

# Best practices for using simulation studies to evaluate statistical methods

---

Julia Wrobel

# Overview

Today, we cover:

- Simulation studies
  - Design
  - Implementation
  - Presentation
- Lab

Announcements

- HW2 posted and due 2/4 at 10:00AM

# Motivation

- In statistics, simulations are typically used to establish **properties** of statistical methods, particularly when it is difficult or impossible to derive **exact analytic expressions**:
  - Is my estimator biased in finite samples?
  - Does my confidence interval achieve nominal coverage?
  - How powerful is my test under different alternatives to the null hypothesis?
  - How does my method compare to a competing method?

# Motivation

- Exact or approximate analytic results may require assumptions
  - e.g. normality
  - Simulations can be used to study behavior when these assumptions are violated

# Monte Carlo Simulation

- The questions on the previous slides can be answered using **Monte Carlo simulation**
- A Monte Carlo simulation is a computer experiment involving (pseudo-)random sampling
- Simulation is used for:
  - Simulation studies (focus of this lecture)
  - Multiple imputation methods
  - Markov Chain Monte Carlo methods (focus of later lectures)

# Monte Carlo History

- New supercomputer (ENIAC) inspired former Manhattan Project scientists Stan Ulam and John von Neumann to try a statistical approach to solving an intractable system of equations related to creation of an atomic bomb

## JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

*Number 247*

---

SEPTEMBER 1949

---

*Volume 44*

### THE MONTE CARLO METHOD

NICHOLAS METROPOLIS AND S. ULAM

*Los Alamos Laboratory*

# Monte Carlo History

- Ulam was playing solitaire and wondered: what are the chances that a hand laid out with 52 cards will come out successfully?
  - $\sim 8 \times 10^{67}$  ways to sort a deck of cards
    - Intractable analytically
- What if instead he could simply lay out the cards one hundred times and count the number of successful plays?

# Simulation studies in biostatistics

Many uses!

- Check algebra (and code) to provide reassurance no error has been made when a new statistical method has been derived analytically
- Assess relevance of large-sample theory in finite samples
- For absolute evaluation of a new method
  - Does it work well in the scenarios for which it was designed?
- For comparative evaluation of two or more methods



# Simulation Study Design

- Simulation studies are **empirical experiments**
- As statisticians, we should apply principles of **experimental design** when conducting simulation studies
- Burton, Altman, Royston, and Holder (2006): Before writing code, develop a **protocol** that provides details about how the study will be performed, analysed, and reported

# Simulation Study Design

- Morris, White, and Crowther (2019) advocate for **ADEMP**:
  - Aims
  - Data-generating Mechanisms
  - Estimands
  - Methods
  - Performance Measures

They also look at 100 papers published in Statistics in Medicine and critique the design of the simulation studies.

# ADEMP Structure

Aims	3.1
· Identify <i>specific</i> aims of simulation study.	
Data-generating mechanisms	3.2
· In relation to the aims, decide whether to use resampling or simulation from some parametric model.	
· For simulation from a parametric model, decide how simple or complex the model should be and whether it should be based on real data.	
· Determine what factors to vary and the levels of factors to use.	
· Decide whether factors should be varied fully factorially, partly factorially or one-at-a-time.	
Estimand/target of analysis	3.3
· Define estimands and/or other targets of the simulation study.	
Methods	3.4
· Identify methods to be evaluated and consider whether they are appropriate for estimand/target identified.	
For method comparison studies, make a careful review of the literature to ensure inclusion of relevant methods.	
Performance measures	3.5, 5.2
· List all performance measures to be estimated, justifying their relevance to estimands or other targets.	
· For less-used performance measures, give explicit formulae for the avoidance of ambiguity.	5.2
· Choose a value of $n_{\text{sim}}$ that achieves acceptable Monte Carlo SE for key performance measures.	5.2, 5.3

