# The statistical power of assumptions tests and the conditional use of non-parameteric tests

Ian Hussey

01 Mai, 2024

## Contents



```r
# dependencies ----
# repeated here for the sake of completeness

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
```

```r
library(forcats)
library(readr)
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
```

```r
library(knitr)
```

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

```r
library(kableExtra)
```

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

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

# Overview

What do most statistics textbooks tell you to do when trying to test if two groups' means differ?

1. Check if assumptions of an independent Student's t-test are met, e.g., normality of data and homogeneity of variances.
2. If so, run and interpret an independent Student's t-test.
3. If not, then perhaps either 'interpret results with caution' (which always feels vague) or run and interpret a non-parametric test instead.

Why? What benefits are there for doing so? Or what bad things happen if you don't?

In a previous session, we observed that violations of the assumption of normality actually has very little impact on the statistical power of a t-test, as long as the two conditions have similarly non-normal data, which is plausible in many situations. Of course, non-normality does distort estimates of population parameters and standardized effect sizes - but often not the p values themselves. This lesson seeks to answer two related questions:

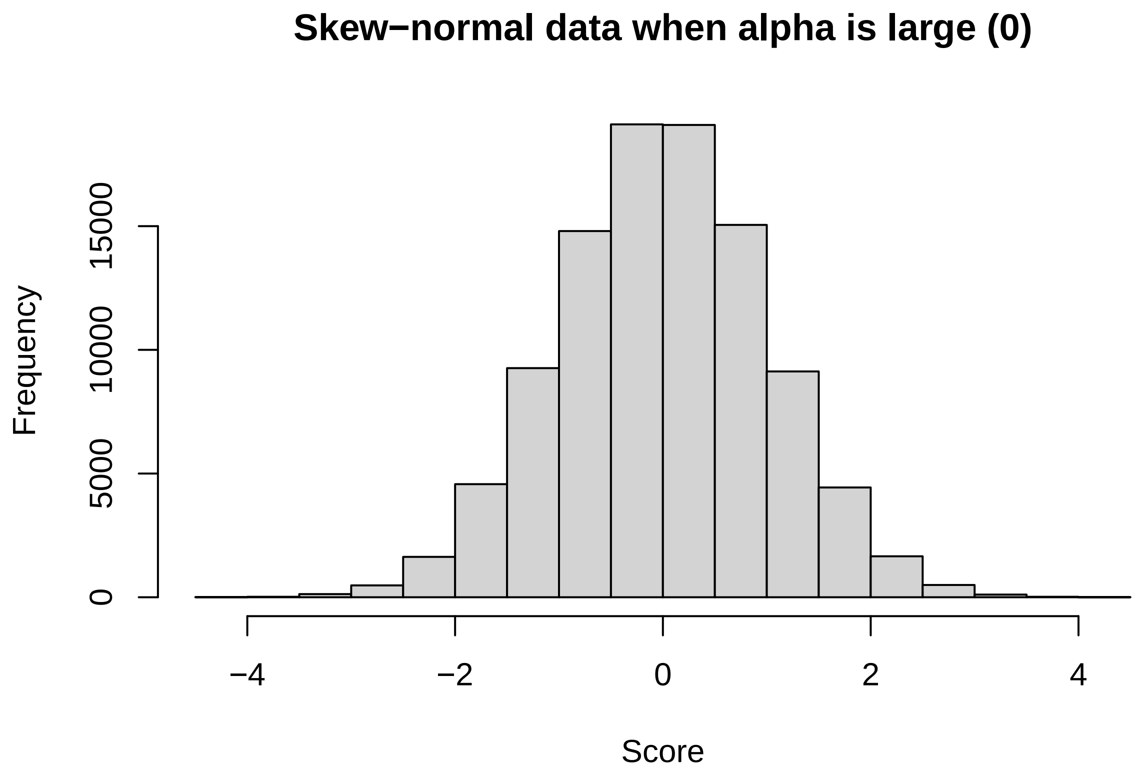
1. Just like hypothesis tests, assumptions tests are just inferential tests of other properties (e.g., differences in SDs rather than differences in means), and as such they have false-positive rates and false-negative rates (statistical power). What is the power of these tests under different degrees of violations of assumptions? I.e. what proportion of the time do they get it wrong?
2. What is the aggregate benefit of choosing a hypothesis test based on the results of assumption tests? This multi-step researcher behaviour can itself be simulated.

# Assumption of normality

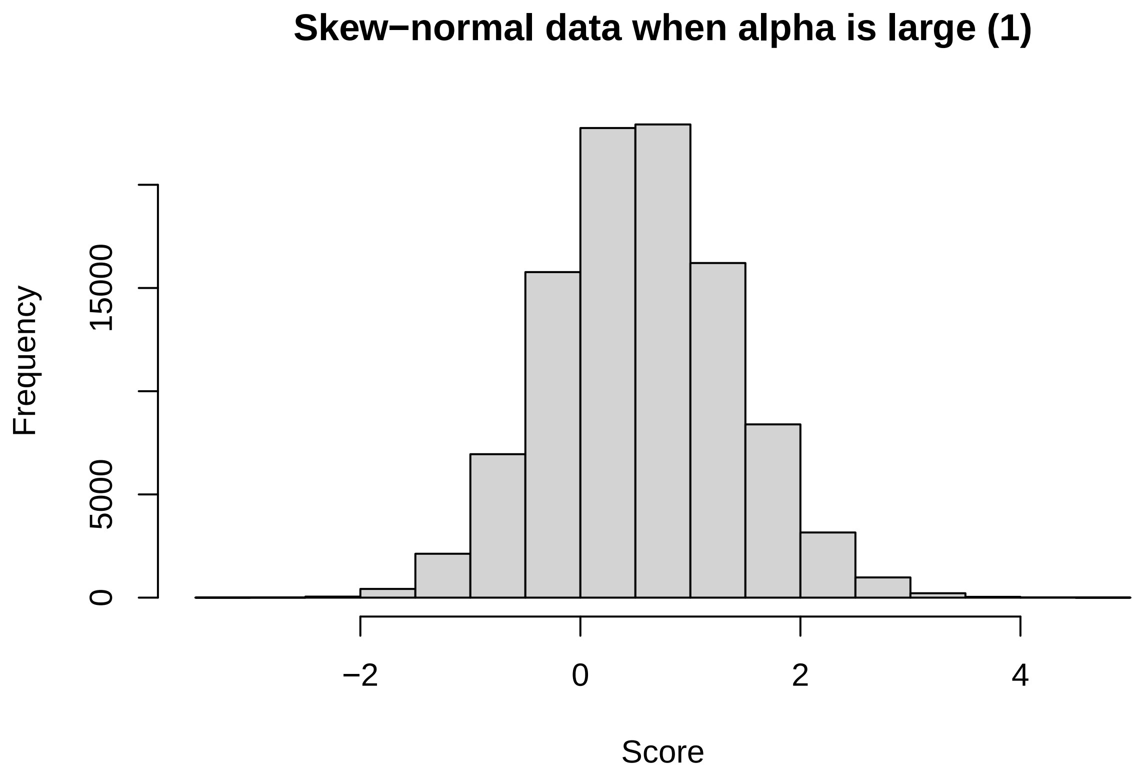
## Illustrate non-normality

In this case using skewed data, although non-normality could take very many different forms.

