Assessing the impact of violating the assumption of normality

Within a Welches' independent t-test

Ian Hussey

06 Mai, 2024

Contents

Overview of tutorial

We are frequently told that statistical tests have assumptions, that it is important to check these assumptions, and that there are consequences for violating them. What consequences, specifically? How badly do they need to be violated to substantively affect our inferences? In a previous lesson we examined how robust Welches' t-test vs. Students' $t$test is to differences in variances between the conditions. In this lesson, we will examine how violating the assumption of normality impacts the results of the Welches' t-test.

Skew-normal distributions

In order to violate normality, we will need to use a different non-normal distribution. For this example we'll use the skew-normal distribution. Where the normal distribution of defined by two parameters, mean and standard deviation, other distributions are controlled by other parameters with different naming conventions, and often more than two parameters.

The skew-normal distribution is defined by:

- 'location', akin to mean, controlled via parameter 'location' in `sn()`. In fact, mean is referred to as 'location' in many distributions.

- ‘scale’, akin to SD, controlled via parameter ‘omega’ in `sn()`. Likewise, ‘scale’ is a common way of referring to measures of dispersion like SD.
- ‘slant’/‘skew’, controlled via parameter ‘alpha’ in `sn()`.

Note that when alpha = 0, skew-normal data is the same as normal data:

```
# dependencies
library(tidyr)
library(dplyr)
```

```
##
## Attache Paket: 'dplyr'

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     filter, lag

## Die folgenden Objekte sind maskiert von 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(purrr)
library(ggplot2)
library(sn)
```

```
## Warning: Paket 'sn' wurde unter R Version 4.3.3 erstellt

## Lade nötiges Paket: stats4

##
## Attache Paket: 'sn'

## Das folgende Objekt ist maskiert 'package:stats':
##
##     sd
```

```
library(knitr)
```

```
## Warning: Paket 'knitr' wurde unter R Version 4.3.3 erstellt
```

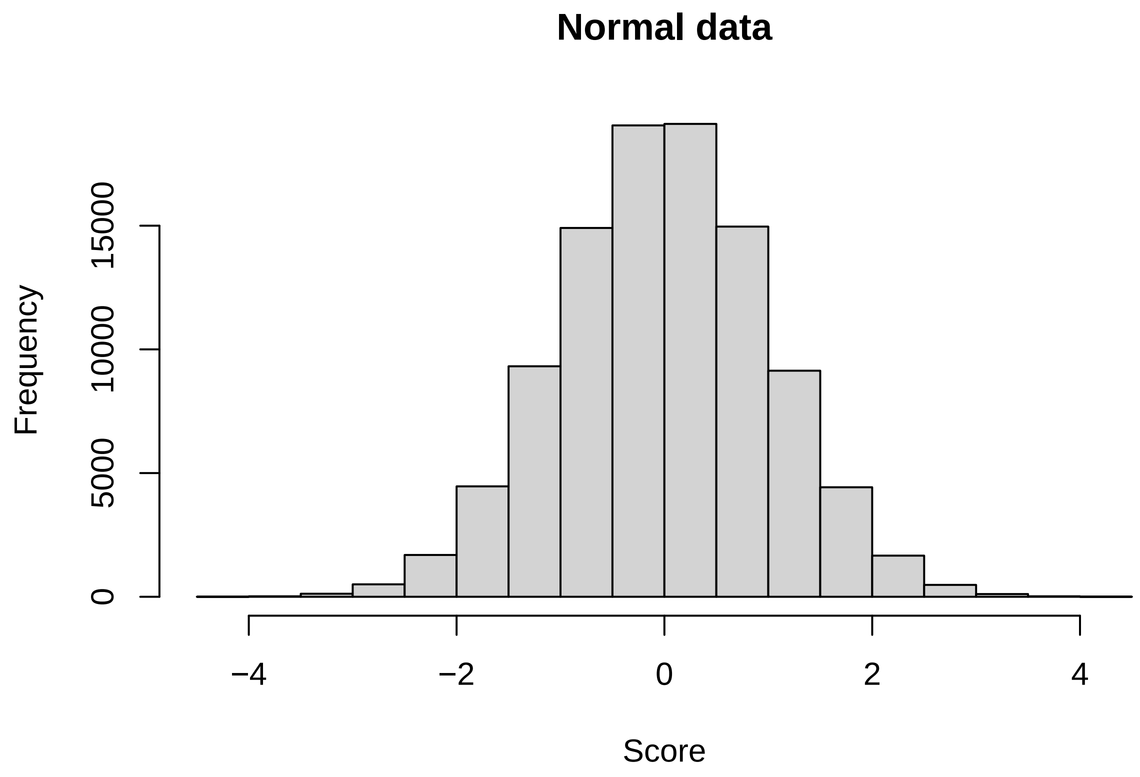
```
library(kableExtra)
```

```
##
## Attache Paket: 'kableExtra'

## Das folgende Objekt ist maskiert 'package:dplyr':
##
##     group_rows
```

