UNIVERSITY OF LEICESTER

Analysing Results From Monte Carlo Simulation Studies

A Gasparini (@ellessenne) · TP Morris · IR White · MJ Crowther

useR! 2019

Monte Carlo simulation studies

Monte Carlo simulation studies are computer experiments that involve generating data by pseudo-random sampling.

Rationale

In a simulation study, we:

1. Generate data from a known distribution (so that we know the “truth”);
2. Analyse the data;
3. Compare the analysis results with the truth.

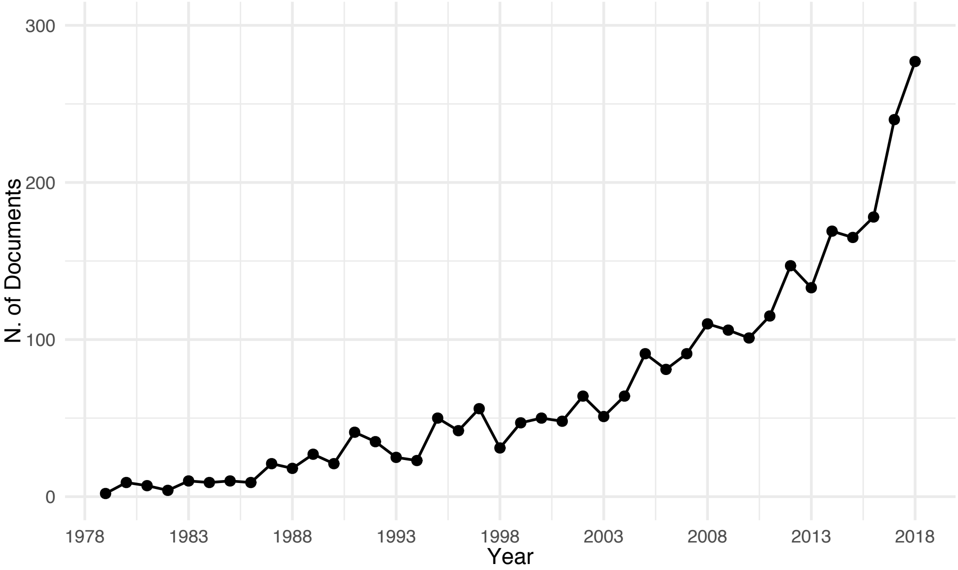
Monte Carlo simulation studies are useful...

Monte Carlo simulation studies provide an invaluable tool for statistical and biostatistical research.

They can also help to answer questions such as:

- Is an estimator biased in a finite sample?
- Do confidence intervals for a given parameter achieve the desired nominal level of coverage?
- How does a newly developed method compare to an established one?
- What is the power to detect a desired effect size under complex experimental settings and analysis methods?
- You name it!

…and increasingly popular!



Scopus search key: `TITLE-ABS-KEY ("Monte Carlo simulation study")`

There are issues...

1. Monte Carlo standard errors of performance measures are often not computed/reported;

There are issues...

1. Monte Carlo standard errors of performance measures are often not computed/reported;
2. Reproducibility of results;

There are issues...

1. Monte Carlo standard errors of performance measures are often not computed/reported;
2. Reproducibility of results;
3. Dissemination of results.

Disclaimer!

I will not cover how to plan, design, and run a Monte Carlo simulation study.

An full example on modelling survival data is included with `rsimsum`:

```
vignette(”relhaz”, package = ”rsimsum”)
```

Also, check out the tutorial paper by Morris *et al*. (2019).

Example: t-test (1)

Aim: investigate the performance of the two-sample t test when

1. Data is skewed;
2. Variances are unequal in the two groups.

Example: t-test (2)

Data-generating mechanisms (DGMs):

1. Simulating 50 observations, with 2:1 groups ratio;
2. Fully fractional design, varying equal/unequal variance, and skewed/non-skewed data.

Example: t-test (2)

Data-generating mechanisms (DGMs):

1. Simulating 50 observations, with 2:1 groups ratio;
2. Fully fractional design, varying equal/unequal variance, and skewed/non-skewed data.

Methods:

1. t-test with pooled variance;
2. t-test with unpooled variance.

Example: t-test (2)

Data-generating mechanisms (DGMs):

1. Simulating 50 observations, with 2:1 groups ratio;
2. Fully fractional design, varying equal/unequal variance, and skewed/non-skewed data.