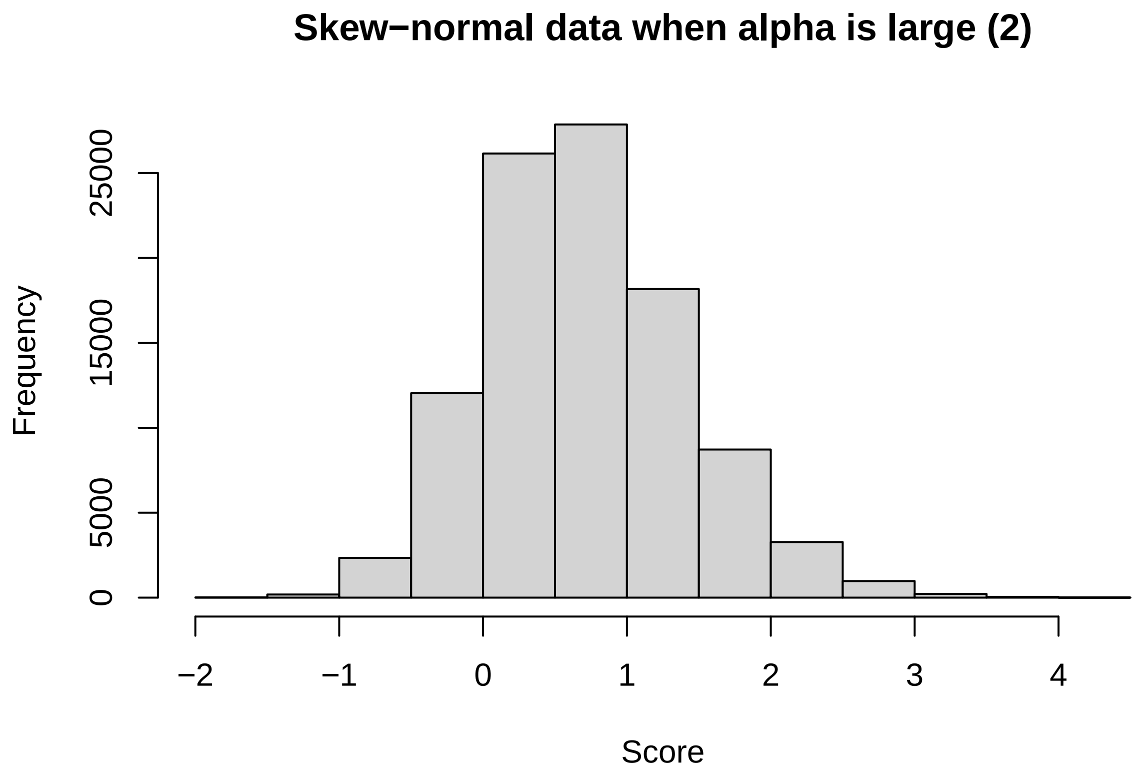
```r
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 0) |>
  hist(main = "Skew-normal data when alpha is large (0)", xlab = "Score")
```



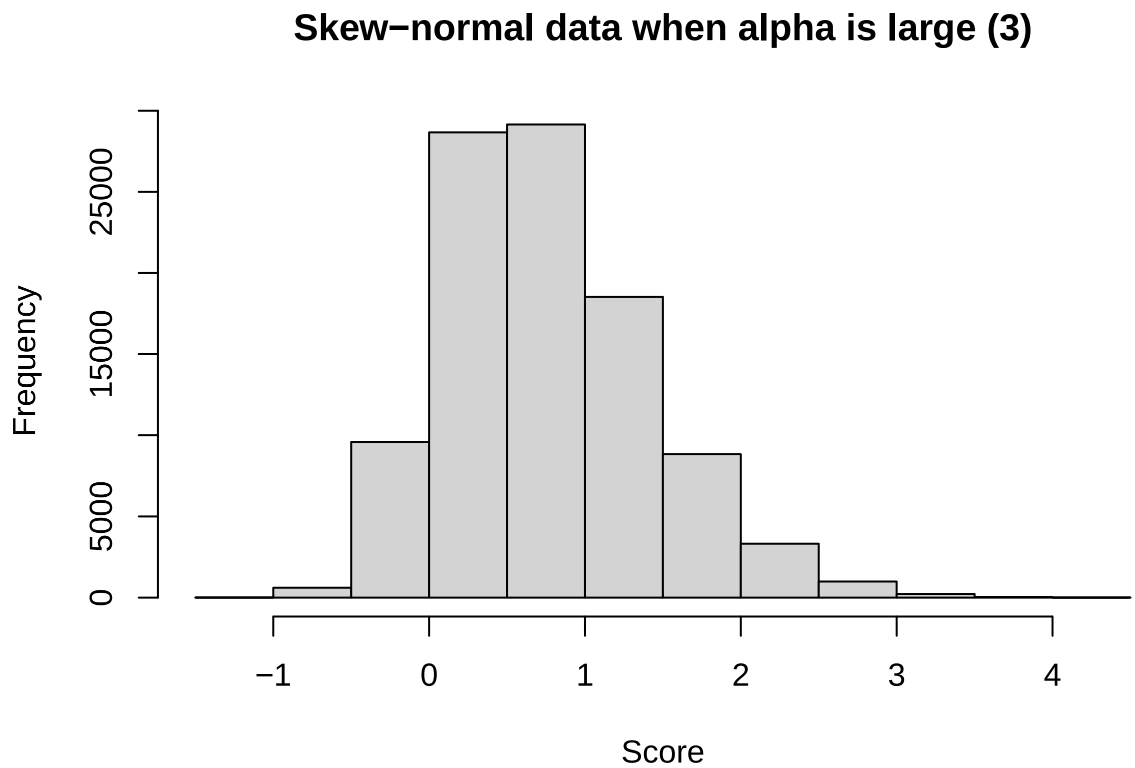

```r
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 1) |>
  hist(main = "Skew-normal data when alpha is large (1)", xlab = "Score")
```



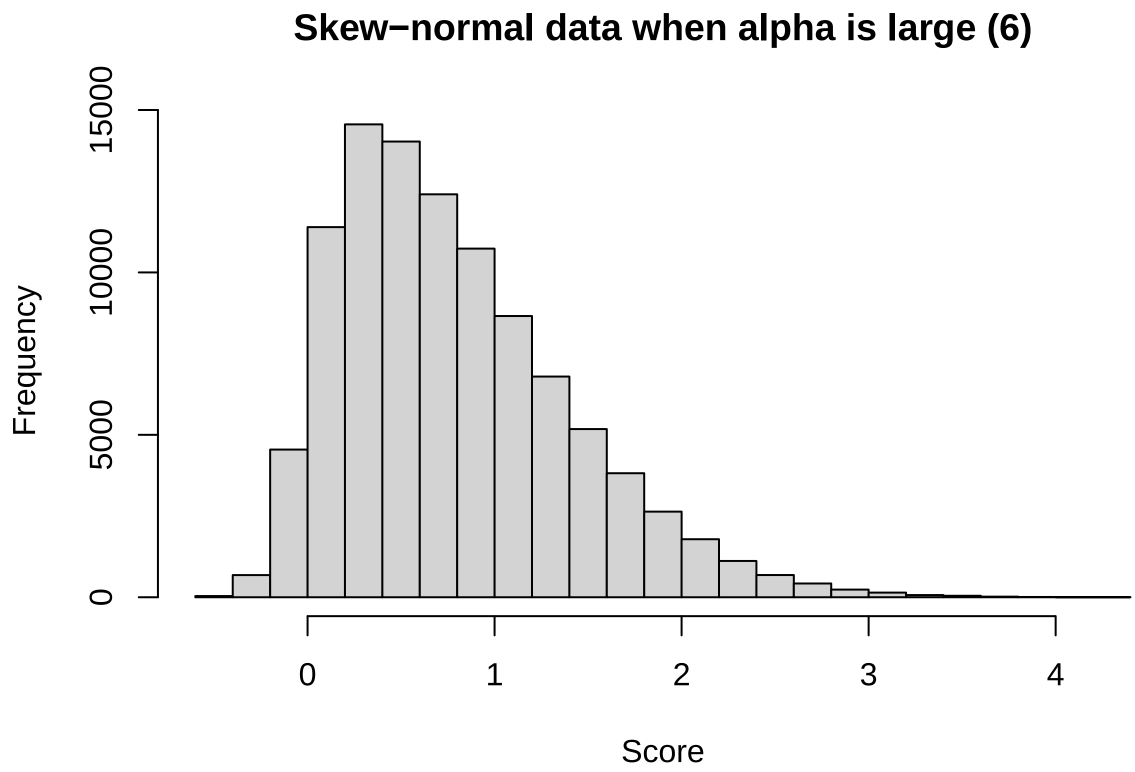

```
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 2) |>
  hist(main = "Skew-normal data when alpha is large (2)", xlab = "Score")
```