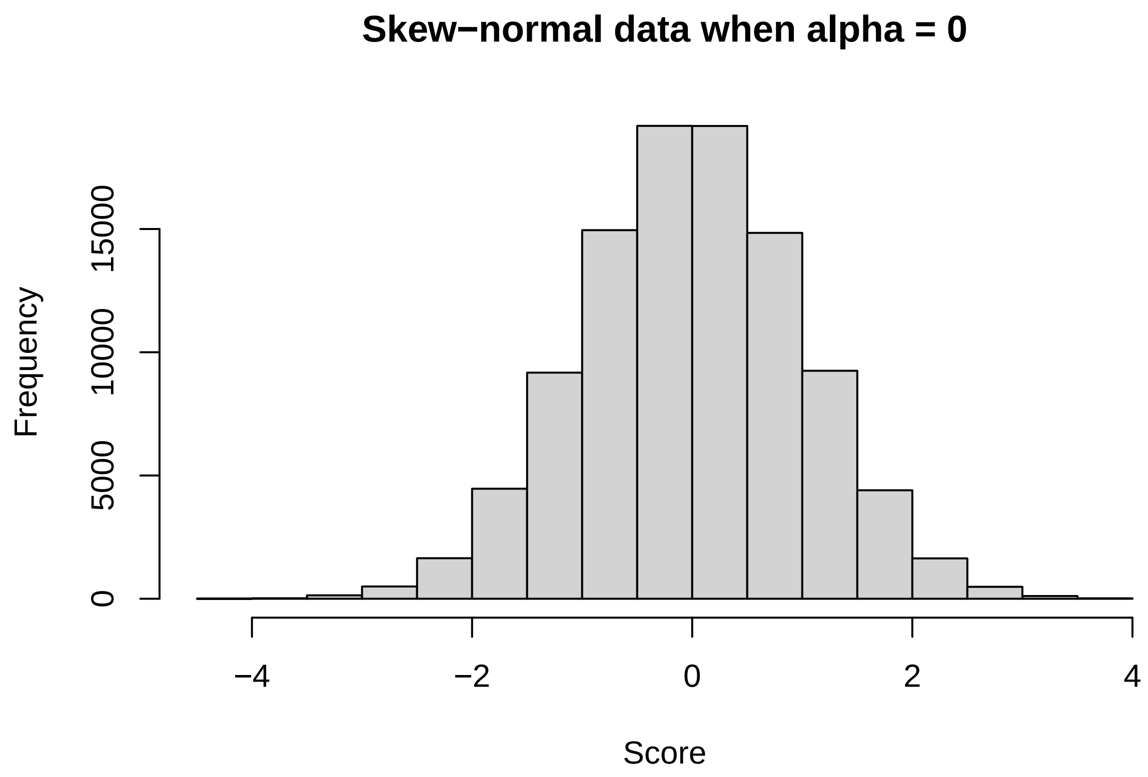
```
library(janitor)
```

```
##
## Attache Paket: 'janitor'

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     chisq.test, fisher.test
```

```
# simple plot of a normal distribution
set.seed(42)

rnorm(n = 100000,
      mean = 0,
      sd = 1) |>
  hist(main = "Normal data", xlab = "Score")
```