Methods:

1. t-test with pooled variance;
2. t-test with unpooled variance.

Replications: 2,000 per DGM.

Case study: data

```
dplyr::glimpse(data)

## Observations: 16,000
## Variables: 8
## $ diff <dbl> -3.56987311, -1.37912667, -3.56987311, -1.37912667, -0.45626...
## $ se <dbl> 1.440554, 1.973592, 1.405570, 2.434108, 1.474513, 2.185057, ...
## $ df <dbl> 48.00000, 48.00000, 35.55937, 21.15939, 48.00000, 48.00000, ...
## $ i <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, ...
## $ dgm <fct> "Equal, Non-skewed", "Unequal, Non-Skewed", "Equal, Non-skew...
## $ method <fct> "t-test (P)", "t-test (P)", "t-test, (NP)", "t-test, (NP)",
...
## $ dist <fct> N, N, N, N, Gamma, Gamma, Gamma, Gamma, N, N, N, N, Gamma, G...
## $ var <fct> Equal, Unequal, Equal, Unequal, Equal, Unequal, Equal, Unequ...
```

rsimsum's main function: `simsum`

```r
args(rsimsum::simsum)
```

```
## function (data, estvarname, true, se, methodvar = NULL, ref = NULL,
##     by = NULL, ci.limits = NULL, dropbig = FALSE, x = FALSE,
##     control = list())
## NULL
```

Documentation: `https://ellessenne.github.io/rsimsum/`

Summarising a simulation study (1)

```
s <- rsimsum::simsum(
  data = data, estvarname = "diff", se = "se", true = -1, methodvar = "method",
  by = "dgm", ref = "t-test, (NP)", x = TRUE
)
s

## Summary of a simulation study with a single estimand.
##
## Method variable: method
##  Unique methods: t-test, (NP), t-test (P)
##  Reference method: t-test, (NP)
##
## By factors: dgm
##
## Monte Carlo standard errors were computed.
```

Summarising a simulation study (2)

```
summary(s, stats = "bias")

## Values are:
##  Point Estimate (Monte Carlo Standard Error)
##
## Bias in point estimate:
##                  dgm     t-test, (NP)        t-test (P)
##     Equal, Non-skewed  0.0048 (0.0335)  0.0048 (0.0335)
##   Unequal, Non-Skewed  0.0288 (0.0591)  0.0288 (0.0591)
##         Equal, Skewed -0.0474 (0.0334) -0.0474 (0.0334)
##       Unequal, Skewed  0.0219 (0.0583)  0.0219 (0.0583)
```

Summarising a simulation study (3)

```
summary(s, stats = "cover")

## Values are:
##  Point Estimate (Monte Carlo Standard Error)
##
## Coverage of nominal 95% confidence interval:
##                      dgm    t-test, (NP)       t-test (P)
##     Equal, Non-skewed 0.9485 (0.0049) 0.9500 (0.0049)
##   Unequal, Non-Skewed 0.9465 (0.0050) 0.8825 (0.0072)
##         Equal, Skewed 0.9535 (0.0047) 0.9570 (0.0045)
##       Unequal, Skewed 0.9200 (0.0061) 0.8775 (0.0073)
```

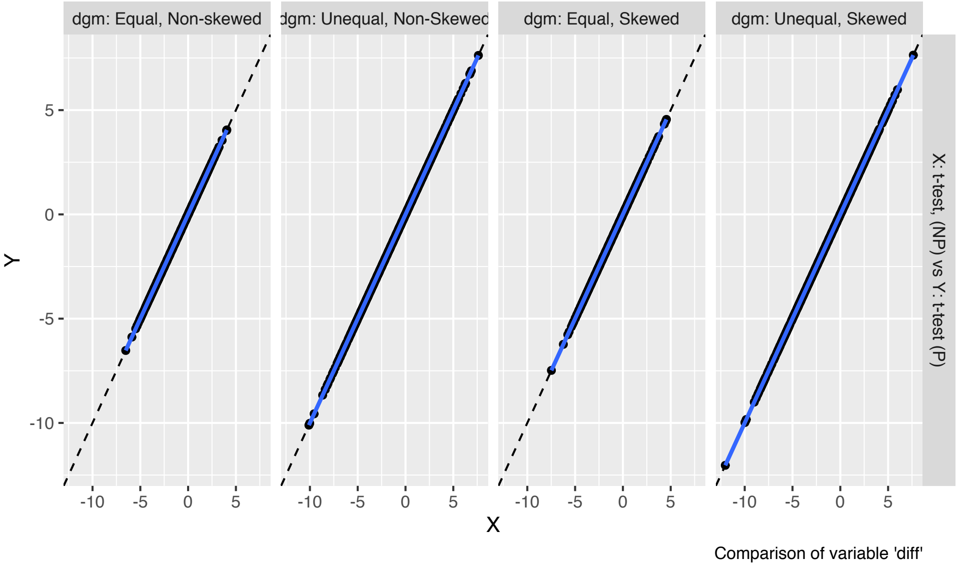
Performance measures implemented in `rsimsum`