# Planning Stage (notation)

$\theta$	An estimand (conceptually); also true value of the estimand
$n_{\text{obs}}$	Sample size of a simulated dataset
$n_{\text{sim}}$	Number of repetitions used; the simulation sample size
$i = 1, \dots, n_{\text{sim}}$	Indexes the repetitions of the simulation
$\hat{\theta}$	the estimator of $\theta$
$\hat{\theta}_i$	the estimate of $\theta$ from the $i$ th repetition
$\bar{\theta}$	the mean of $\hat{\theta}_i$ across repetitions
$\text{Var}(\hat{\theta})$	the true variance of $\hat{\theta}$ , which can be estimated with large $n_{\text{sim}}$
$\widehat{\text{Var}}(\hat{\theta}_i)$	an estimate of $\text{Var}(\hat{\theta})$ from the $i$ th repetition
$\alpha$	the nominal significance level
$p_i$	the p-value returned by the $i$ th repetition

# Estimands

What is an estimand?

# Estimands

What is an estimand?

- An estimand is the precise quantity or parameter of interest that a study aims to estimate to address its research question.

# Estimands

What is an estimand?

- An estimand is the precise quantity or parameter of interest that a study aims to estimate to address its research question.

**Example:** treatment effect in a clinical trial

# Simple exam: linear regression

$$Y_i = \beta_0 + \beta_{treatment} X_{i1} + \mathbf{Z}_i^T \gamma + \epsilon_i$$

- $Y_i$ : continuous outcome
- $X_{i1}$ : treatment group indicator;  $X_{i1} = 1$  for treated
- $\mathbf{Z}_i$ : vector of potential confounders
- $\beta_{treatment}$ : average treatment effect, adjusting for  $\mathbf{Z}_i$
- $\gamma$ : vector of regression coefficient values for confounders
- $\epsilon_i$ : iid error,  $\sim N(0, \sigma^2)$



# Planning Stage: Aims

Desirable (asymptotic) properties of an estimator  $\hat{\theta}$  from a frequentist perspective:

1.  $\hat{\theta}$  should be consistent: as  $n \rightarrow \infty$ ,  $\hat{\theta} \rightarrow \theta$ 
  - Also desirable that  $\hat{\theta}$  is unbiased:  $E(\hat{\theta}) = \theta$
2. The sample estimate  $\widehat{Var}(\hat{\theta})$  consistent estimate of true sampling variance of  $\hat{\theta}$ ,  $Var(\hat{\theta})$
3. Confidence intervals should have **good coverage**: at least  $100(1 - \alpha)\%$  of intervals contain  $\theta$
4.  $\hat{\theta}$  should be efficient:  $Var(\hat{\theta})$  should be as small as possible

# Planning Stage: Aims

Desirable properties of an estimator  $\hat{\theta}$  from a frequentist perspective:

1.  $\hat{\theta}$  should be consistent: as  $n \rightarrow \infty$ ,  $\hat{\theta} \rightarrow \theta$ 
  - Also desirable that  $\hat{\theta}$  is unbiased:  $E(\hat{\theta}) = \theta$
2. The sample estimate  $\widehat{Var}(\hat{\theta})$  consistent estimate of true sampling variance of  $\hat{\theta}$ ,  $Var(\hat{\theta})$
3. Confidence intervals should have **good coverage**: at least  $100(1 - \alpha)\%$  of intervals contain  $\theta$
4.  $\hat{\theta}$  should be efficient:  $Var(\hat{\theta})$  should be as small as possible
  - Computational efficiency

# Planning Stage: Aims

Proof-of-concept vs. stretching simulation studies

- Proof-of-concept: aim to show that a method is viable in some settings
- Stretch/break objective: identify settings where the method may fail

Examples of each in the linear regression setting?

# Data-generating mechanisms

Refers to how random numbers are used to generate a simulated dataset.

1. Parametric draws from a known model
  - true data generating model is known
  - more flexible but may be overly simplistic for real data
2. Repeated resampling with replacement from a specific dataset
  - true data generating model is unknown
  - less flexible but relevant for at least the study at hand

# Data-generating mechanisms

If using a parametric model, need to fully specify your generative model:

- Functional form(s)
- True values of the parameters
- Error distributions and their parameters

Without every detail of the data generation process, others will not be able to reproduce your simulation results!