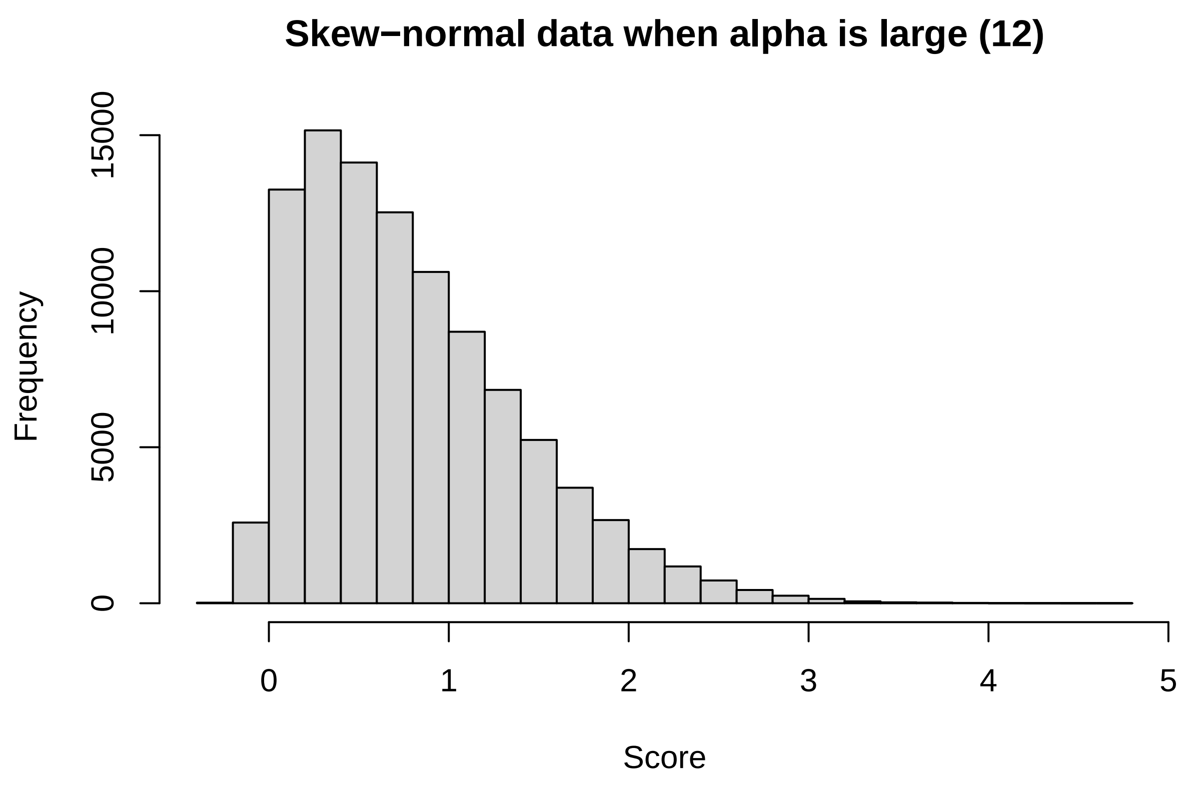


```
# simple plot of a skew-normal distribution
set.seed(42)

rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 0) |>
  hist(main = "Skew-normal data when alpha = 0", xlab = "Score")
```



For more extreme values of alpha data becomes increasingly skewed. e.g., alpha = 12.

```
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 12) |>
  hist(main = "Skew-normal data when alpha is large (12)", xlab = "Score")
```