The following performance measures are implemented in `rsimsum`:

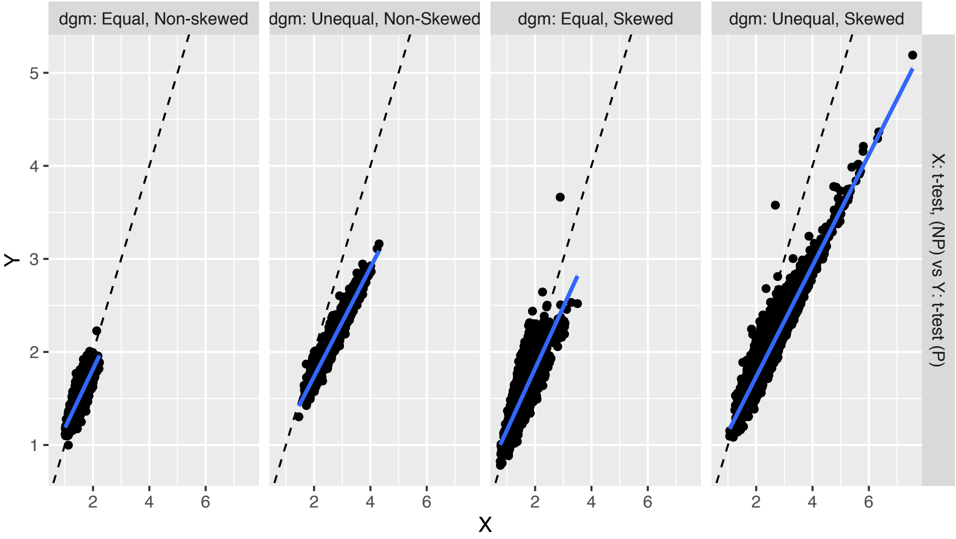
- Bias;
- Empirical SE, relative % increase in precision, model-based SE, and relative % error in model-based SE;
- Mean squared error (MSE);
- Coverage probability and bias-corrected coverage probability;
- Power of type I error.

Each performance measure is described in more detail elsewhere (Morris *et al.*, 2019).

```
autoplot(object = s, type = "est")
```



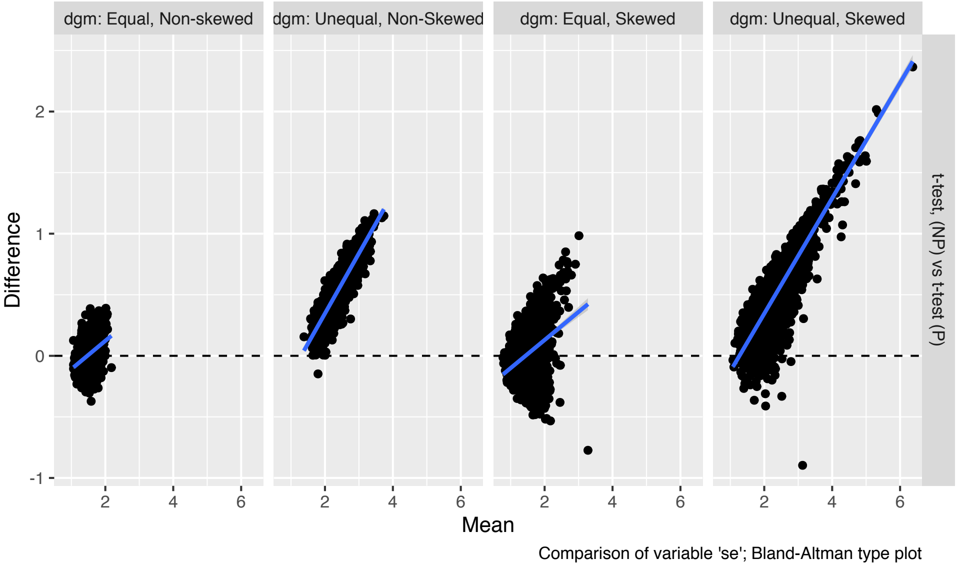
```
autoplot(object = s, type = "se")
```