# Data-generating mechanisms

Suppose you want to calculate coverage for  $\hat{\beta}_{treatment}$  under several conditions:

- Sample sizes  $n \in \{50, 100, 200\}$
- Error variances  $\sigma^2 \in \{1, 3\}$
- True treatment effects  $\beta_{treatment} \in \{0, 0.5, 2\}$
- **Full factorial design:** evaluate all combinations of these conditions/factors
  - $3 \times 2 \times 3 = 18$  simulation scenarios
  - This is the preferred approach, but can be computationally demanding

# Estimands and other targets

- Most simulation studies evaluate or compare methods for estimating one or more population quantities, which we term **estimands** and denote by  $\theta$
- Choice of estimand(s) depends on the aims of your study!
  - If you are interested in how well treatment works:  $\beta_{treatment}$
  - If interested in the patient outcome value for a given set of characteristics:  $E(Y_i|X_i, \beta)$
- Not all simulation studies involve an estimand. These other quantities we might want to focus on are referred to as **targets** of a simulation study

# Estimands and other targets

Statistical Task	Target	Examples of Performance Measures
<i>Analysis</i>		
Estimation	Estimand	Bias, empirical SE, mean-squared error, coverage
Testing	Null hypothesis	Type I error rate, power
Model selection	Model	Correct model rate, sensitivity or specificity for covariate selection
Prediction	Prediction/s	Measures of predictive accuracy, calibration, discrimination
<i>Design</i>		
Design a study	Selected design	Sample size, expected sample size, power/precision



# Planning: Methods

What method is being evaluated? Is it being compared to other methods?

- When comparing across multiple methods, it is important to consider:
  - Are all relevant methods in the literature being included in your study?
    - If not, what is your justification?
  - Do competing methods have open source implementations?
    - Will you need to contact author(s) for code?
  - What are the assumptions of each method?
    - Will your simulation design favor one method over another?

# Planning: Performance measures

A **performance measure** is a numerical quantity used to assess the performance of a method. Examples of common measures:

- bias:  $E[\hat{\theta}] - \theta$ 
  - used for estimands
- coverage:  $Pr(\hat{\theta}_{low} \leq \theta \leq \hat{\theta}_{high})$ 
  - used for estimands
- power and type 1 error
  - used in evaluating a method that targets a hypothesis test

# Implementing the simulation study

## CODING AND EXECUTION

4

- Separate scripts used to analyze simulated datasets from scripts to analyze estimates datasets.
- Start small and build up code, including plenty of checks.
- Set the random number seed once per simulation repetition.
- Store the random number states at the start of each repetition.
- If running chunks of the simulation in parallel, use separate streams of random numbers.<sup>17</sup>

## ANALYSIS

5

- Conduct exploratory analysis of results, particularly graphical exploration.
- Compute estimates of performance and Monte Carlo SEs for these estimates.

5.2

## REPORTING

6

- Describe simulation study using ADEMP structure with sufficient rationale for choices.
- Structure graphical and tabular presentations to place performance of competing methods side-by-side.
- Include Monte Carlo SE as an estimate of simulation uncertainty.
- Publish code to execute the simulation study including user-written routines.

5.2

8

# Simulations for estimands

Simple example: consider three estimators for mean  $\mu$  of a distribution based on iid draws  $Y_1 \dots Y_n$ :

- $\theta_1$ : sample mean
- $\theta_2$ : sample 20% trimmed mean
- $\theta_3$ : sample median

If the distribution is symmetric, all three estimators should estimate the mean

- If the distribution is skewed, they will give different answers

# Simulations for estimands

For a particular choice of  $\mu$ ,  $n$ , and true underlying distribution:

- Generate independent draws  $Y_1 \dots Y_n$  from the distribution
- Compute  $\theta_1, \theta_2, \theta_3$  and repeat  $n_{sim}$  times to get:
  - $\theta_{1,1}, \dots, \theta_{1,n_{sim}}$
  - $\theta_{2,1}, \dots, \theta_{2,n_{sim}}$
  - $\theta_{3,1}, \dots, \theta_{3,n_{sim}}$