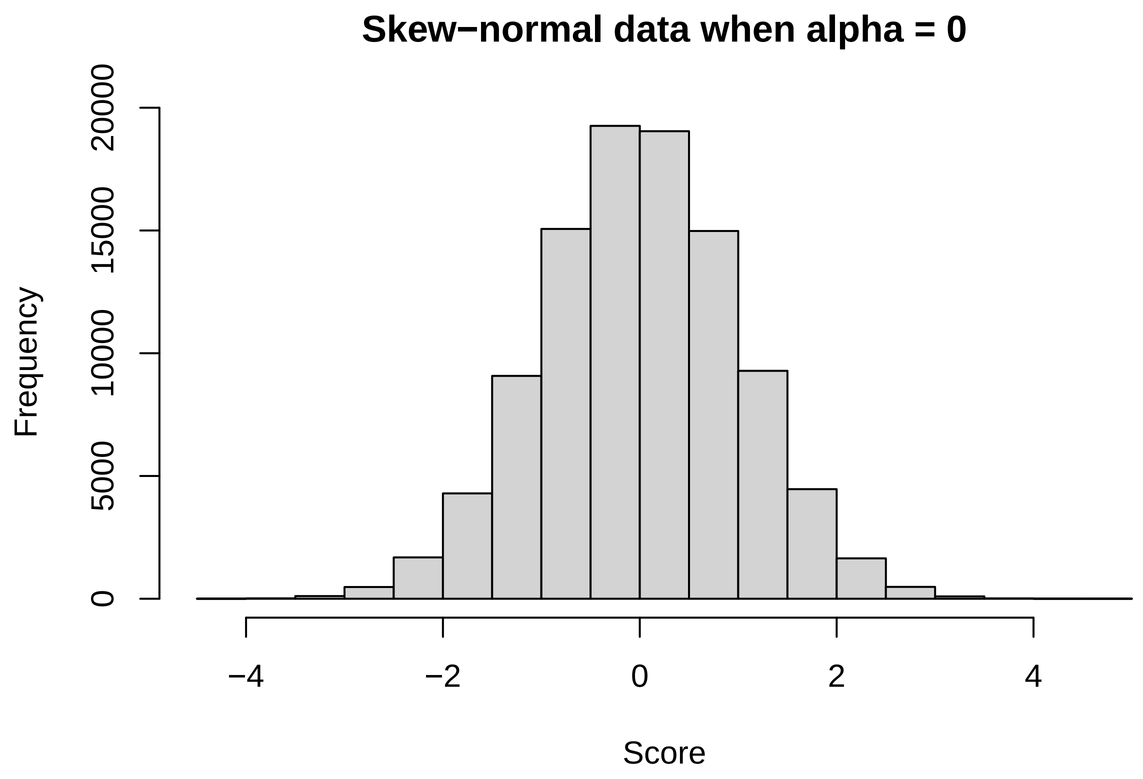
Impact of non-normality on *p* value false-positive rates

Data generation and analysis functions

```
# define data generating function ----
generate_data <- function(n_control,
                          n_intervention,
                          location_control, # location, akin to mean
                          location_intervention,
                          scale_control, # scale, akin to SD
                          scale_intervention,
                          skew_control, # slant/skew. When 0, produces normal/gaussian data
                          skew_intervention) {

  data_control <-
    tibble(condition = "control",
           score = rsn(n = n_control,
                        xi = location_control, # location, akin to mean
                        omega = scale_control, # scale, akin to SD
                        alpha = skew_control)) # slant/skew. When 0, produces normal/gaussian data

  data_intervention <-
    tibble(condition = "intervention",
           score = rsn(n = n_intervention,
                        xi = location_intervention, # location, akin to mean
                        omega = scale_intervention, # scale, akin to SD
                        alpha = skew_intervention)) # slant/skew. When 0, produces normal/gaussian data

  data <- bind_rows(data_control,
                    data_intervention)

  return(data)
}


# define data analysis function ----
analyse_data <- function(data) {
  res_t_test <- t.test(formula = score ~ condition,
                       data = data,
                       var.equal = TRUE,
                       alternative = "two.sided")

  res <- tibble(p = res_t_test$p.value)

  return(res)
}
```