Comparison of variable 'se'

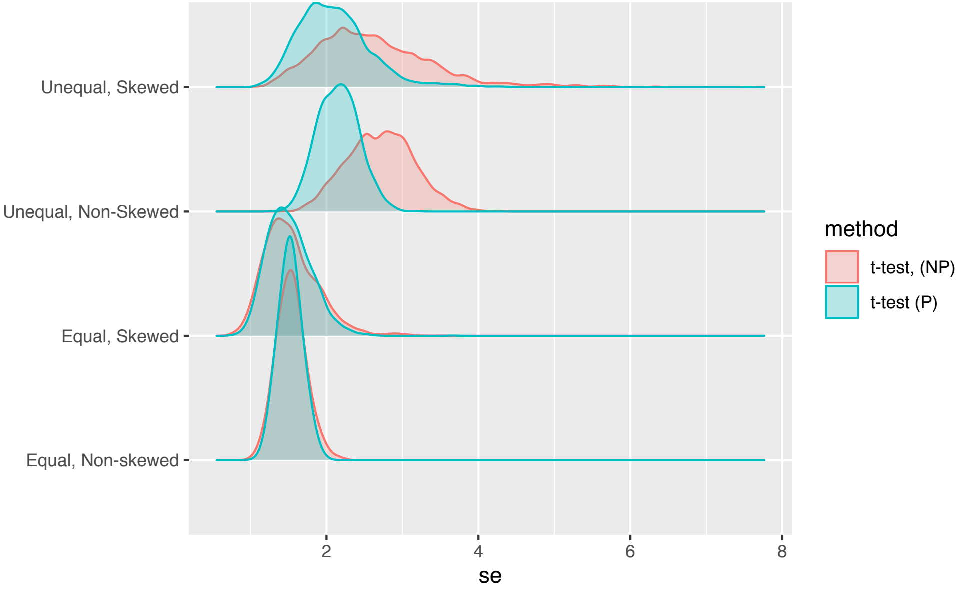
Plotting standard errors

```
autoplot(object = s, type = "se_ba")
```



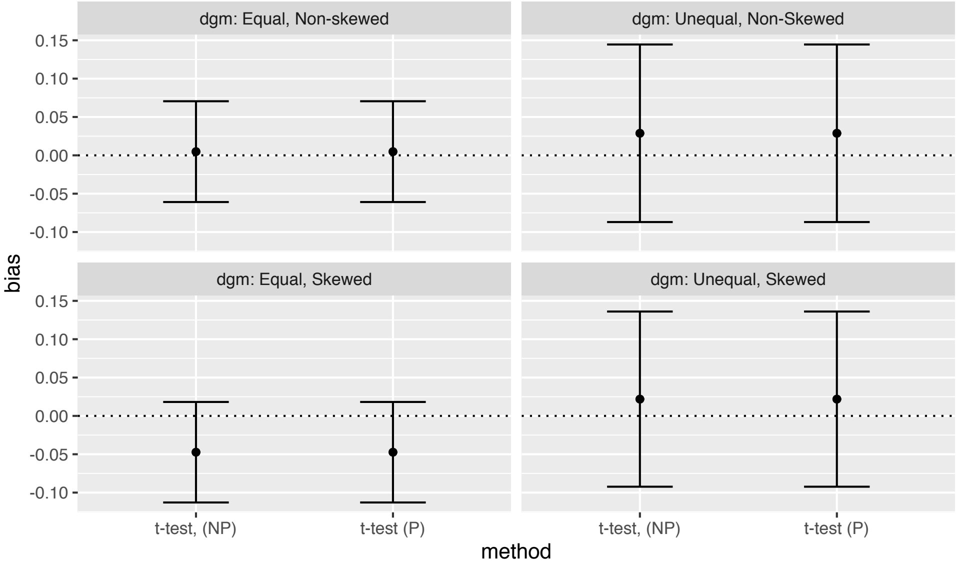
Plotting standard errors

```
autoplot(object = s, type = "se_ridge")
```