# Simulations for estimands

For  $k = 1, 2, 3$ , compute:

- $\widehat{E}[\hat{\theta}_k]$
- $\widehat{bias}(\hat{\theta})$
- $\widehat{Var}(\hat{\theta})$
- $\widehat{coverage}(\hat{\theta})$

1. What is the estimand(s)?
2. What is the estimator(s)?
3. What is the performance measure(s)?

# Simulations for hypothesis tests

Consider a t-test for whether the mean is equal to a specified value:

$$H_0 : \mu = \mu_0 \text{ vs. } H_1 : \mu \neq \mu_0$$

Suppose we want to evaluate whether the size/level of test achieves the advertised  $\alpha$ . We would:

- Generate data under the **null hypothesis** and calculate proportion of rejections of  $H_0$ .
- This approximates  $\Pr(\text{reject } H_0 \mid H_0 \text{ true})$ 
  - Should be  $\approx \alpha$  if the method works well

# Simulations for hypothesis tests

Suppose we want to evaluate **power**. We would:

- Generate data under some value of the alternative hypothesis  $\mu \neq \mu_0$  and calculate proportion of rejection of  $H_0$
- Approximates power, or  $\Pr(\text{reject } H_0 \mid H_1 \text{ true})$



# Simulations for hypothesis tests

Size/level

```
set.seed(125)

nsim = 10000
n = 20
sigma = sqrt(5/3)
mu0 = 1

# Generate data from null distribution:
dat = matrix(rnorm(n*nsim, mu0, sigma), ncol=nsim, byrow=TRUE)

opmean = apply(dat, 2, mean)
ses = sqrt(apply(dat, 2, var)/n)
tstats = (opmean - mu0)/ses
t05 = qt(0.975, n-1)
type1 = sum(abs(tstats) > t05)/nsim

type1
```

# Simulations for hypothesis tests

Power

```
set.seed(125)
nsim = 10000; n = 20; sigma = sqrt(5/3)
mu0 = 1
mu = 1.85 ## Generate data from alternative

dat = matrix(rnorm(n*nsim, mu, sigma), ncol=nsim, byrow=T)
opmean = apply (dat, 2, mean)
ses = sqrt(apply(dat, 2, var)/n)
tstats = (opmean - mu0)/ses
t05 = qt(0.975, n-1)

power = sum(abs (tstats) > t05)/nsim

power
```

```
[1] 0.7963
```

# Setting the seed

Simulations use pseudo-random numbers generated by a random number generating algorithm

Each random number is a deterministic function of the current *state* of the random number generator

- After a random number is produced, the state changes, ready to produce next random number
- State is set using a **seed**
  - After enough random draws, the state will eventually repeat (the path is circular)

Setting the seed ensures **reproducibility** of the simulation results.

# Setting the seed

How do you typically set the seed?

# Setting the seed

Morris et. al. recommends setting seed *once per simulation scenario*

- All  $n_{sim}$  simulated datasets will be generated with the same seed
- Avoids potential non-independence in simulated datasets

# Setting the seed

A simple simulation study shows that using sequential seeds for each dataset is a problem in Stata:

- 4 datasets generated with `nsim = nob = 4`, use seed 1:4 in stata

$x_1 = (0.1338766, 0.1364070, 0.4512149, \mathbf{0.0210242})$

$x_2 = (0.1364070, 0.4512149, \mathbf{0.0210242}, 0.3508981)$

$x_3 = (0.4512149, \mathbf{0.0210242}, 0.3508981, 0.9113581)$

$x_4 = (\mathbf{0.0210242}, 0.3508981, 0.9113581, 0.4707521).$

# Setting the seed

Using sequential seeds for each dataset appears to be less of a problem in R:

```
nobs = nsim = 4
set.seed(2025) # Morris approach
for(i in 1:nsim){
  # set.seed(i) # common approach
  print(runif(nobs))
}
```

# Setting the seed

Morris et. al. recommends setting seed *once per simulation scenario*

- In practice this is often unrealistic
  - Simulation code is often run in parallel or on a cluster
  - *Common approach is to set seed for each simulated dataset*

```
nobs = nsim = 4
# set.seed(2025) # Morris approach
for(i in 1:nsim){
  set.seed(i) # common approach
  print(runif(nobs))
}
```