Between condition A: normal data

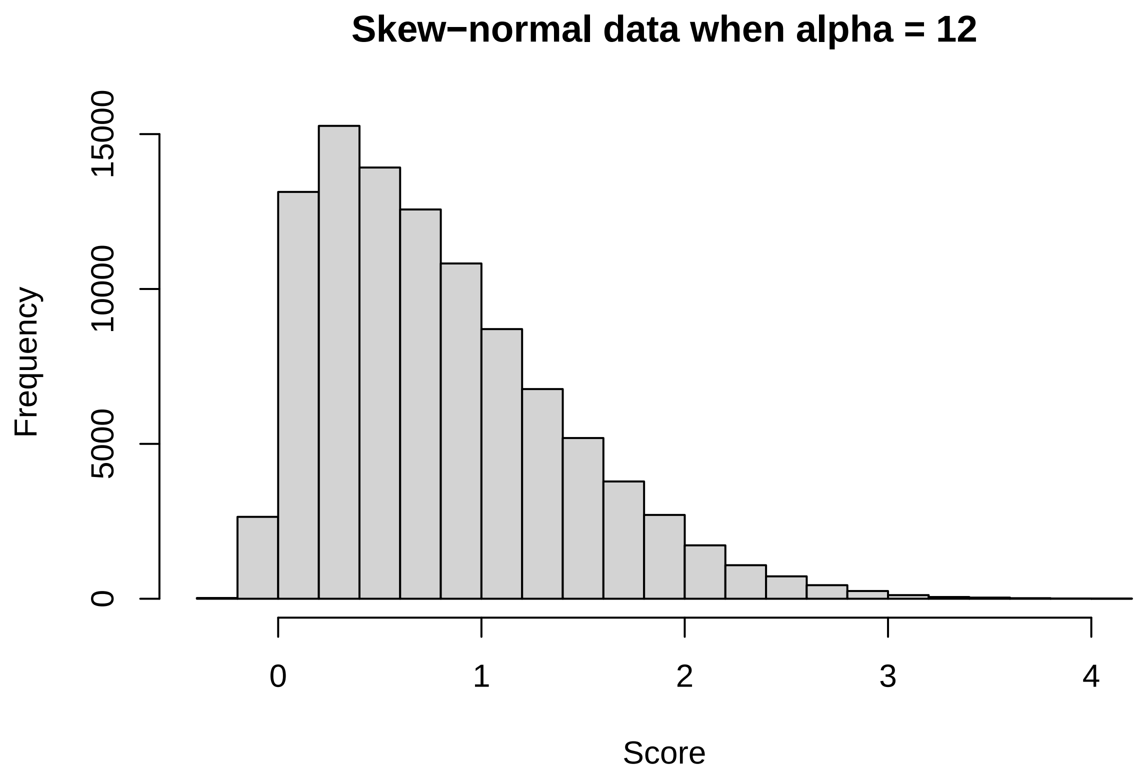
```
# reminder of the distribution used in the simulation
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 0) |>
  hist(main = "Skew-normal data when alpha = 0", xlab = "Score")
```



```
# set the seed ----
# for the pseudo random number generator to make results reproducible
set.seed(42)

# define experiment parameters ----
experiment_parameters_grid_a <- expand_grid(
  n_control = 100,
  n_intervention = 100,
  location_control = 0,
  location_intervention = 0,
  scale_control = 1,
  scale_intervention = 1,
  skew_control = 0,
  skew_intervention = 0,
  iteration = 1:10000
)

# run simulation ----
simulation_a <-
  # using the experiment parameters
  experiment_parameters_grid_a |>

  # generate data using the data generating function and the parameters relevant to data generation
  mutate(generated_data = pmap(list(n_control,
                                    n_intervention,
                                    location_control,
                                    location_intervention,
                                    scale_control,
                                    scale_intervention,
                                    skew_control,
                                    skew_intervention),
                               generate_data)) |>

  # apply the analysis function to the generated data using the parameters relevant to analysis
  mutate(analysis_results = pmap(list(generated_data),
                                 analyse_data))


# summarise simulation results over the iterations ----
simulation_a_summary <- simulation_a |>
  unnest(analysis_results) |>
  summarize(false_positive_rate = janitor::round_half_up(mean(p < .05), digits = 3)) |>
  mutate(population_distribution = "normal")
```

Between condition B: highly skewed data

```
# reminder of the distribution used in the simulation
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 12) |>
  hist(main = "Skew-normal data when alpha = 12", xlab = "Score")
```



```
# set the seed ----
# for the pseudo random number generator to make results reproducible
set.seed(42)

# define experiment parameters ----
experiment_parameters_grid_b <- expand_grid(
  n_control = 100,
  n_intervention = 100,
  location_control = 0,
  location_intervention = 0,
  scale_control = 1,
  scale_intervention = 1,
  skew_control = 12, # data in both conditions is highly skewed
  skew_intervention = 12, # data in both conditions is highly skewed
  iteration = 1:10000
)

# run simulation ----
simulation_b <-
  # using the experiment parameters
  experiment_parameters_grid_b |>

  # generate data using the data generating function and the parameters relevant to data generation
  mutate(generated_data = pmap(list(n_control,
                                    n_intervention,
                                    location_control,
                                      location_intervention,
                                      scale_control,
                                      scale_intervention,
                                      skew_control,
                                      skew_intervention),
                                 generate_data)) |>

  # apply the analysis function to the generated data using the parameters relevant to analysis
  mutate(analysis_results = pmap(list(generated_data),
                                   analyse_data))


# summarise simulation results over the iterations ----
simulation_b_summary <- simulation_b |>
  unnest(analysis_results) |>
  summarize(false_positive_rate = janitor::round_half_up(mean(p < .05), digits = 3)) |>
  mutate(population_distribution = "highly skewed non-normal")
```