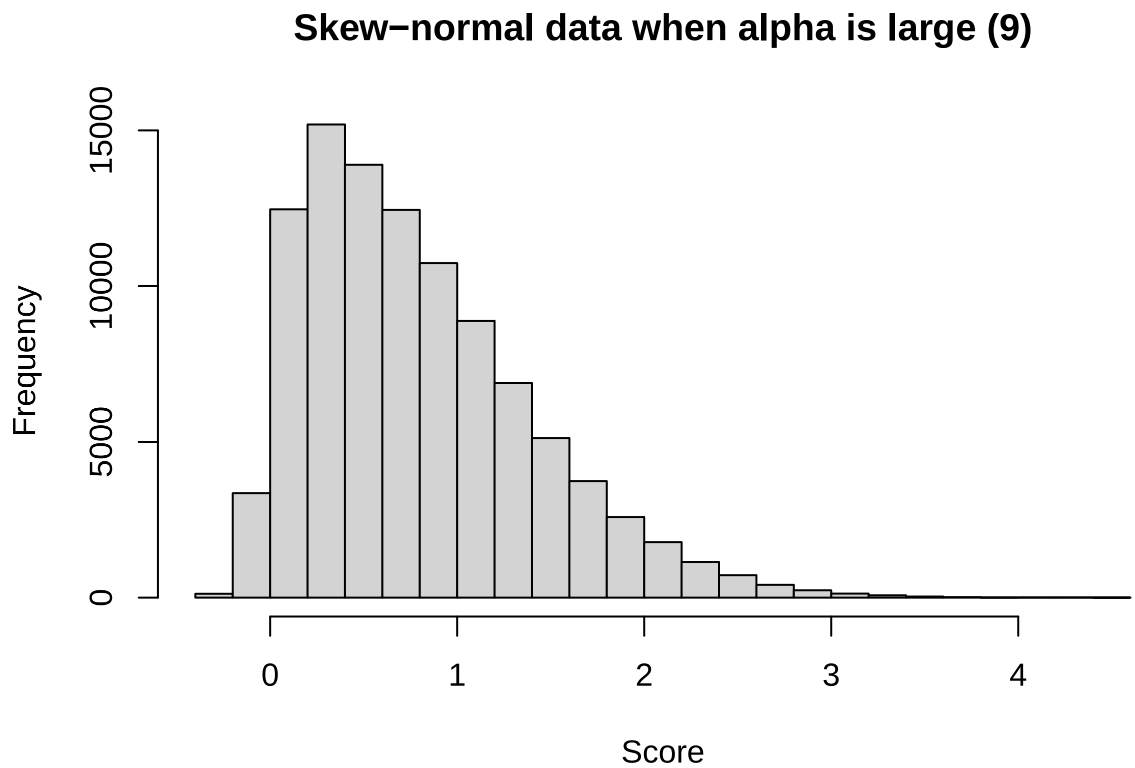

```
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 3) |>
  hist(main = "Skew-normal data when alpha is large (3)", xlab = "Score")
```



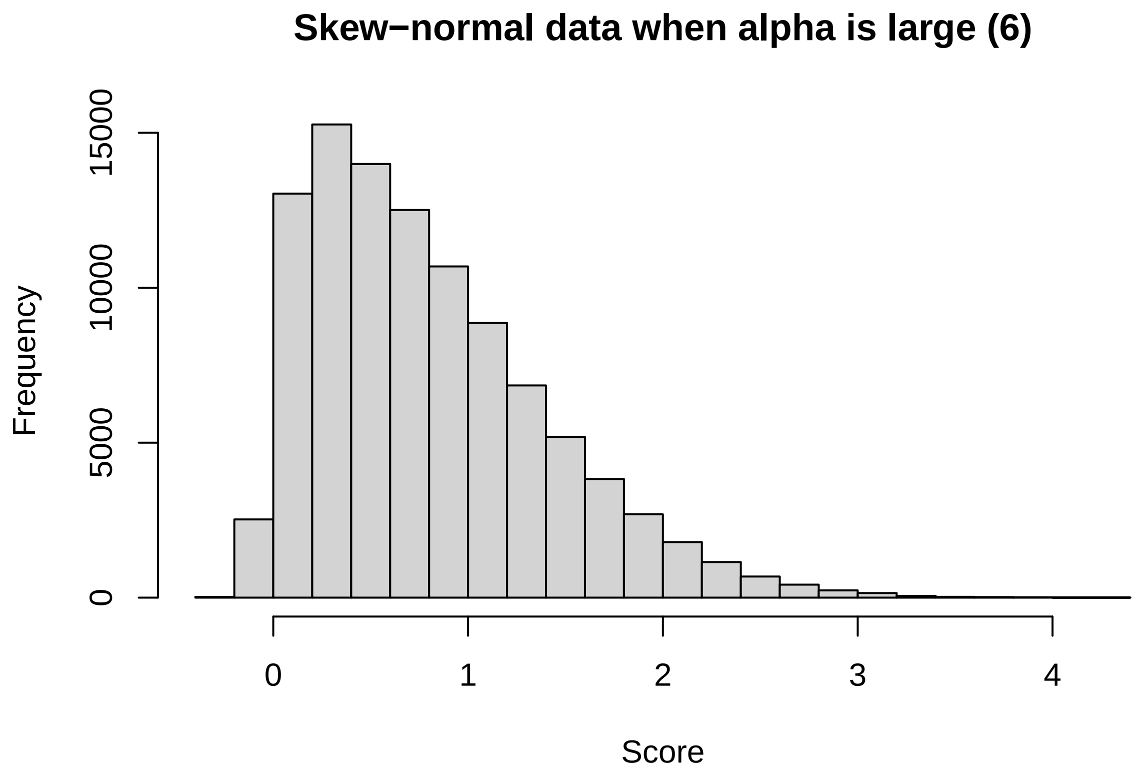

```r
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 6) |>
  hist(main = "Skew-normal data when alpha is large (6)", xlab = "Score")
```




```r
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 9) |>
  hist(main = "Skew-normal data when alpha is large (9)", xlab = "Score")
```




```
rsn(n = 100000,
    xi = 0,
    omega = 1,
    alpha = 12) |>
  hist(main = "Skew-normal data when alpha is large (6)", xlab = "Score")
```




## Power of Kolmogov-Smirnov test with skew-normal data

The assumption of normality can be assessed using multiple tests, including the Kolmogov-Smirnov test. This test assesses whether two distributions come from the same distribution. It is known to have low power: i.e., in low sample sizes, this test seldomly correctly detects when the two samples come from different distributions.

Slightly confusingly, although the tests compares two samples, it can also be used to test *one* sample for normality by comparing the one *observed* sample with a second hypothetical perfectly normal distribution.

Let's demonstrate that the test does indeed have low power before we move on to an alternative test.