# Setting the seed, best practices

- *Common approach is to set seed for each simulated dataset*
  - Best practice for this approach is to draw a random seed

```
nobs = nsim = 3
seeds = sample(1:10000, nsim)
print(seeds)
for(i in 1:nsim){
  # set.seed(i) # common approach
  set.seed(seeds[i]) # slightly better approach
  print(runif(nobs))
}
```

# Setting the seed, best practices

- Do not set the seed again later in the program, even if the methods being used have a stochastic component
- When writing functions and R packages, **do not hard code** a certain seed into the function/package
  - In fact, don't set a seed in the function or package at all, leave that up to the user

# Number of simulations

How many simulated datasets do you need to capture the true sampling distribution of your estimator?

# Number of simulations

How many simulated datasets do you need to capture the true sampling distribution of your estimator?

- Often this is chosen arbitrarily
  - $n_{sim} = 50, 100$  chosen to evaluate bias
  - $n_{sim} = 500, 1000$  chosen to evaluate coverage

Are these values large enough?

# Number of simulations

How many simulated datasets do you need to capture the true sampling distribution of your estimator?

- Often this is chosen arbitrarily
  - $n_{sim} = 50, 100$  chosen to evaluate bias
  - $n_{sim} = 500, 1000$  chosen to evaluate coverage

Are these values large enough?

- each simulated dataset yields a draw from the true sampling distribution of the estimator, so  $n_{sim}$  is the “**sample size**” on which Monte Carlo estimates of the mean, bias, SD, etc are based

# Monte Carlo standard errors

Monte Carlo standard errors quantify simulation uncertainty: they provide an estimate of the SE of (estimated) performance due to using finite  $n_{sim}$

Performance Measure	Definition	Estimate	Monte Carlo SE of Estimate
Bias	$E[\hat{\theta}] - \theta$	$\frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} \hat{\theta}_i - \theta$	$\sqrt{\frac{1}{n_{sim}(n_{sim}-1)} \sum_{i=1}^{n_{sim}} (\hat{\theta}_i - \bar{\theta})^2}$
EmpSE	$\sqrt{\text{Var}(\hat{\theta})}$	$\sqrt{\frac{1}{n_{sim}-1} \sum_{i=1}^{n_{sim}} (\hat{\theta}_i - \bar{\theta})^2}$	$\frac{\widehat{\text{EmpSE}}}{\sqrt{2(n_{sim}-1)}}$
Relative % increase in precision (B vs A) <sup>a</sup>	$100 \left( \frac{\widehat{\text{Var}}(\hat{\theta}_A)}{\widehat{\text{Var}}(\hat{\theta}_B)} - 1 \right)$	$100 \left( \left( \frac{\widehat{\text{EmpSE}}_A}{\widehat{\text{EmpSE}}_B} \right)^2 - 1 \right)$	$200 \left( \frac{\widehat{\text{EmpSE}}_A}{\widehat{\text{EmpSE}}_B} \right)^2 \sqrt{\frac{1 - \text{Corr}(\hat{\theta}_A, \hat{\theta}_B)^2}{n_{sim} - 1}}$
MSE	$E[(\hat{\theta} - \theta)^2]$	$\frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} (\hat{\theta}_i - \theta)^2$	$\sqrt{\frac{\sum_{i=1}^{n_{sim}} [(\hat{\theta}_i - \theta)^2 - \text{MSE}]^2}{n_{sim}(n_{sim}-1)}}$
Average ModSE <sup>a</sup>	$\sqrt{E[\widehat{\text{Var}}(\hat{\theta})]}$	$\sqrt{\frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} \widehat{\text{Var}}(\hat{\theta}_i)}$	$\sqrt{\frac{\widehat{\text{Var}}(\widehat{\text{Var}}(\hat{\theta}))}{4n_{sim} \times \text{ModSE}}}$ <sup>b</sup>
Relative % error in ModSE <sup>a</sup>	$100 \left( \frac{\text{ModSE}}{\widehat{\text{EmpSE}}} - 1 \right)$	$100 \left( \frac{\text{ModSE}}{\widehat{\text{EmpSE}}} - 1 \right)$	$100 \left( \frac{\text{ModSE}}{\widehat{\text{EmpSE}}} \right) \sqrt{\frac{\widehat{\text{Var}}(\widehat{\text{Var}}(\hat{\theta}))}{4n_{sim} \times \text{ModSE}} + \frac{1}{2(n-1)}}$
Coverage	$\Pr(\hat{\theta}_{low} \leq \theta \leq \hat{\theta}_{upp})$	$\frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} 1(\hat{\theta}_{low,i} \leq \theta \leq \hat{\theta}_{upp,i})$	$\sqrt{\frac{\text{Cover} \times (1 - \text{Cover})}{n_{sim}}}$
Bias-eliminated coverage	$\Pr(\hat{\theta}_{low} \leq \bar{\theta} \leq \hat{\theta}_{upp})$	$\frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} 1(\hat{\theta}_{low,i} \leq \bar{\theta} \leq \hat{\theta}_{upp,i})$	$\sqrt{\frac{B-E \text{ Cover} \times (1 - B-E \text{ Cover})}{n_{sim}}}$
Rejection % (power or type I error)	$\Pr(p_i \leq \alpha)$	$\frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} 1(p_i \leq \alpha)$	$\sqrt{\frac{\text{Power} \times (1 - \text{Power})}{n_{sim}}}$

