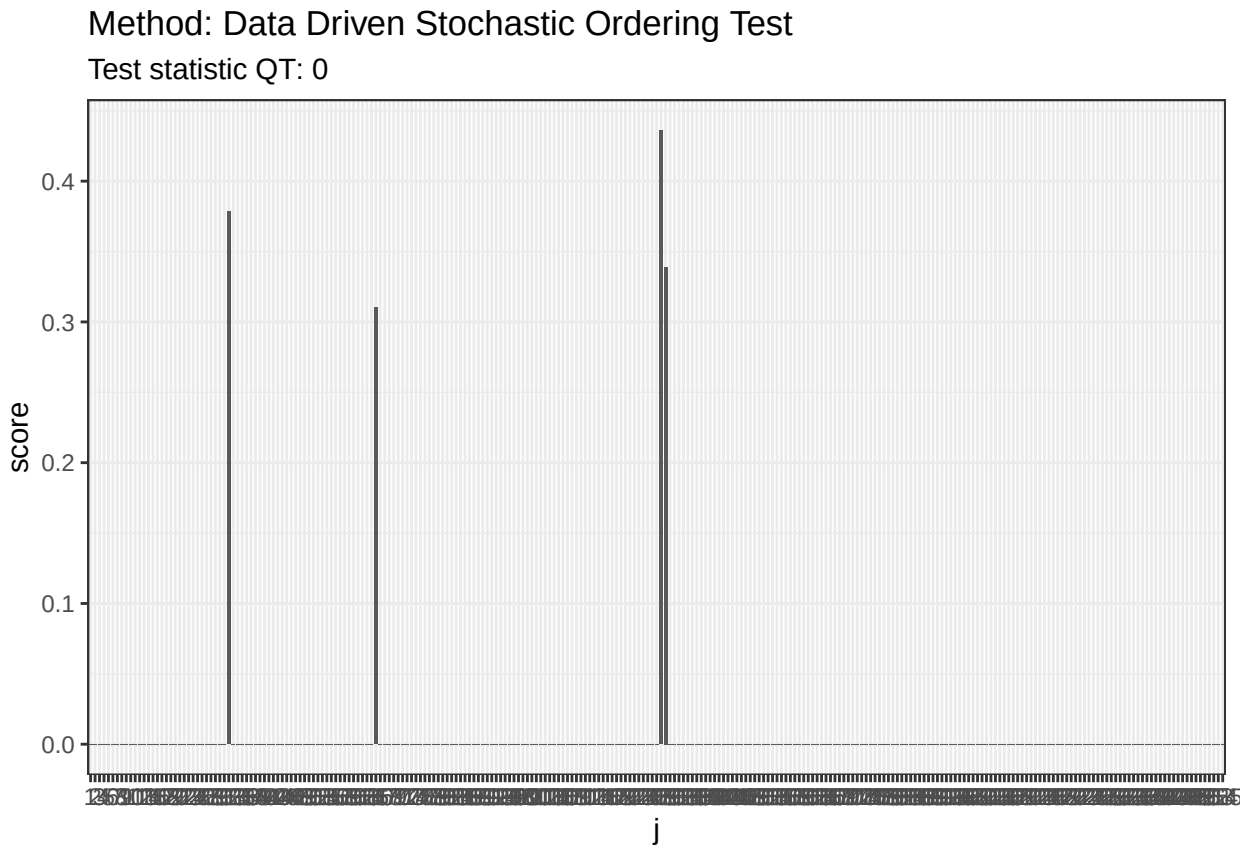
```
##
## [[10]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
1.601197e−16
1.181836e−16
7.624747e−17
3.431136e−17
−7.624747e−18
score
j

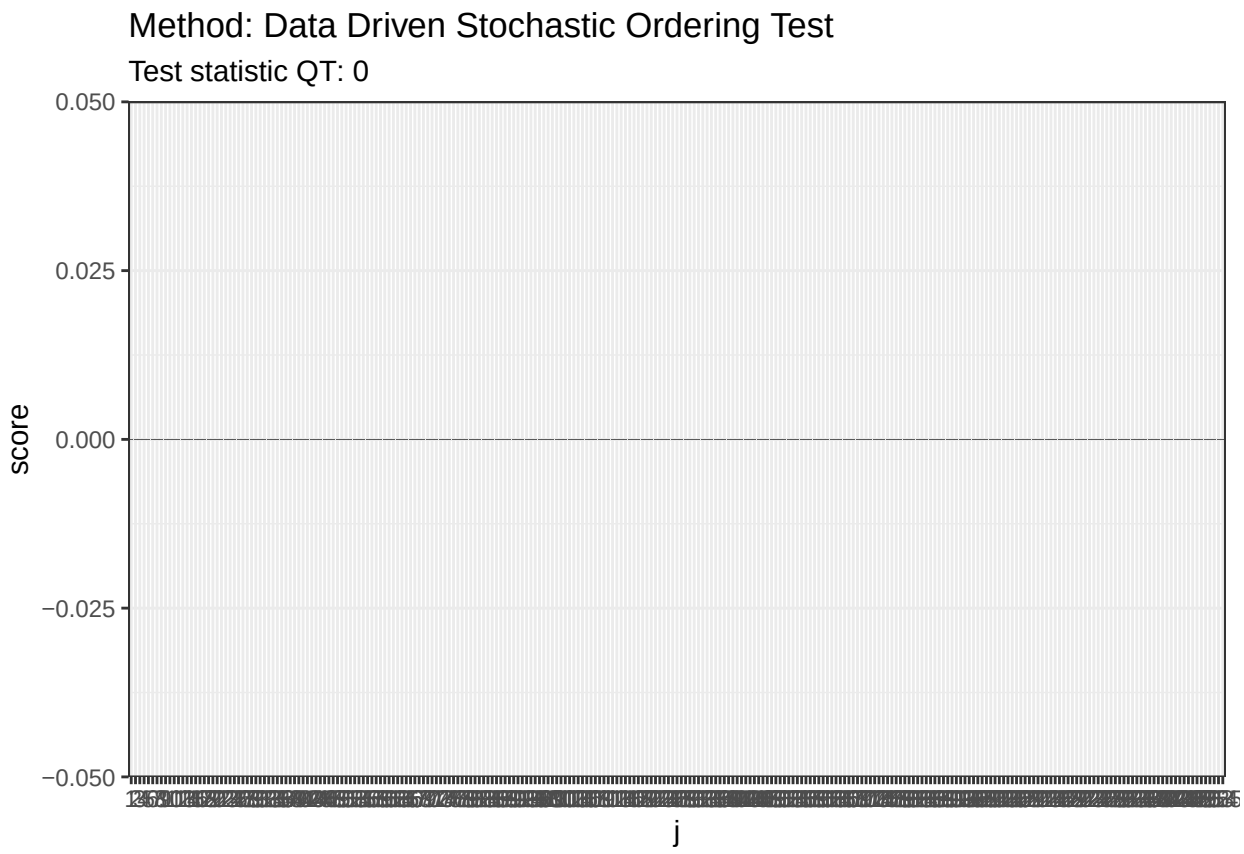
```