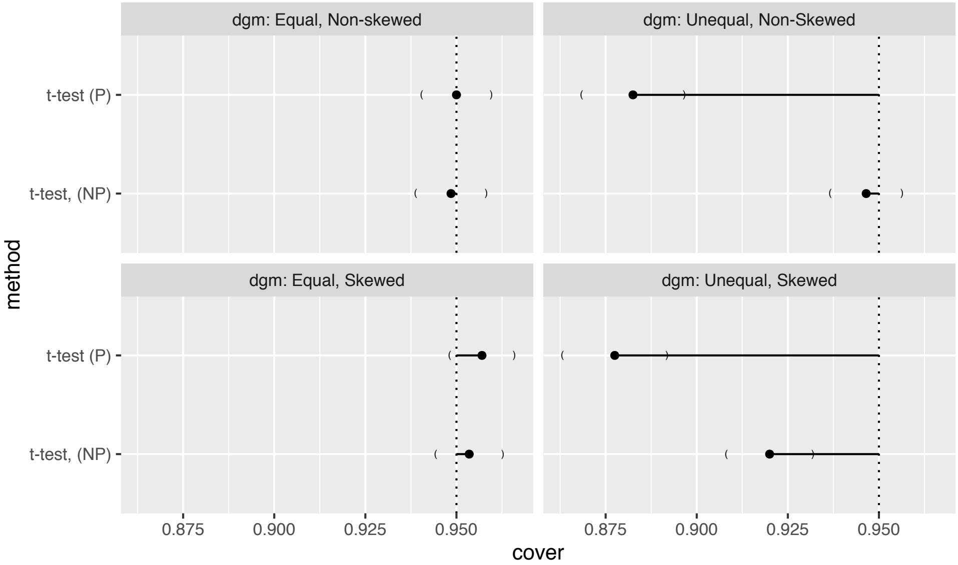


Plotting results

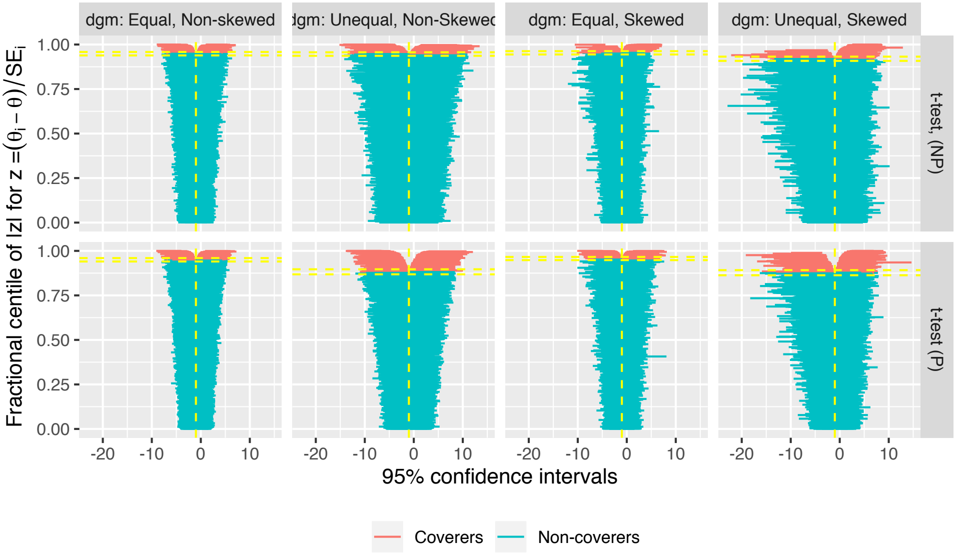
```
autoplot(object = summary(s), type = "forest", stats = "bias")
```