# Number of simulations (bias)

Let's say we are interested in the **bias** of an estimator  $\theta$  (whose true value is  $\theta_0$ . )

# Number of simulations (bias)

Let's say we are interested in the **bias** of an estimator  $\theta$  (whose true value is  $\theta_0$ . )

- True bias:  $bias(\hat{\theta}) = E(\hat{\theta}) - \theta_0$
- Estimated bias:  $\widehat{bias}(\hat{\theta}) = \frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} \hat{\theta}_i - \theta_0$
- True variance:  $Var(\hat{\theta}) = E[(\hat{\theta} - \theta_0)^2]$
- Estimated variance:  $\widehat{Var}(\hat{\theta}) = \frac{1}{n_{sim}-1} \sum_i^{n_{sim}} (\hat{\theta}_i - \bar{\theta})^2$



# Number of simulations (bias)

Derive the Monte Carlo standard error of  $\widehat{bias(\hat{\theta})}$

$$\begin{aligned} SE\left(\widehat{bias(\hat{\theta})}\right) &= \sqrt{Var\left(\widehat{bias(\hat{\theta})}\right)} = \sqrt{Var\left(\frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} \hat{\theta}_i - \theta_0\right)} \\ &= \sqrt{\frac{1}{n_{sim}(n_{sim} - 1)} \sum_{i=1}^{n_{sim}} (\hat{\theta}_i - \bar{\theta})^2} \\ &= \frac{\sqrt{\widehat{Var(\hat{\theta})}}}{\sqrt{n_{sim}}} \end{aligned}$$

# Number of simulations (bias)

To ensure our estimate of the bias has an acceptable amount of Monte Carlo standard error  $e$ , we can set

$$\frac{\sqrt{\widehat{Var}(\hat{\theta})}}{\sqrt{n_{sim}}} = e$$

Solving for  $n_{sim}$  gives:

$$n_{sim} = \frac{\widehat{Var}(\hat{\theta})}{e^2}$$

Can guess  $\widehat{Var}(\hat{\theta})$  from prior knowledge or preliminary runs.