##
## [[11]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
1.601197e−16
1.181836e−16
7.624747e−17
3.431136e−17
−7.624747e−18
score
j

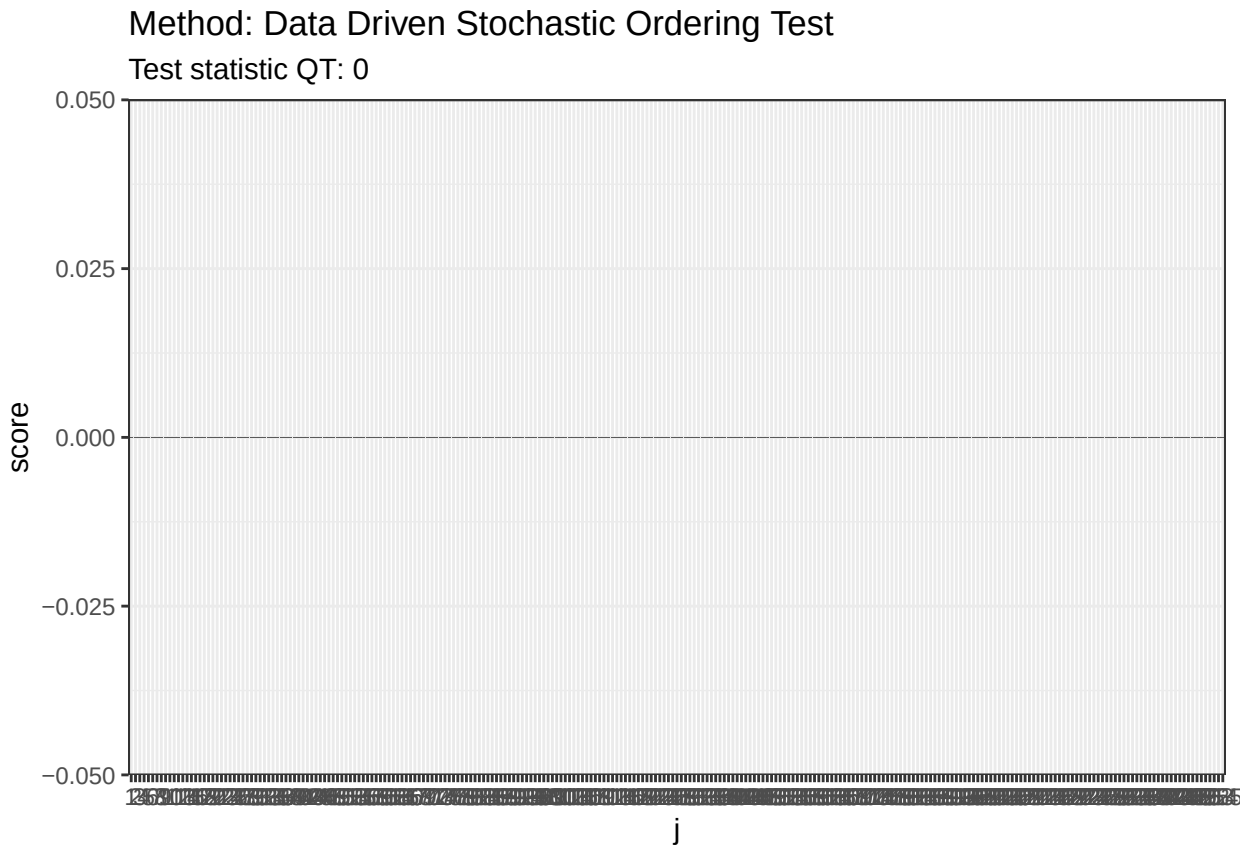
```
##
## [[12]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
score