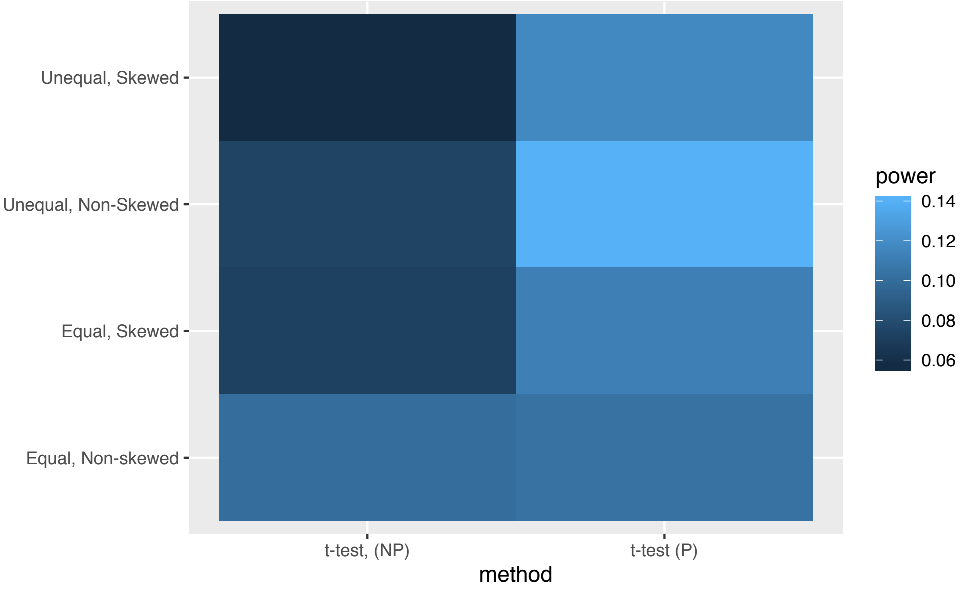
```
autoplot(object = summary(s), type = "lolly", stats = "cover")
```



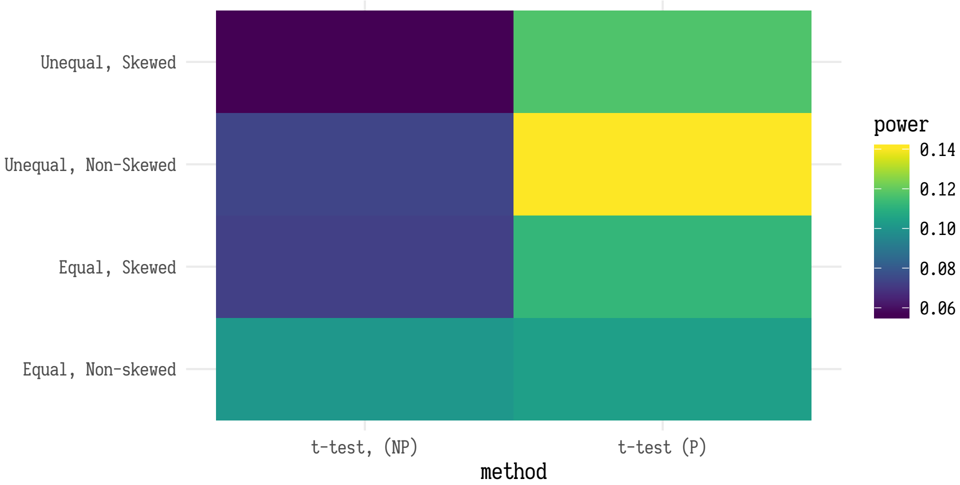
```
autoplot(object = s, type = "zip")
```



```
autoplot(object = s, type = "heat", stats = "power")
```



```
autoplot(object = s, type = "heat", stats = "power") +
  viridis::scale_fill_viridis() +
  ggplot2::theme_minimal(base_family = "Iosevka Slab")
```



Multiple estimands

```
args(rsimsum::multisimsum)
```

```
## function (data, par, estvarname, true, se, methodvar = NULL,
##     ref = NULL, by = NULL, ci.limits = NULL, dropbig = FALSE,
##     x = FALSE, control = list())
## NULL
```

INteractive **T**ool for **E**xploring **RE**sults from **S**imulation s**T**udies

Why a Shiny app?