### In a single sample

This generates data from a single sample and the analysis only tests the assumption of normality. This is a useful first step to developing the code.

```
# remove all objects from environment ----
rm(list = ls())


# dependencies ----
# repeated here for the sake of completeness

library(tidyr)
library(dplyr)
library(forcats)
library(readr)
```

```r
library(purrr)
library(ggplot2)
library(sn)
library(knitr)
library(kableExtra)

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


# define data generating function ----
generate_data <- function(n,
                          location, # location, akin to mean
                          scale, # scale, akin to SD
                          skew) { # slant/skew. When 0, produces normal/gaussian data

  data <-
    tibble(score = rsn(n = n,
                       xi = location, # location, akin to mean
                       omega = scale, # scale, akin to SD
                       alpha = skew)) # slant/skew. When 0, produces normal/gaussian data

  return(data)
}


# define data analysis function ----
analyse_data <- function(data) {

  fit <- ks.test(data$score, "pnorm", mean = mean(data$score), sd = sd(data$score))
  results <- tibble(p = fit$p.value)

  return(results)
}


# define experiment parameters ----
experiment_parameters_grid <- expand_grid(
  n = seq(from = 10, to = 100, by = 10),
  location = 0, # location, akin to mean
  scale = 1,    # scale, akin to SD
  skew = c(0, 1, 2, 3, 6, 9, 12),     # slant/skew. When 0, produces normal/gaussian data
  iteration = 1:1000
)


# run simulation ----
simulation <-
  # using the experiment parameters
  experiment_parameters_grid |>

  # generate data using the data generating function and the parameters relevant to data generation
```

```r
  mutate(generated_data = pmap(list(n,
                                    location,
                                    scale,
                                    skew),
                               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 ----
## ie what proportion of p values are significant (< .05)
simulation_summary <- simulation |>
  unnest(analysis_results) |>
  mutate(n = as.factor(n)) |>
  group_by(n,
           location,
           scale,
           skew) |>
  summarize(proportion_of_significant_results = mean(p < .05),
            .groups = "drop") #true of false positives

simulation_summary |>
  #filter(proportion_of_significant_results >= .8) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n | location | scale | skew | proportion_of_significant_results |
|---|---|---|---|---|
| 10 | 0 | 1 | 0 | 0.000 |
| 10 | 0 | 1 | 1 | 0.000 |
| 10 | 0 | 1 | 2 | 0.001 |
| 10 | 0 | 1 | 3 | 0.000 |
| 10 | 0 | 1 | 6 | 0.000 |
| 10 | 0 | 1 | 9 | 0.000 |
| 10 | 0 | 1 | 12 | 0.000 |
| 20 | 0 | 1 | 0 | 0.000 |
| 20 | 0 | 1 | 1 | 0.000 |
| 20 | 0 | 1 | 2 | 0.001 |
| 20 | 0 | 1 | 3 | 0.000 |
| 20 | 0 | 1 | 6 | 0.002 |
| 20 | 0 | 1 | 9 | 0.002 |
| 20 | 0 | 1 | 12 | 0.004 |
| 30 | 0 | 1 | 0 | 0.000 |
| 30 | 0 | 1 | 1 | 0.000 |
| 30 | 0 | 1 | 2 | 0.000 |
| 30 | 0 | 1 | 3 | 0.003 |
| 30 | 0 | 1 | 6 | 0.004 |
| 30 | 0 | 1 | 9 | 0.006 |
| 30 | 0 | 1 | 12 | 0.005 |
| 40 | 0 | 1 | 0 | 0.000 |
| 40 | 0 | 1 | 1 | 0.000 |

| | | | | |
|---|---|---|---|---|
| 40 | 0 | 1 | 2 | 0.002 |
| 40 | 0 | 1 | 3 | 0.004 |
| 40 | 0 | 1 | 6 | 0.016 |
| 40 | 0 | 1 | 9 | 0.012 |
| 40 | 0 | 1 | 12 | 0.015 |
| 50 | 0 | 1 | 0 | 0.000 |
| 50 | 0 | 1 | 1 | 0.001 |
| 50 | 0 | 1 | 2 | 0.001 |
| 50 | 0 | 1 | 3 | 0.002 |
| 50 | 0 | 1 | 6 | 0.018 |
| 50 | 0 | 1 | 9 | 0.026 |
| 50 | 0 | 1 | 12 | 0.026 |
| 60 | 0 | 1 | 0 | 0.000 |
| 60 | 0 | 1 | 1 | 0.000 |
| 60 | 0 | 1 | 2 | 0.000 |
| 60 | 0 | 1 | 3 | 0.003 |
| 60 | 0 | 1 | 6 | 0.024 |
| 60 | 0 | 1 | 9 | 0.040 |
| 60 | 0 | 1 | 12 | 0.041 |
| 70 | 0 | 1 | 0 | 0.000 |
| 70 | 0 | 1 | 1 | 0.000 |
| 70 | 0 | 1 | 2 | 0.002 |
| 70 | 0 | 1 | 3 | 0.011 |
| 70 | 0 | 1 | 6 | 0.055 |
| 70 | 0 | 1 | 9 | 0.062 |
| 70 | 0 | 1 | 12 | 0.076 |
| 80 | 0 | 1 | 0 | 0.000 |
| 80 | 0 | 1 | 1 | 0.001 |
| 80 | 0 | 1 | 2 | 0.006 |
| 80 | 0 | 1 | 3 | 0.014 |
| 80 | 0 | 1 | 6 | 0.062 |
| 80 | 0 | 1 | 9 | 0.075 |
| 80 | 0 | 1 | 12 | 0.097 |
| 90 | 0 | 1 | 0 | 0.000 |
| 90 | 0 | 1 | 1 | 0.000 |
| 90 | 0 | 1 | 2 | 0.004 |
| 90 | 0 | 1 | 3 | 0.011 |
| 90 | 0 | 1 | 6 | 0.095 |
| 90 | 0 | 1 | 9 | 0.105 |
| 90 | 0 | 1 | 12 | 0.118 |
| 100 | 0 | 1 | 0 | 0.000 |
| 100 | 0 | 1 | 1 | 0.000 |
| 100 | 0 | 1 | 2 | 0.002 |
| 100 | 0 | 1 | 3 | 0.022 |
| 100 | 0 | 1 | 6 | 0.095 |
| 100 | 0 | 1 | 9 | 0.135 |
| 100 | 0 | 1 | 12 | 0.150 |

Very low power in common sample sizes. Let's examine the Shapiro-Wilk's test instead.

## Power of Shapiro-Wilk's test with skew-normal data

### In a single sample

This generates data from a single sample and the analysis only tests the assumption of normality. This is a useful first step to developing the code.

```r
# remove all objects from environment ----
rm(list = ls())


# dependencies ----
# repeated here for the sake of completeness

library(tidyr)
library(dplyr)
library(forcats)
library(readr)
library(purrr)
library(ggplot2)
library(sn)
library(knitr)
library(kableExtra)

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


# define data generating function ----
generate_data <- function(n,
                          location, # location, akin to mean
                          scale, # scale, akin to SD
                          skew) { # slant/skew. When 0, produces normal/gaussian data

  data <-
    tibble(score = rsn(n = n,
                       xi = location, # location, akin to mean
                       omega = scale, # scale, akin to SD
                       alpha = skew)) # slant/skew. When 0, produces normal/gaussian data

  return(data)
}


# define data analysis function ----
analyse_data <- function(data) {

  fit <- shapiro.test(data$score)
  results <- tibble(p = fit$p.value)

  return(results)
}
```

```r
# define experiment parameters ----
experiment_parameters_grid <- expand_grid(
  n = seq(from = 10, to = 100, by = 10),
  location = 0, # location, akin to mean
  scale = 1,    # scale, akin to SD
  skew = c(0, 1, 2, 3, 6, 9, 12),     # slant/skew. When 0, produces normal/gaussian data
  iteration = 1:1000
)


# run simulation ----
simulation <-
  # using the experiment parameters
  experiment_parameters_grid |>

  # generate data using the data generating function and the parameters relevant to data generation
  mutate(generated_data = pmap(list(n,
                                    location,
                                    scale,
                                    skew),
                               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 ----
## ie what proportion of p values are significant (< .05)
simulation_summary <- simulation |>
  unnest(analysis_results) |>
  mutate(n = as.factor(n)) |>
  group_by(n,
           location,
           scale,
           skew) |>
  summarize(proportion_of_significant_results = mean(p < .05),
            .groups = "drop")

simulation_summary |>
  filter(proportion_of_significant_results >= .95) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n | location | scale | skew | proportion_of_significant_results |
|---|---|---|---|---|
| 70 | 0 | 1 | 12 | 0.973 |
| 80 | 0 | 1 | 9 | 0.972 |
| 80 | 0 | 1 | 12 | 0.983 |
| 90 | 0 | 1 | 6 | 0.961 |
| 90 | 0 | 1 | 9 | 0.992 |
| 90 | 0 | 1 | 12 | 0.998 |
| 100 | 0 | 1 | 6 | 0.979 |
| 100 | 0 | 1 | 9 | 0.994 |