0.4
0.3
0.2
0.1
0.0
j

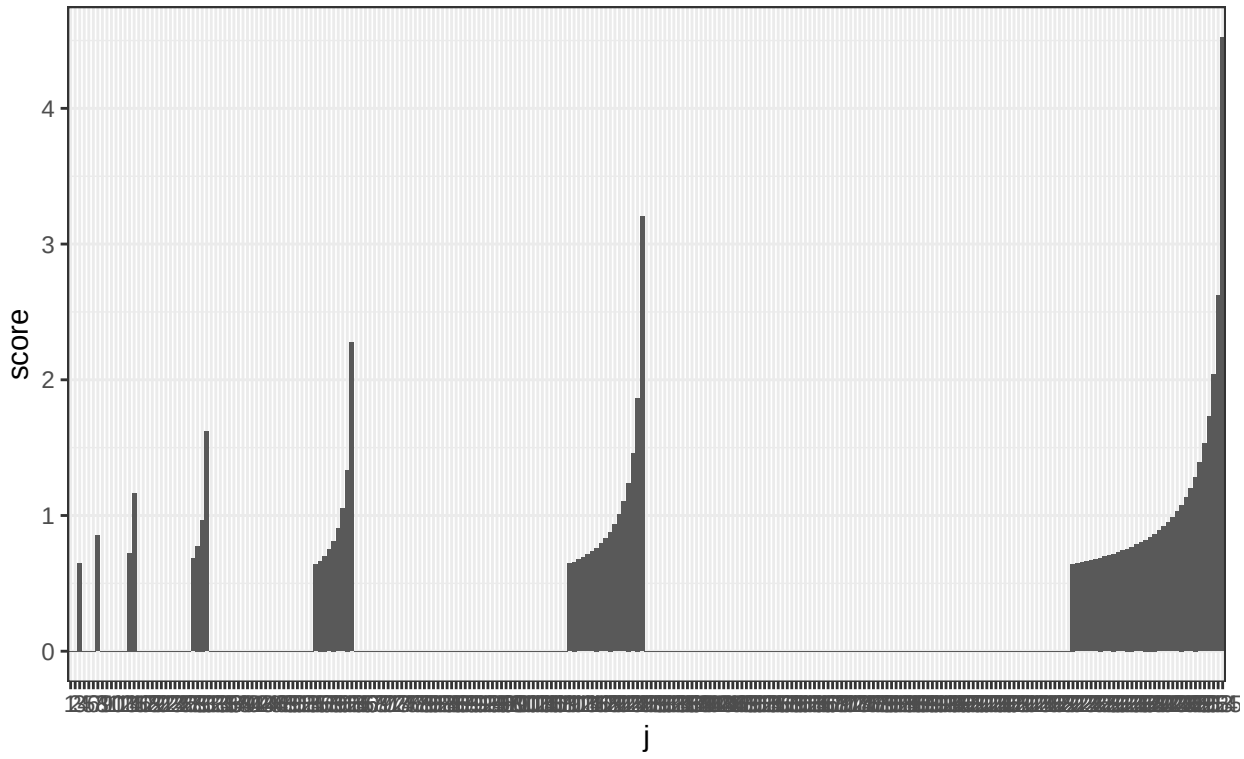
```
##
## [[13]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
0.050
0.025
0.000
−0.025
−0.050
score
j

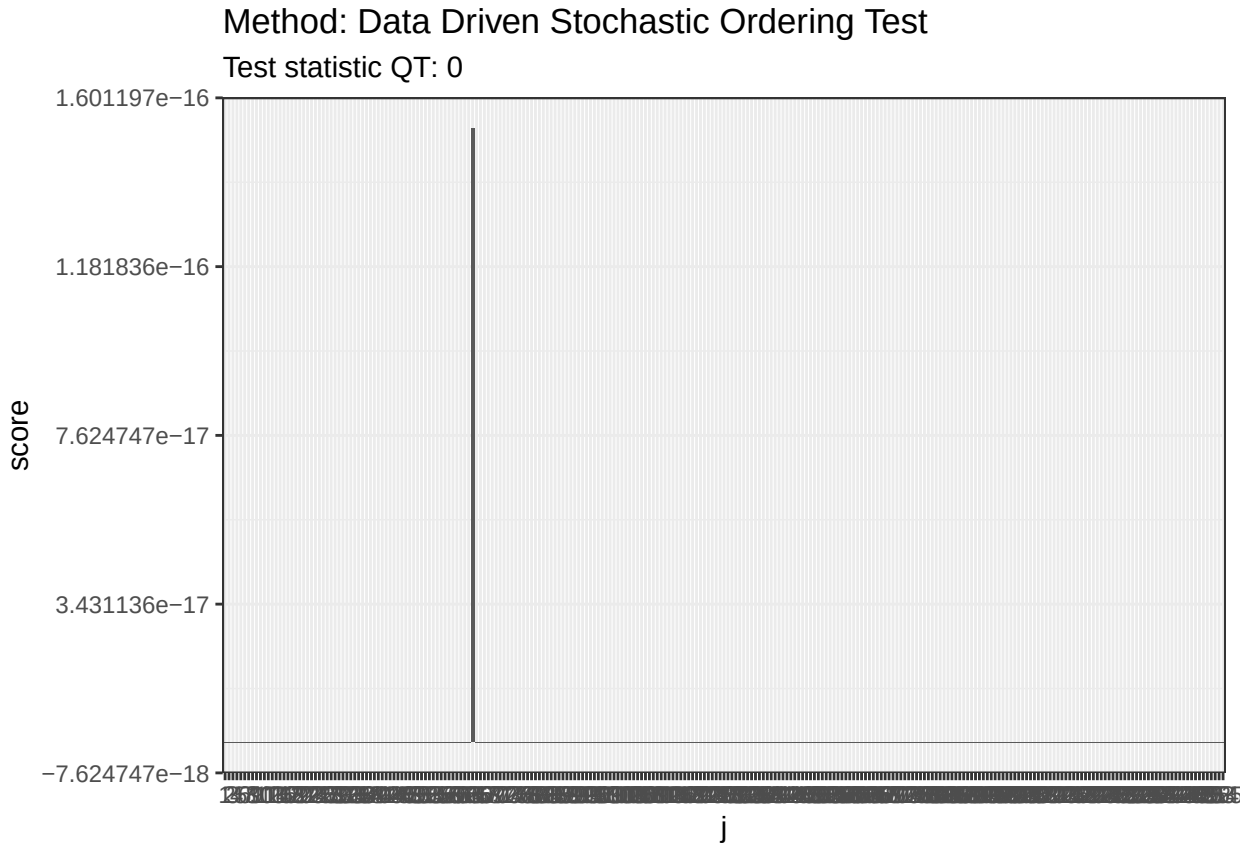
```