# Number of simulations (coverage)

**Coverage** is the probability that a confidence interval contains  $\theta$ .

- If  $\alpha$  is set to 0.05, estimated coverage should be about 0.95 if a method is performing well
- Common performance measure for an estimator for  $\theta$

The Monte Carlo standard errors for coverage are

$$SE\left(\widehat{coverage}(\hat{\theta})\right) = \frac{\sqrt{\widehat{cover}(1 - \widehat{cover})}}{\sqrt{n_{sim}}}$$

# Number of simulations (coverage)

Rearranging the previous equation, we get

$$n_{sim} = \frac{\widehat{cover}(1 - \widehat{cover})}{\left(\widehat{SE(coverage(\hat{\theta}))}\right)^2}$$

Let's assume  $\alpha = 0.05$  and we want the Monte Carlo SE to be no more than 0.5% (.005):

$$n_{sim} = \frac{0.95(1 - 0.95)}{(0.005)^2} = 1900$$

# Presenting your results

- **Key principle:** your simulation is useless unless other people (and your future self) can clearly understand what you did, why you did it, and what it means

# Presenting the results

- Before giving results, you must give the reader enough information to appreciate them.
  - State the objectives: Why do this simulation? What specific questions are you trying to answer?
  - State the **rationale** for choice of factors studied, assumptions made
  - Review all methods under study – be precise and detailed
  - Describe exactly how you generated data for each choice of factors. Enough detail should be given so that a reader could write his/her own program to reproduce your results

# Presenting the results

- Results must be presented in a form that:
  - Clearly answers the questions
  - Makes it easy to appreciate the main conclusions
- Some basic principles:
  - Only present a subset of results (“Results were qualitatively similar for all other scenarios we tried.”)
  - Only present information that is interesting (“Relative biases for all estimators were less than 2% under all scenarios and hence are not shown in the table.”)
  - The mode of presentation should be friendly...

# Presenting the results

- **Tables** are an obvious way to present results, however, some caveats:
  - Would a figure be more clear/compelling?
  - Place things to be compared adjacent to one another

Simple example: consider three estimators for mean  $\mu$  of a distribution based on iid draws  $Y_1 \dots Y_n$ :

- $\theta_1$ : sample mean
- $\theta_2$ : sample 20% trimmed mean
- $\theta_3$ : sample median



# Presenting the results

What's bad about this table?

	Sample	Mean	Trimmed	Mean	Median	
	Normal	$t_5$	Normal	$t_5$	Normal	$t_5$
Mean	0.98515	0.98304	0.98690	0.98499	0.99173	0.98474
Bias	-0.01485	-0.01696	-0.01310	-0.01501	-0.00827	-0.01526
SD	0.33088	0.33067	0.34800	0.31198	0.39763	0.35016
MSE	0.10959	0.10952	0.12116	0.09746	0.15802	0.12273

# Presenting the results

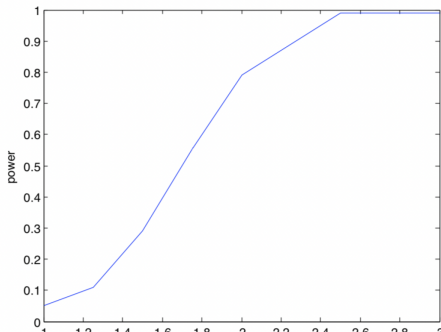
## Improved version

	Normal			$t_5$		
	Sample Mean	Trim Mean	Median	Sample Mean	Trim Mean	Median
Mean	0.99	0.99	0.99	0.98	0.98	0.98
Bias	-0.01	-0.01	-0.01	-0.02	-0.02	-0.02
SD	0.33	0.35	0.40	0.33	0.31	0.35
MSE	0.11	0.12	0.16	0.11	0.10	0.12