Summarize results

```
bind_rows(
  simulation_a_summary,
  simulation_b_summary
) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

false_positive_rate	population_distribution
0.047	normal
0.049	highly skewed non-normal

Impact of non-normality on estimates of mean differences

Modify the data analysis function

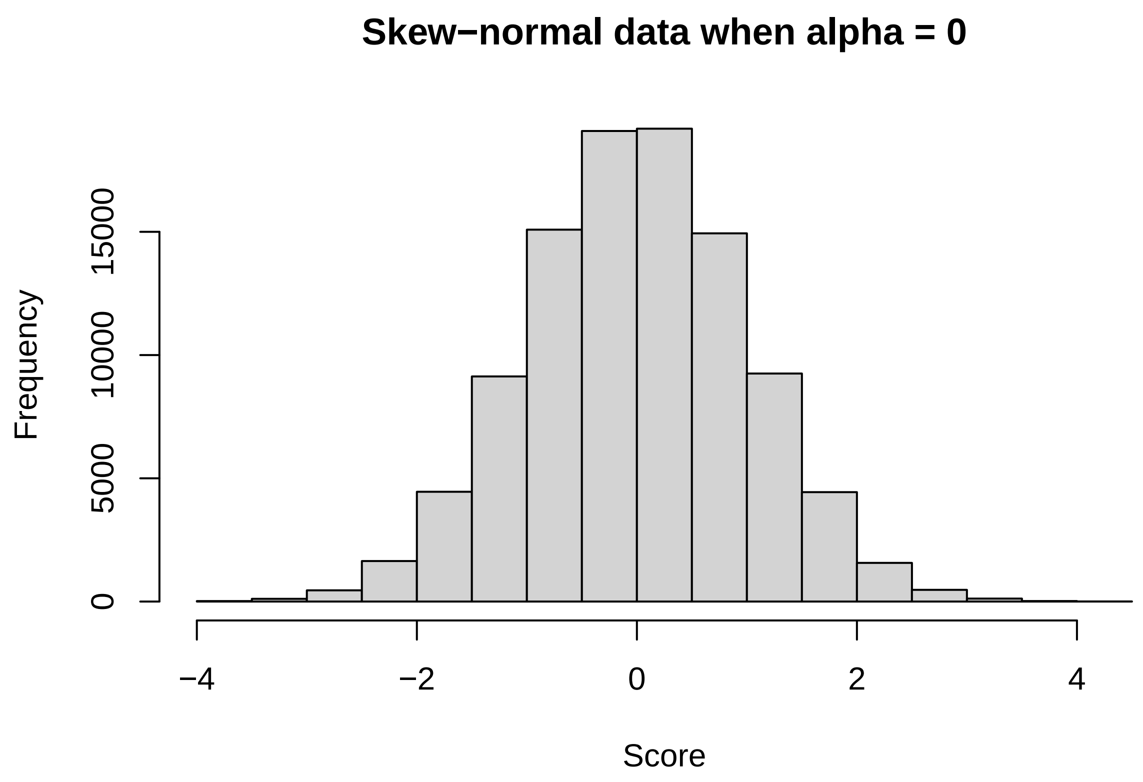
NB we reuse the data generation function from above.

```
# define data analysis function ----
analyse_data <- function(data) {
  res_t_test <- t.test(formula = score ~ condition,
                       data = data,
                       var.equal = TRUE,
                       alternative = "two.sided")

  res <- tibble(p = res_t_test$p.value,
                mean_control = res_t_test$estimate[1],
                mean_intervention = res_t_test$estimate[2])

  return(res)
}
```

Between condition A: normal data

Medium sized difference difference betweeen the groups (difference in location = 0.5)