| | | | | |
|---|---|---|---|---|
| 100 | 0 | 1 | 12 | 0.999 |

**In either of two samples**

To make the stimulation more realistic, we should test for normality in each of two samples, and return a decision of non-normality if it is found in either.

```
# remove all objects from environment ----
rm(list = ls())


# dependencies ----
# repeated here for the sake of completeness

library(tidyr)
library(dplyr)
library(forcats)
library(readr)
library(purrr)
library(ggplot2)
library(sn)
library(knitr)
library(kableExtra)

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


# define data generating function ----
generate_data <- function(n,
                          location, # location, akin to mean
                          scale, # scale, akin to SD
                          skew) { # slant/skew. When 0, produces normal/gaussian data


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

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

  data <- bind_rows(data_control,
                    data_intervention)

  return(data)
```

```r
}


# define data analysis function ----
analyse_data <- function(data) {

  fit_intervention <- shapiro.test(data$score[data$condition == "intervention"])
  fit_control <- shapiro.test(data$score[data$condition == "control"])

  results <- tibble(p_intervention = fit_intervention$p.value,
                    p_control = fit_control$p.value)

  return(results)
}


# define experiment parameters ----
experiment_parameters_grid <- expand_grid(
  n = seq(from = 10, to = 100, by = 10), # n per condition, not total
  location = 0, # location, akin to mean
  scale = 1,    # scale, akin to SD
  skew = c(0, 1, 2, 3, 6, 9, 12),     # slant/skew. When 0, produces normal/gaussian data
  iteration = 1:1000
)


# run simulation ----
simulation <-
  # using the experiment parameters
  experiment_parameters_grid |>

  # generate data using the data generating function and the parameters relevant to data generation
  mutate(generated_data = pmap(list(n,
                                    location,
                                    scale,
                                    skew),
                               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 ----
## ie what proportion of p values are significant (< .05)
simulation_summary <- simulation |>
  unnest(analysis_results) |>
  mutate(n = as.factor(n),
         lower_p = ifelse(p_intervention < p_control, p_intervention, p_control)) |>
  group_by(n,
           location,
           scale,
           skew) |>
  summarize(proportion_of_significant_results = mean(lower_p < .05),
```

```
            .groups = "drop")

simulation_summary |>
  filter(proportion_of_significant_results >= .95) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n | location | scale | skew | proportion_of_significant_results |
|---|---|---|---|---|
| 40 | 0 | 1 | 12 | 0.952 |
| 50 | 0 | 1 | 9 | 0.979 |
| 50 | 0 | 1 | 12 | 0.983 |
| 60 | 0 | 1 | 6 | 0.971 |
| 60 | 0 | 1 | 9 | 0.989 |
| 60 | 0 | 1 | 12 | 0.994 |
| 70 | 0 | 1 | 6 | 0.987 |
| 70 | 0 | 1 | 9 | 0.999 |
| 70 | 0 | 1 | 12 | 0.999 |
| 80 | 0 | 1 | 6 | 0.994 |
| 80 | 0 | 1 | 9 | 0.999 |
| 80 | 0 | 1 | 12 | 1.000 |
| 90 | 0 | 1 | 6 | 0.999 |
| 90 | 0 | 1 | 9 | 1.000 |
| 90 | 0 | 1 | 12 | 1.000 |
| 100 | 0 | 1 | 6 | 0.998 |
| 100 | 0 | 1 | 9 | 1.000 |
| 100 | 0 | 1 | 12 | 1.000 |

### Conditionally running a Welches' t-test or a Wilcoxon signed-rank test depending on Shapiro-Wilk's test for normality

This stimulation tests normality in both conditions' data, as well as testing for differences in the central tendency using both parametric and non-parametric tests. Which test of the differences in central tendency is used for each simulated data set is determined by whether the assumption of normality is detectably violated.

```
# remove all objects from environment ----
rm(list = ls())