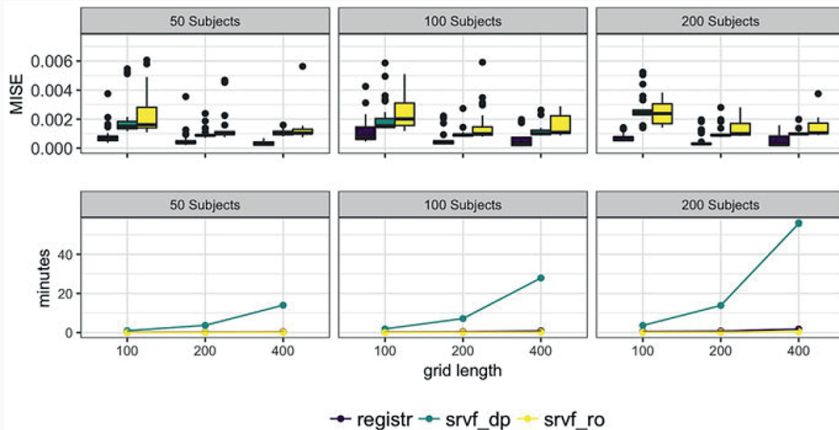
# Presenting the results

- Graphs/figures are often more effective than tables
- Example: Power of the t-test for  $H_0 : \mu = 1$  vs.  $H_1 : \mu \neq 1$  for normal data with  $n_{sim} = 10000$ ,  $n = 15$

$\mu$	1.0	1.25	1.50	1.75	2.00	2.50	3.00
power	0.05	0.11	0.29	0.55	0.79	0.99	0.99



# Presenting the results



# Final Takeaways

- **Principle 1:** A Monte Carlo simulation is just like any other experiment and requires planning
  - Can use experimental design principles
  - Shouldn't only choose factors favorable to a method you developed
  - $n_{sim}$  should be chosen to achieve acceptable precision

# Final Takeaways

- **Principle 2:** Save everything often!
  - Save the individual estimates in a file and then compute the mean, bias, SD, etc. later, as opposed to computing these summaries and saving only them
  - This is especially critical if the simulation takes a long time to run
  - Don't wait until the simulation ends to save. Long tasks can often be interrupted.

# Final Takeaways

- **Principle 3:** Keep  $n_{sim}$  small at first
  - Test and refine code until you are sure everything is working correctly before carrying out final production runs
  - Get an idea of how long it takes to process one data set, i.e., one iteration
  - Particularly important if production run will be submitted to the cluster to determine how many cores may be necessary to finish in an acceptable time frame.

# Final Takeaways

- **Principle 4:** Keep everything as reproducible as possible
  - Set and record the seed
  - Document your code!! Should be readable to your peer or your future self
  - Backup your code and results



# References

[1] A. Burton, D. G. Altman, P. Royston, et al. "The design of simulation studies in medical statistics". In: *Statistics in medicine* 25.24 (2006), pp. 4279-4292.

[2] N. Metropolis. "THE BEGINNING of the Monte Carlo Method". In: *Los Alamos Science* 15 (1987), pp. 125-30.

[3] T. P. Morris, I. R. White, and M. J. Crowther. "Using simulation studies to evaluate statistical methods". In: *Statistics in medicine* 38.11 (2019), pp. 2074-2102.



# Allowing for failures

Anticipate and **allow for failures** in a given simulation run. These can be do to:

- Rare events (e.g., assigned all subjects to a single treatment by chance, treatment perfectly confounded by a covariate in a permutation test)
- Lack of convergence of an optimization routine (fairly common in logistic regression)

Discard the failed iteration and repeat with the next random state but **record the number of failures** that occur to gauge how likely failure is in practice

- Use the results to judge whether the statistical procedure can reliably be used in the situation(s) being investigated

# Allowing for failures

To handle errors and other conditions in R

- `try()`: execution will continue when an error occurs
- `tryCatch()`: allows you to specify handler functions that give the computer directions about how to proceed when a condition is encountered

# Allowing for failures

This will throw an error:

```
f1 = function (x){  
  log(x)  
}  
  
f1("x")  
# Error in log(x) : non-numeric argument to  
# mathematical function
```

# Allowing for failures

Using `try()` avoids failure:

```
f1 = function (x){  
  try(log(x))  
}  
  
f1("x")
```

- Will print the error encountered during the `try()` statement, but doesn't break your code
- Using option `silent=TRUE` will suppress the message

# Allowing for failures

In addition to handling errors, `tryCatch()` allows you to specify how to proceed in the event of a warning, message, or **interrupt**

- An interrupt is raised when the user terminates the program execution by pressing Ctrl+C (or similar).

```
result <- tryCatch(  
  10 / "5", # Code that might cause an error  
  error = function(e) {  
    message("Caught an error: ", e$message)  
    return(NA)  
  }  
)  
  
print(result)
```