```r
# reminder of the distribution used in the simulation
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 0) |>
  hist(main = "Skew-normal data when alpha = 0", xlab = "Score")
```



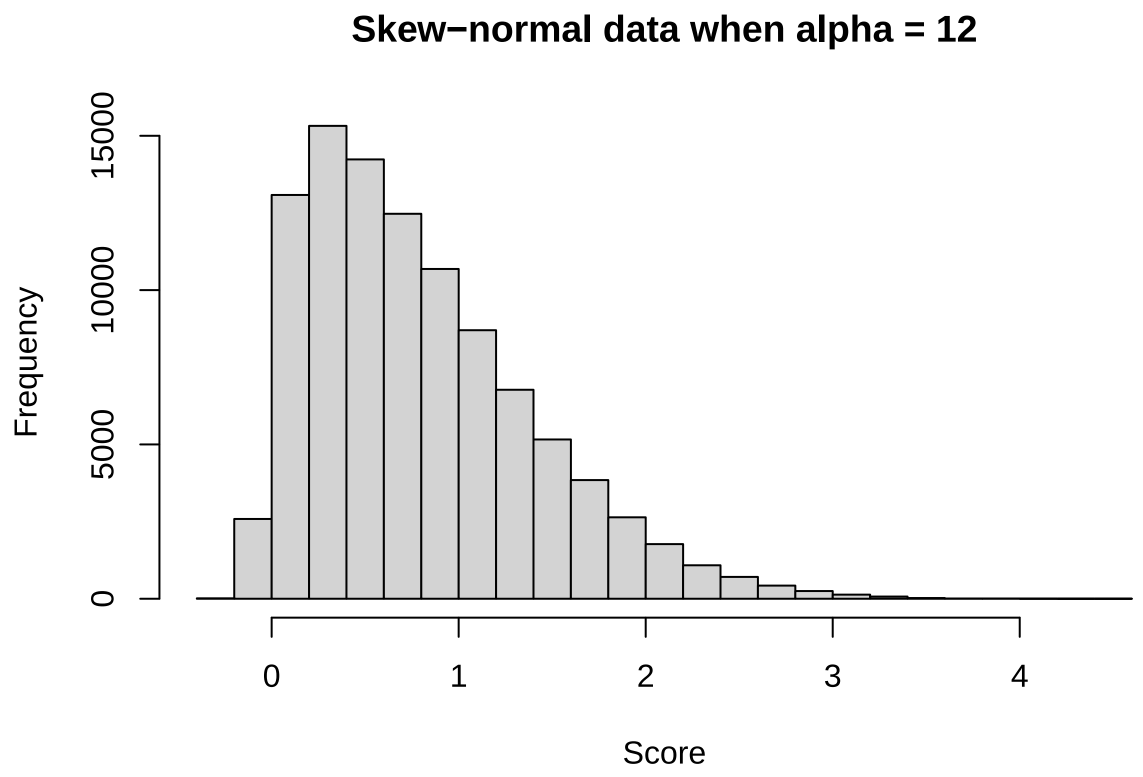
```r
# set the seed ----
# for the pseudo random number generator to make results reproducible
set.seed(42)

# define experiment parameters ----
experiment_parameters_grid_c <- expand_grid(
  n_control = 50,
  n_intervention = 50,
  location_control = 0,
  location_intervention = 0.5, # medium sized difference difference betweeen the groups. proportion of
  scale_control = 1,
  scale_intervention = 1,
  skew_control = 0,
  skew_intervention = 0,
  iteration = 1:10000
)
```

```r
# run simulation ----
simulation_c <-
  # using the experiment parameters
  experiment_parameters_grid_c |>

  # generate data using the data generating function and the parameters relevant to data generation
  mutate(generated_data = pmap(list(n_control,
                                    n_intervention,
                                    location_control,
                                    location_intervention,
                                    scale_control,
                                    scale_intervention,
                                    skew_control,
                                    skew_intervention),
                               generate_data)) |>

  # apply the analysis function to the generated data using the parameters relevant to analysis
  mutate(analysis_results = pmap(list(generated_data),
                                 analyse_data))


# summarise simulation results over the iterations ----
simulation_c_summary <- simulation_c |>
  unnest(analysis_results) |>
  summarize(sample_mean_control = mean(mean_control),
            sample_mean_intervention = mean(mean_intervention)) |>
  mutate(pop_distribution = "normal",
         pop_location_control = 0,
         pop_location_intervention = 0.5)
```

Between condition B: highly skewed data

Medium sized difference difference betweeen the groups (difference in location = 0.5)

```r
# reminder of the distribution used in the simulation
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 12) |>
  hist(main = "Skew-normal data when alpha = 12", xlab = "Score")
```



```r
# set the seed ----
# for the pseudo random number generator to make results reproducible
set.seed(42)

# define experiment parameters ----
experiment_parameters_grid_d <- expand_grid(
  n_control = 50,
  n_intervention = 50,
  location_control = 0,
  location_intervention = 0.5, # medium sized difference difference betweeen the groups. proportion of
  scale_control = 1,
  scale_intervention = 1,
  skew_control = 12, # data in both conditions is highly skewed
  skew_intervention = 12, # data in both conditions is highly skewed
  iteration = 1:10000
)