##
## [[14]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
0.050
0.025
0.000
−0.025
−0.050
score
j

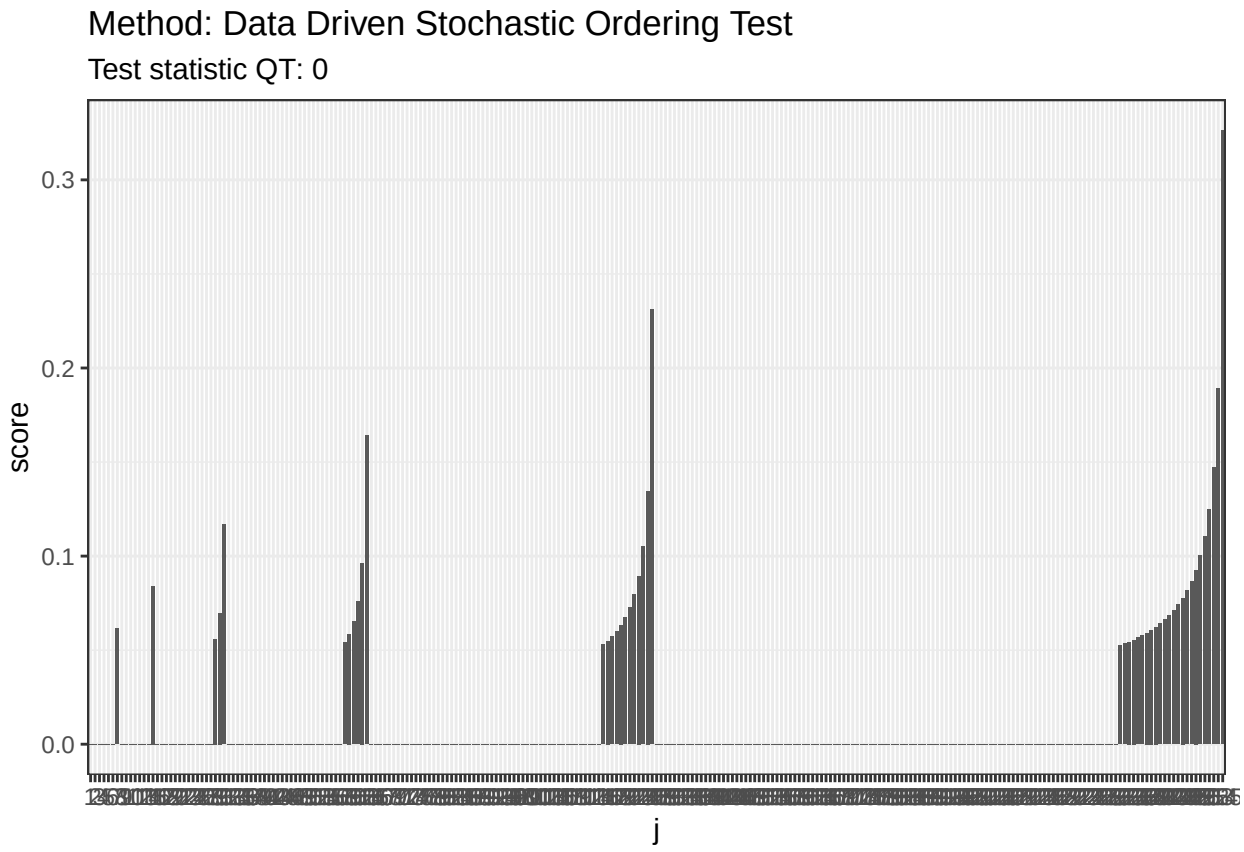
```
##
## [[15]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 104
score
0
1
2
3
4
j

```
##
## [[16]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
1.601197e−16
1.181836e−16
7.624747e−17
3.431136e−17
−7.624747e−18
score
j

```
##
## [[17]]
```


Method: Data Driven Stochastic Ordering Test
Test statistic QT: 0
score
0.0
0.1
0.2
0.3
j