# dependencies ----
# repeated here for the sake of completeness

library(tidyr)
library(dplyr)
library(forcats)
library(readr)
library(purrr)
library(ggplot2)
library(sn)
library(knitr)
library(kableExtra)
```

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


# define data generating function ----
generate_data <- function(n,
                          location_control, # location, akin to mean
                          location_intervention,
                          scale, # scale, akin to SD
                          skew) { # slant/skew. When 0, produces normal/gaussian data


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

  data_intervention <-
    tibble(condition = "intervention",
           score = rsn(n = n,
                       xi = location_intervention, # location, akin to mean
                       omega = scale, # scale, akin to SD
                       alpha = skew)) # 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) {

  assumption_test_intervention   <- shapiro.test(data$score[data$condition == "intervention"])
  assumption_test_control        <- shapiro.test(data$score[data$condition == "control"])

  hypothesis_test_welches_t      <- t.test(formula = score ~ condition,
                                           data = data,
                                           var.equal = TRUE,
                                           alternative = "two.sided")

  hypothesis_test_mann_whitney_u <- wilcox.test(formula = score ~ condition,
                                                data = data,
                                                alternative = "two.sided")

  results <- tibble(
    assumption_test_p_intervention = assumption_test_intervention$p.value,
    assumption_test_p_control = assumption_test_control$p.value,
    hypothesis_test_p_welches_t = hypothesis_test_welches_t$p.value,
```

```
    hypothesis_test_p_mann_whitney_u = hypothesis_test_mann_whitney_u$p.value
  ) |>
    mutate(hypothesis_test_p_conditional = ifelse(min(assumption_test_p_intervention, assumption_test_p_
                                              hypothesis_test_p_mann_whitney_u,
                                              hypothesis_test_p_welches_t))

  return(results)
}


# define experiment parameters ----
experiment_parameters_grid <- expand_grid(
  n = seq(from = 10, to = 100, by = 10), # n per condition, not total
  location_control = 0, # location, akin to mean
  location_intervention = 0.2,
  scale = 1,    # scale, akin to SD
  skew = c(0, 1, 2, 3, 6, 9, 12),     # slant/skew. When 0, produces normal/gaussian data
  iteration = 1:1000
)


# run simulation ----
simulation <-
  # using the experiment parameters
  experiment_parameters_grid |>

  # generate data using the data generating function and the parameters relevant to data generation
  mutate(generated_data = pmap(list(n,
                                    location_control,
                                    location_intervention,
                                    scale,
                                    skew),
                               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 ----
## ie what proportion of p values are significant (< .05)
simulation_summary <- simulation |>
  unnest(analysis_results) |>
  mutate(n_per_group = as.factor(n)) |>
  group_by(n_per_group,
           location_control,
           location_intervention,
           scale,
           skew) |>
  summarize(power_assumption_test = mean(assumption_test_p_intervention < .05 | assumption_test_p_contr
            power_u = mean(hypothesis_test_p_mann_whitney_u < .05),
            power_t = mean(hypothesis_test_p_welches_t < .05),
            power_u = mean(hypothesis_test_p_mann_whitney_u < .05),
            power_conditional = mean(hypothesis_test_p_conditional < .05),
```

```r
             .groups = "drop") |>
  mutate(conditional_better_than_t = power_conditional > power_t,
         conditional_better_than_u = power_conditional > power_u,
         u_better_than_t = power_u > power_t,
         conditional_much_better_than_t = (power_conditional - power_t) >= .05,
         conditional_much_better_than_u = (power_conditional - power_u) >= .05,
         u_much_better_than_t = (power_u - power_t) >= .05)

simulation_summary |>
  arrange(skew, n_per_group) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n_per_group | location_control | location_intervention | scale | skew | power_assumption_test | power_u | power_t |
|---|---|---|---|---|---|---|---|
| 10 | 0 | 0.2 | 1 | 0 | 0.090 | 0.062 | 0.072 |
| 20 | 0 | 0.2 | 1 | 0 | 0.108 | 0.090 | 0.095 |
| 30 | 0 | 0.2 | 1 | 0 | 0.087 | 0.124 | 0.136 |
| 40 | 0 | 0.2 | 1 | 0 | 0.106 | 0.140 | 0.146 |
| 50 | 0 | 0.2 | 1 | 0 | 0.092 | 0.174 | 0.177 |
| 60 | 0 | 0.2 | 1 | 0 | 0.099 | 0.180 | 0.190 |
| 70 | 0 | 0.2 | 1 | 0 | 0.088 | 0.191 | 0.191 |
| 80 | 0 | 0.2 | 1 | 0 | 0.097 | 0.207 | 0.224 |
| 90 | 0 | 0.2 | 1 | 0 | 0.093 | 0.254 | 0.270 |
| 100 | 0 | 0.2 | 1 | 0 | 0.099 | 0.277 | 0.286 |
| 10 | 0 | 0.2 | 1 | 1 | 0.106 | 0.069 | 0.079 |
| 20 | 0 | 0.2 | 1 | 1 | 0.125 | 0.105 | 0.107 |
| 30 | 0 | 0.2 | 1 | 1 | 0.098 | 0.144 | 0.151 |
| 40 | 0 | 0.2 | 1 | 1 | 0.103 | 0.165 | 0.163 |
| 50 | 0 | 0.2 | 1 | 1 | 0.142 | 0.210 | 0.229 |
| 60 | 0 | 0.2 | 1 | 1 | 0.126 | 0.241 | 0.247 |
| 70 | 0 | 0.2 | 1 | 1 | 0.127 | 0.277 | 0.290 |
| 80 | 0 | 0.2 | 1 | 1 | 0.140 | 0.327 | 0.337 |
| 90 | 0 | 0.2 | 1 | 1 | 0.135 | 0.347 | 0.352 |
| 100 | 0 | 0.2 | 1 | 1 | 0.168 | 0.390 | 0.398 |
| 10 | 0 | 0.2 | 1 | 2 | 0.117 | 0.093 | 0.098 |
| 20 | 0 | 0.2 | 1 | 2 | 0.191 | 0.137 | 0.141 |
| 30 | 0 | 0.2 | 1 | 2 | 0.252 | 0.211 | 0.211 |
| 40 | 0 | 0.2 | 1 | 2 | 0.312 | 0.261 | 0.261 |
| 50 | 0 | 0.2 | 1 | 2 | 0.354 | 0.283 | 0.283 |
| 60 | 0 | 0.2 | 1 | 2 | 0.428 | 0.343 | 0.348 |
| 70 | 0 | 0.2 | 1 | 2 | 0.451 | 0.389 | 0.400 |
| 80 | 0 | 0.2 | 1 | 2 | 0.522 | 0.431 | 0.427 |
| 90 | 0 | 0.2 | 1 | 2 | 0.558 | 0.485 | 0.490 |
| 100 | 0 | 0.2 | 1 | 2 | 0.617 | 0.507 | 0.509 |
| 10 | 0 | 0.2 | 1 | 3 | 0.186 | 0.088 | 0.090 |
| 20 | 0 | 0.2 | 1 | 3 | 0.294 | 0.147 | 0.148 |
| 30 | 0 | 0.2 | 1 | 3 | 0.442 | 0.216 | 0.222 |
| 40 | 0 | 0.2 | 1 | 3 | 0.550 | 0.275 | 0.270 |
| 50 | 0 | 0.2 | 1 | 3 | 0.671 | 0.330 | 0.326 |
| 60 | 0 | 0.2 | 1 | 3 | 0.724 | 0.396 | 0.377 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 70 | 0 | 0.2 | 1 | 3 | 0.798 | 0.451 | 0.432 |
| 80 | 0 | 0.2 | 1 | 3 | 0.845 | 0.495 | 0.461 |
| 90 | 0 | 0.2 | 1 | 3 | 0.867 | 0.592 | 0.568 |
| 100 | 0 | 0.2 | 1 | 3 | 0.922 | 0.603 | 0.581 |
| 10 | 0 | 0.2 | 1 | 6 | 0.251 | 0.104 | 0.102 |
| 20 | 0 | 0.2 | 1 | 6 | 0.538 | 0.175 | 0.167 |
| 30 | 0 | 0.2 | 1 | 6 | 0.731 | 0.259 | 0.229 |
| 40 | 0 | 0.2 | 1 | 6 | 0.855 | 0.342 | 0.303 |
| 50 | 0 | 0.2 | 1 | 6 | 0.934 | 0.384 | 0.335 |
| 60 | 0 | 0.2 | 1 | 6 | 0.971 | 0.473 | 0.425 |
| 70 | 0 | 0.2 | 1 | 6 | 0.987 | 0.505 | 0.464 |
| 80 | 0 | 0.2 | 1 | 6 | 0.994 | 0.588 | 0.542 |
| 90 | 0 | 0.2 | 1 | 6 | 0.999 | 0.642 | 0.585 |
| 100 | 0 | 0.2 | 1 | 6 | 0.998 | 0.686 | 0.623 |
| 10 | 0 | 0.2 | 1 | 9 | 0.312 | 0.084 | 0.093 |
| 20 | 0 | 0.2 | 1 | 9 | 0.608 | 0.210 | 0.188 |
| 30 | 0 | 0.2 | 1 | 9 | 0.802 | 0.260 | 0.224 |
| 40 | 0 | 0.2 | 1 | 9 | 0.929 | 0.343 | 0.312 |
| 50 | 0 | 0.2 | 1 | 9 | 0.979 | 0.442 | 0.399 |
| 60 | 0 | 0.2 | 1 | 9 | 0.989 | 0.496 | 0.431 |
| 70 | 0 | 0.2 | 1 | 9 | 0.999 | 0.568 | 0.483 |
| 80 | 0 | 0.2 | 1 | 9 | 0.999 | 0.627 | 0.556 |
| 90 | 0 | 0.2 | 1 | 9 | 1.000 | 0.652 | 0.595 |
| 100 | 0 | 0.2 | 1 | 9 | 1.000 | 0.728 | 0.649 |
| 10 | 0 | 0.2 | 1 | 12 | 0.324 | 0.109 | 0.112 |
| 20 | 0 | 0.2 | 1 | 12 | 0.619 | 0.211 | 0.189 |
| 30 | 0 | 0.2 | 1 | 12 | 0.830 | 0.309 | 0.270 |
| 40 | 0 | 0.2 | 1 | 12 | 0.952 | 0.343 | 0.312 |
| 50 | 0 | 0.2 | 1 | 12 | 0.983 | 0.431 | 0.372 |
| 60 | 0 | 0.2 | 1 | 12 | 0.994 | 0.474 | 0.417 |
| 70 | 0 | 0.2 | 1 | 12 | 0.999 | 0.546 | 0.490 |
| 80 | 0 | 0.2 | 1 | 12 | 1.000 | 0.622 | 0.546 |
| 90 | 0 | 0.2 | 1 | 12 | 1.000 | 0.690 | 0.615 |
| 100 | 0 | 0.2 | 1 | 12 | 1.000 | 0.724 | 0.647 |