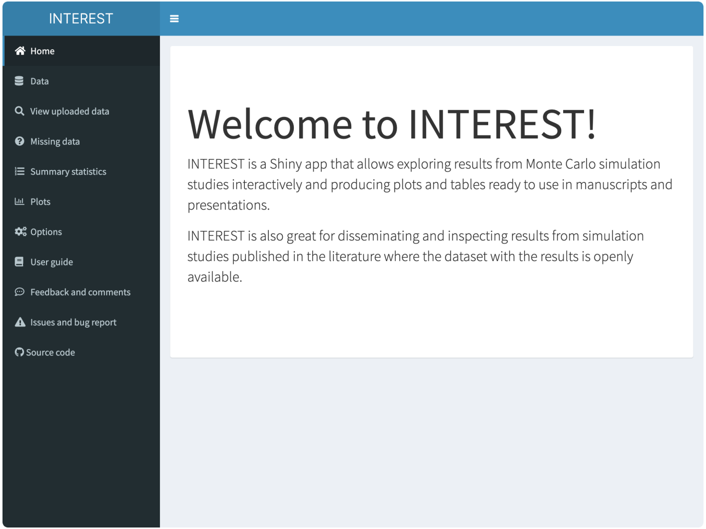
Why a Shiny app?

- Dissemination of results and open science;
- Fast iteration and exploration of results;
- Supporting devices where R does not run natively (smartphones, Chromebooks, ...).

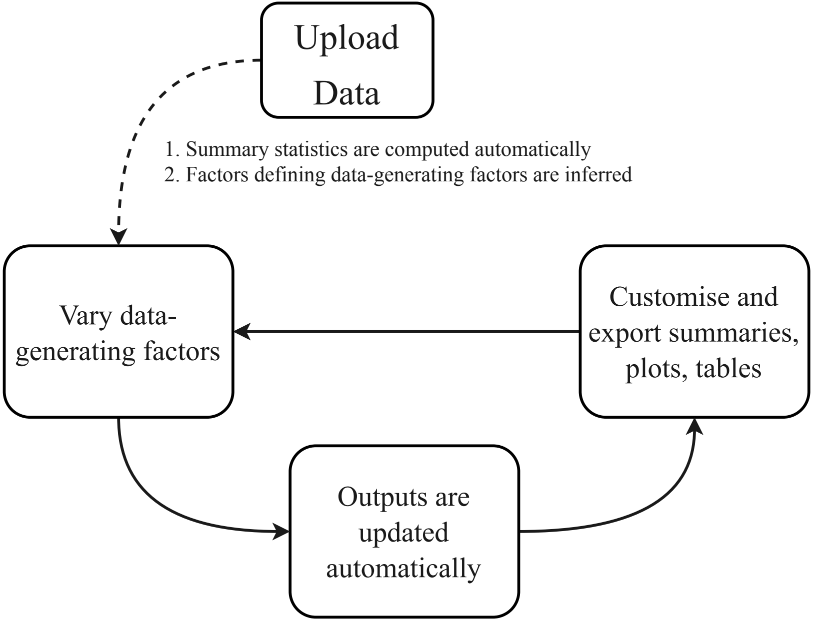
Dissemination is key

1. Can drive practitioners and applied statisticians to methods that have been shown to perform well in their practical settings;
2. Can guide researchers to develop new methods in promising directions;
3. Can provide insights into less established methods.

INTEREST: Homepage



INTEREST: Workflow



http://interest.shinyapps.io/interest/

How to get rsimsum and INTEREST?

rsimsum can be installed directly from CRAN:

```
install.packages('rsimsum')
# Development version on GitHub:
# require('remotes')
# remotes::install_github(repo = 'ellessenne/rsimsum')
```

INTEREST is on GitHub:

```
# require('remotes')
remotes::install_github(repo = 'ellessenne/interest')
```

What is coming next?

- Nested loop plot for simulation studies with several DGMs (Rücker and Schwarzer, 2014);
- Methods to easily reproduce plots generated by `autoplot`;
- Methods to directly export (pretty) LaTeX tables;
- Additional exporting tools for INTEREST;
- Support for simulation studies with multiple estimands in INTEREST;
- ...

Thank you!

References:

- *Using simulation studies to evaluate statistical methods*. Morris TP, White IR, and Crowther MJ (2019). Statistics in Medicine 38(11):2074–2102, DOI: 10.1002/sim.8086
- *rsimsum: Summarise results from Monte Carlo simulation studies*. Gasparini A (2018). Journal of Open Source Software 3(26):739, DOI: 10.21105/joss.00739
- `rsimsum`'s website: `https://ellessenne.github.io/rsimsum/`

Slides available online from add GitHub (short) link here!