# run simulation ----
simulation_d <-
  # using the experiment parameters
  experiment_parameters_grid_d |>

  # generate data using the data generating function and the parameters relevant to data generation
  mutate(generated_data = pmap(list(n_control,
                                    n_intervention,
                                    location_control,
                                    location_intervention,
                                    scale_control,
                                    scale_intervention,
                                    skew_control,
                                    skew_intervention),
                               generate_data)) |>

  # apply the analysis function to the generated data using the parameters relevant to analysis
  mutate(analysis_results = pmap(list(generated_data),
                                 analyse_data))


# summarise simulation results over the iterations ----
simulation_d_summary <- simulation_d |>
  unnest(analysis_results) |>
  summarize(sample_mean_control = mean(mean_control),
            sample_mean_intervention = mean(mean_intervention)) |>
  mutate(pop_distribution = "non-normal",
         pop_location_control = 0,
         pop_location_intervention = 0.5)
```

Summarize results

```
bind_rows(
  simulation_c_summary,
  simulation_d_summary
) |>
  relocate(pop_distribution, pop_location_control, sample_mean_control, pop_location_intervention, sampl
  mutate(sample_diff = sample_mean_intervention - sample_mean_control) |>
  mutate_if(is.numeric, janitor::round_half_up, digits = 2) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

pop_distribution	pop_location_control	sample_mean_control	pop_location_intervention	sample_mean_interver
normal	0	0.0	0.5	
non-normal	0	0.8	0.5	

- The sample means do match the population locations when the data is normal (i.e., skew normal when alpha = 0)
- The sample means do *not* match the population locations when the data is highly non-normal (i.e., highly skewed, alpha = 12)
- Nonetheless, the differences in sample means between the conditions *does* match the difference in population locations even when the data is highly non-normal.

Wrapping up

Why don't the sample means match the population locations? Because mean and location are comparable but not the same thing. A simple demonstration using a large sample shows us this:

```
rnorm(n = 1000000,
      mean = 0.45,
      sd = 1) |>
```

```r
  mean() |>
  round_half_up(digits = 2)
```

```
## [1] 0.45
```

```r
rsn(n = 1000000,
    xi = 0.45, # location, akin to mean but not equal to mean
    omega = 1,
    alpha = 12) |>
  mean() |>
  round_half_up(digits = 2)
```

```
## [1] 1.25
```

So, why do we care about assumptions not being violated, if it doesn't change either (a) the p value false positive rate or (b) the (unstandardized) estimate of effect size?

In class we will brainstorm answers to this question together.

Session info

```r
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22631)
## 
## Matrix products: default
## 
## 
## locale:
## [1] LC_COLLATE=German_Switzerland.utf8  LC_CTYPE=German_Switzerland.utf8
## [3] LC_MONETARY=German_Switzerland.utf8 LC_NUMERIC=C
## [5] LC_TIME=German_Switzerland.utf8
## 
## time zone: Europe/Zurich
## tzcode source: internal
## 
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
## 
## other attached packages:
## [1] janitor_2.2.0    kableExtra_1.4.0 knitr_1.46       sn_2.1.1
## [5] ggplot2_3.5.1    purrr_1.0.2      dplyr_1.1.4      tidyr_1.3.1
## 
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.5       highr_0.10         compiler_4.3.2
##  [4] tidyselect_1.2.1   xml2_1.3.6         stringr_1.5.1
##  [7] snakecase_0.11.1   systemfonts_1.0.6  scales_1.3.0
## [10] yaml_2.3.8         fastmap_1.1.1      R6_2.5.1
## [13] generics_0.1.3     tibble_3.2.1       munsell_0.5.1
## [16] lubridate_1.9.3    svglite_2.1.3      pillar_1.9.0
## [19] rlang_1.1.3        utf8_1.2.4         stringi_1.8.3
## [22] xfun_0.43          timechange_0.3.0   viridisLite_0.4.2
## [25] cli_3.6.2          withr_3.0.0        magrittr_2.0.3
## [28] digest_0.6.35      grid_4.3.2         rstudioapi_0.16.0
## [31] lifecycle_1.0.4    vctrs_0.6.5        mnormt_2.1.1
## [34] evaluate_0.23      glue_1.7.0         numDeriv_2016.8-1.1
## [37] fansi_1.0.6        colorspace_2.1-0   rmarkdown_2.26
## [40] tools_4.3.2        pkgconfig_2.0.3    htmltools_0.5.8.1
```