```r
simulation_summary |>
  arrange(skew, n_per_group) |>
  select(n_per_group, skew, power_t, power_u, power_conditional, conditional_better_than_t) |>
  mutate(power_diff_cond_t = power_conditional - power_t) |>
  filter(conditional_better_than_t == TRUE &
           power_diff_cond_t >= 0.05) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n_per_group | skew | power_t | power_u | power_conditional | conditional_better_than_t | power_diff_cond_t |
|---|---|---|---|---|---|---|
| 50 | 6 | 0.335 | 0.384 | 0.386 | TRUE | 0.051 |
| 90 | 6 | 0.585 | 0.642 | 0.642 | TRUE | 0.057 |
| 100 | 6 | 0.623 | 0.686 | 0.686 | TRUE | 0.063 |
| 60 | 9 | 0.431 | 0.496 | 0.496 | TRUE | 0.065 |
| 70 | 9 | 0.483 | 0.568 | 0.568 | TRUE | 0.085 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 80 | 9 | 0.556 | 0.627 | 0.627 | TRUE | 0.071 |
| 90 | 9 | 0.595 | 0.652 | 0.652 | TRUE | 0.057 |
| 100 | 9 | 0.649 | 0.728 | 0.728 | TRUE | 0.079 |
| 50 | 12 | 0.372 | 0.431 | 0.430 | TRUE | 0.058 |
| 60 | 12 | 0.417 | 0.474 | 0.474 | TRUE | 0.057 |
| 70 | 12 | 0.490 | 0.546 | 0.546 | TRUE | 0.056 |
| 80 | 12 | 0.546 | 0.622 | 0.622 | TRUE | 0.076 |
| 90 | 12 | 0.615 | 0.690 | 0.690 | TRUE | 0.075 |
| 100 | 12 | 0.647 | 0.724 | 0.724 | TRUE | 0.077 |

```r
simulation_summary |>
  arrange(skew, n_per_group) |>
  select(n_per_group, skew, power_t, power_u, power_conditional, conditional_better_than_u) |>
  mutate(power_diff_cond_u = power_conditional - power_u) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n_per_group | skew | power_t | power_u | power_conditional | conditional_better_than_u | power_diff_cond_u |
|---|---|---|---|---|---|---|
| 10 | 0 | 0.072 | 0.062 | 0.074 | TRUE | 0.012 |
| 20 | 0 | 0.095 | 0.090 | 0.099 | TRUE | 0.009 |
| 30 | 0 | 0.136 | 0.124 | 0.137 | TRUE | 0.013 |
| 40 | 0 | 0.146 | 0.140 | 0.144 | TRUE | 0.004 |
| 50 | 0 | 0.177 | 0.174 | 0.179 | TRUE | 0.005 |
| 60 | 0 | 0.190 | 0.180 | 0.188 | TRUE | 0.008 |
| 70 | 0 | 0.191 | 0.191 | 0.196 | TRUE | 0.005 |
| 80 | 0 | 0.224 | 0.207 | 0.221 | TRUE | 0.014 |
| 90 | 0 | 0.270 | 0.254 | 0.268 | TRUE | 0.014 |
| 100 | 0 | 0.286 | 0.277 | 0.284 | TRUE | 0.007 |
| 10 | 1 | 0.079 | 0.069 | 0.078 | TRUE | 0.009 |
| 20 | 1 | 0.107 | 0.105 | 0.114 | TRUE | 0.009 |
| 30 | 1 | 0.151 | 0.144 | 0.150 | TRUE | 0.006 |
| 40 | 1 | 0.163 | 0.165 | 0.167 | TRUE | 0.002 |
| 50 | 1 | 0.229 | 0.210 | 0.222 | TRUE | 0.012 |
| 60 | 1 | 0.247 | 0.241 | 0.249 | TRUE | 0.008 |
| 70 | 1 | 0.290 | 0.277 | 0.288 | TRUE | 0.011 |
| 80 | 1 | 0.337 | 0.327 | 0.338 | TRUE | 0.011 |
| 90 | 1 | 0.352 | 0.347 | 0.357 | TRUE | 0.010 |
| 100 | 1 | 0.398 | 0.390 | 0.402 | TRUE | 0.012 |
| 10 | 2 | 0.098 | 0.093 | 0.105 | TRUE | 0.012 |
| 20 | 2 | 0.141 | 0.137 | 0.147 | TRUE | 0.010 |
| 30 | 2 | 0.211 | 0.211 | 0.220 | TRUE | 0.009 |
| 40 | 2 | 0.261 | 0.261 | 0.276 | TRUE | 0.015 |
| 50 | 2 | 0.283 | 0.283 | 0.285 | TRUE | 0.002 |
| 60 | 2 | 0.348 | 0.343 | 0.353 | TRUE | 0.010 |
| 70 | 2 | 0.400 | 0.389 | 0.405 | TRUE | 0.016 |
| 80 | 2 | 0.427 | 0.431 | 0.437 | TRUE | 0.006 |
| 90 | 2 | 0.490 | 0.485 | 0.490 | TRUE | 0.005 |
| 100 | 2 | 0.509 | 0.507 | 0.512 | TRUE | 0.005 |
| 10 | 3 | 0.090 | 0.088 | 0.098 | TRUE | 0.010 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 20 | 3 | 0.148 | 0.147 | 0.161 | TRUE | 0.014 |
| 30 | 3 | 0.222 | 0.216 | 0.229 | TRUE | 0.013 |
| 40 | 3 | 0.270 | 0.275 | 0.285 | TRUE | 0.010 |
| 50 | 3 | 0.326 | 0.330 | 0.340 | TRUE | 0.010 |
| 60 | 3 | 0.377 | 0.396 | 0.400 | TRUE | 0.004 |
| 70 | 3 | 0.432 | 0.451 | 0.457 | TRUE | 0.006 |
| 80 | 3 | 0.461 | 0.495 | 0.498 | TRUE | 0.003 |
| 90 | 3 | 0.568 | 0.592 | 0.590 | FALSE | -0.002 |
| 100 | 3 | 0.581 | 0.603 | 0.606 | TRUE | 0.003 |
| 10 | 6 | 0.102 | 0.104 | 0.114 | TRUE | 0.010 |
| 20 | 6 | 0.167 | 0.175 | 0.187 | TRUE | 0.012 |
| 30 | 6 | 0.229 | 0.259 | 0.268 | TRUE | 0.009 |
| 40 | 6 | 0.303 | 0.342 | 0.344 | TRUE | 0.002 |
| 50 | 6 | 0.335 | 0.384 | 0.386 | TRUE | 0.002 |
| 60 | 6 | 0.425 | 0.473 | 0.475 | TRUE | 0.002 |
| 70 | 6 | 0.464 | 0.505 | 0.505 | FALSE | 0.000 |
| 80 | 6 | 0.542 | 0.588 | 0.589 | TRUE | 0.001 |
| 90 | 6 | 0.585 | 0.642 | 0.642 | FALSE | 0.000 |
| 100 | 6 | 0.623 | 0.686 | 0.686 | FALSE | 0.000 |
| 10 | 9 | 0.093 | 0.084 | 0.100 | TRUE | 0.016 |
| 20 | 9 | 0.188 | 0.210 | 0.211 | TRUE | 0.001 |
| 30 | 9 | 0.224 | 0.260 | 0.270 | TRUE | 0.010 |
| 40 | 9 | 0.312 | 0.343 | 0.345 | TRUE | 0.002 |
| 50 | 9 | 0.399 | 0.442 | 0.444 | TRUE | 0.002 |
| 60 | 9 | 0.431 | 0.496 | 0.496 | FALSE | 0.000 |
| 70 | 9 | 0.483 | 0.568 | 0.568 | FALSE | 0.000 |
| 80 | 9 | 0.556 | 0.627 | 0.627 | FALSE | 0.000 |
| 90 | 9 | 0.595 | 0.652 | 0.652 | FALSE | 0.000 |
| 100 | 9 | 0.649 | 0.728 | 0.728 | FALSE | 0.000 |
| 10 | 12 | 0.112 | 0.109 | 0.117 | TRUE | 0.008 |
| 20 | 12 | 0.189 | 0.211 | 0.219 | TRUE | 0.008 |
| 30 | 12 | 0.270 | 0.309 | 0.313 | TRUE | 0.004 |
| 40 | 12 | 0.312 | 0.343 | 0.345 | TRUE | 0.002 |
| 50 | 12 | 0.372 | 0.431 | 0.430 | FALSE | -0.001 |
| 60 | 12 | 0.417 | 0.474 | 0.474 | FALSE | 0.000 |
| 70 | 12 | 0.490 | 0.546 | 0.546 | FALSE | 0.000 |
| 80 | 12 | 0.546 | 0.622 | 0.622 | FALSE | 0.000 |
| 90 | 12 | 0.615 | 0.690 | 0.690 | FALSE | 0.000 |
| 100 | 12 | 0.647 | 0.724 | 0.724 | FALSE | 0.000 |

```r
simulation_summary |>
  arrange(skew, n_per_group) |>
  select(n_per_group, skew, power_t, power_u, power_conditional, conditional_better_than_t) |>
  mutate(power_diff_u_t = power_u - power_t) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| n_per_group | skew | power_t | power_u | power_conditional | conditional_better_than_t | power_diff_u_t |
|---|---|---|---|---|---|---|
| 10 | 0 | 0.072 | 0.062 | 0.074 | TRUE | -0.010 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 20 | 0 | 0.095 | 0.090 | 0.099 | TRUE | -0.005 |
| 30 | 0 | 0.136 | 0.124 | 0.137 | TRUE | -0.012 |
| 40 | 0 | 0.146 | 0.140 | 0.144 | FALSE | -0.006 |
| 50 | 0 | 0.177 | 0.174 | 0.179 | TRUE | -0.003 |
| 60 | 0 | 0.190 | 0.180 | 0.188 | FALSE | -0.010 |
| 70 | 0 | 0.191 | 0.191 | 0.196 | TRUE | 0.000 |
| 80 | 0 | 0.224 | 0.207 | 0.221 | FALSE | -0.017 |
| 90 | 0 | 0.270 | 0.254 | 0.268 | FALSE | -0.016 |
| 100 | 0 | 0.286 | 0.277 | 0.284 | FALSE | -0.009 |
| 10 | 1 | 0.079 | 0.069 | 0.078 | FALSE | -0.010 |
| 20 | 1 | 0.107 | 0.105 | 0.114 | TRUE | -0.002 |
| 30 | 1 | 0.151 | 0.144 | 0.150 | FALSE | -0.007 |
| 40 | 1 | 0.163 | 0.165 | 0.167 | TRUE | 0.002 |
| 50 | 1 | 0.229 | 0.210 | 0.222 | FALSE | -0.019 |
| 60 | 1 | 0.247 | 0.241 | 0.249 | TRUE | -0.006 |
| 70 | 1 | 0.290 | 0.277 | 0.288 | FALSE | -0.013 |
| 80 | 1 | 0.337 | 0.327 | 0.338 | TRUE | -0.010 |
| 90 | 1 | 0.352 | 0.347 | 0.357 | TRUE | -0.005 |
| 100 | 1 | 0.398 | 0.390 | 0.402 | TRUE | -0.008 |
| 10 | 2 | 0.098 | 0.093 | 0.105 | TRUE | -0.005 |
| 20 | 2 | 0.141 | 0.137 | 0.147 | TRUE | -0.004 |
| 30 | 2 | 0.211 | 0.211 | 0.220 | TRUE | 0.000 |
| 40 | 2 | 0.261 | 0.261 | 0.276 | TRUE | 0.000 |
| 50 | 2 | 0.283 | 0.283 | 0.285 | TRUE | 0.000 |
| 60 | 2 | 0.348 | 0.343 | 0.353 | TRUE | -0.005 |
| 70 | 2 | 0.400 | 0.389 | 0.405 | TRUE | -0.011 |
| 80 | 2 | 0.427 | 0.431 | 0.437 | TRUE | 0.004 |
| 90 | 2 | 0.490 | 0.485 | 0.490 | FALSE | -0.005 |
| 100 | 2 | 0.509 | 0.507 | 0.512 | TRUE | -0.002 |
| 10 | 3 | 0.090 | 0.088 | 0.098 | TRUE | -0.002 |
| 20 | 3 | 0.148 | 0.147 | 0.161 | TRUE | -0.001 |
| 30 | 3 | 0.222 | 0.216 | 0.229 | TRUE | -0.006 |
| 40 | 3 | 0.270 | 0.275 | 0.285 | TRUE | 0.005 |
| 50 | 3 | 0.326 | 0.330 | 0.340 | TRUE | 0.004 |
| 60 | 3 | 0.377 | 0.396 | 0.400 | TRUE | 0.019 |
| 70 | 3 | 0.432 | 0.451 | 0.457 | TRUE | 0.019 |
| 80 | 3 | 0.461 | 0.495 | 0.498 | TRUE | 0.034 |
| 90 | 3 | 0.568 | 0.592 | 0.590 | TRUE | 0.024 |
| 100 | 3 | 0.581 | 0.603 | 0.606 | TRUE | 0.022 |
| 10 | 6 | 0.102 | 0.104 | 0.114 | TRUE | 0.002 |
| 20 | 6 | 0.167 | 0.175 | 0.187 | TRUE | 0.008 |
| 30 | 6 | 0.229 | 0.259 | 0.268 | TRUE | 0.030 |
| 40 | 6 | 0.303 | 0.342 | 0.344 | TRUE | 0.039 |
| 50 | 6 | 0.335 | 0.384 | 0.386 | TRUE | 0.049 |
| 60 | 6 | 0.425 | 0.473 | 0.475 | TRUE | 0.048 |
| 70 | 6 | 0.464 | 0.505 | 0.505 | TRUE | 0.041 |
| 80 | 6 | 0.542 | 0.588 | 0.589 | TRUE | 0.046 |
| 90 | 6 | 0.585 | 0.642 | 0.642 | TRUE | 0.057 |
| 100 | 6 | 0.623 | 0.686 | 0.686 | TRUE | 0.063 |
| 10 | 9 | 0.093 | 0.084 | 0.100 | TRUE | -0.009 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 20 | 9 | 0.188 | 0.210 | 0.211 | TRUE | 0.022 |
| 30 | 9 | 0.224 | 0.260 | 0.270 | TRUE | 0.036 |
| 40 | 9 | 0.312 | 0.343 | 0.345 | TRUE | 0.031 |
| 50 | 9 | 0.399 | 0.442 | 0.444 | TRUE | 0.043 |
| 60 | 9 | 0.431 | 0.496 | 0.496 | TRUE | 0.065 |
| 70 | 9 | 0.483 | 0.568 | 0.568 | TRUE | 0.085 |
| 80 | 9 | 0.556 | 0.627 | 0.627 | TRUE | 0.071 |
| 90 | 9 | 0.595 | 0.652 | 0.652 | TRUE | 0.057 |
| 100 | 9 | 0.649 | 0.728 | 0.728 | TRUE | 0.079 |
| 10 | 12 | 0.112 | 0.109 | 0.117 | TRUE | -0.003 |
| 20 | 12 | 0.189 | 0.211 | 0.219 | TRUE | 0.022 |
| 30 | 12 | 0.270 | 0.309 | 0.313 | TRUE | 0.039 |
| 40 | 12 | 0.312 | 0.343 | 0.345 | TRUE | 0.031 |
| 50 | 12 | 0.372 | 0.431 | 0.430 | TRUE | 0.059 |
| 60 | 12 | 0.417 | 0.474 | 0.474 | TRUE | 0.057 |
| 70 | 12 | 0.490 | 0.546 | 0.546 | TRUE | 0.056 |
| 80 | 12 | 0.546 | 0.622 | 0.622 | TRUE | 0.076 |
| 90 | 12 | 0.615 | 0.690 | 0.690 | TRUE | 0.075 |
| 100 | 12 | 0.647 | 0.724 | 0.724 | TRUE | 0.077 |

```r
simulation_summary |>
  summarize(percent_u_better_than_t = mean(u_better_than_t)*100,
            percent_conditional_better_than_t = mean(conditional_better_than_t)*100,
            percent_conditional_better_than_u = mean(conditional_better_than_u)*100) |>
  mutate_if(is.numeric, janitor::round_half_up, digits = 1) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| percent_u_better_than_t | percent_conditional_better_than_t | percent_conditional_better_than_u |
|---|---|---|
| 52.9 | 85.7 | 78.6 |

```r
simulation_summary |>
  summarize(percent_u_much_better_than_t = mean(u_much_better_than_t)*100,
            percent_conditional_much_better_than_t = mean(conditional_much_better_than_t)*100,
            percent_conditional_much_better_than_u = mean(conditional_much_better_than_u)*100) |>
  mutate_if(is.numeric, janitor::round_half_up, digits = 1) |>
  kable() |>
  kable_classic(full_width = FALSE)
```

| percent_u_much_better_than_t | percent_conditional_much_better_than_t | percent_conditional_much_better_tha |
|---|---|---|
| 18.6 | 20 | |

## 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] kableExtra_1.4.0 knitr_1.46       sn_2.1.1         ggplot2_3.5.1
## [5] purrr_1.0.2      readr_2.1.5      forcats_1.0.0    dplyr_1.1.4
## [9] tidyr_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] utf8_1.2.4         generics_0.1.3     xml2_1.3.6
##  [4] stringi_1.8.3      hms_1.1.3          digest_0.6.35
##  [7] magrittr_2.0.3     evaluate_0.23      grid_4.3.2
## [10] timechange_0.3.0   fastmap_1.1.1      fansi_1.0.6
## [13] viridisLite_0.4.2  scales_1.3.0       numDeriv_2016.8-1.1
## [16] mnormt_2.1.1       cli_3.6.2          rlang_1.1.3
## [19] munsell_0.5.1      withr_3.0.0        yaml_2.3.8
## [22] tools_4.3.2        tzdb_0.4.0         colorspace_2.1-0
## [25] vctrs_0.6.5        R6_2.5.1           lifecycle_1.0.4
## [28] lubridate_1.9.3    snakecase_0.11.1   stringr_1.5.1
## [31] janitor_2.2.0      pkgconfig_2.0.3    pillar_1.9.0
## [34] gtable_0.3.5       glue_1.7.0         systemfonts_1.0.6
## [37] xfun_0.43          tibble_3.2.1       tidyselect_1.2.1
## [40] highr_0.10         rstudioapi_0.16.0  htmltools_0.5.8.1
## [43] rmarkdown_2.26     svglite_2.1.3      compiler_4.3.2
```