Do It Again

An Introduction to Simulation Experiments

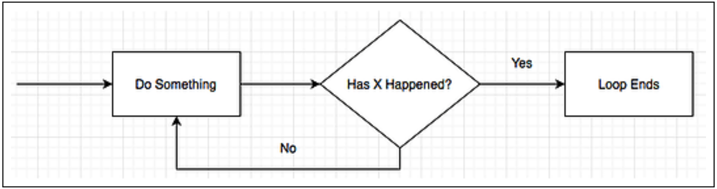
Dr. Matthew Sigal
Simon Fraser University

January, 2020

SIMULATION

Primary Goals

- Introduce Monte Carlo Simulation Study (MCSS) designs
  - What? Why? How?
  - How are results typically presented?
  - How could they be improved?
- Showcase how they are implemented in R with some best practice guidelines



Monte Carlo Designs. . .

Monte Carlo Simulation Studies provide a pivotal foundation for research in Quantitative Psychology and applied statistics at large.

Monte Carlo Designs...

Monte Carlo Simulation Studies provide a pivotal foundation for research in Quantitative Psychology and applied statistics at large.

What are Monte Carlo Simulation Studies?

MCSS are **experiments** with a wide variety of applications. Generally, certain parameters, which are known and fixed by the researcher, are used to **generate** random data and then estimate or **analyse** the behavior of other statistics across many *conditions*.

This is repeated over many *iterations* and then results are **summarized** for dissemination.

MCSS and the Central Limit Theorem

Given the population parameter ψ, let $\hat{\psi} = f(D)$ be the associated sample estimate, which is a function of data input D.

Theoretical CLT: given an *infinite number* of randomly sampled datasets D_i of size n, ψ can be recovered as the mean of all $f(D_i)$s.

MCSS and the Central Limit Theorem

Given the population parameter ψ, let $\hat{\psi} = f(D)$ be the associated sample estimate, which is a function of data input D.

Theoretical CLT: given an *infinite number* of randomly sampled datasets D_i of size n, ψ can be recovered as the mean of all $f(D_i)$s.

MCSS: Generate a large (but finite!) number of datasets ("replications", R) to obtain a sample approximation of the population parameter ($\tilde{\psi}$):

$$\tilde{\psi} = \frac{f(D_1) + f(D_2) + \cdots + f(D_R)}{R}$$

Further...

- While this seems reasonable for explaining concepts like the standard error of the mean, this holds for virtually *any* statistic and data generating mechanism (Mooney, 1997).
- Further, the sampling error of ψ can be approximated by finding the standard deviation of all $f(D_i)$ sets:

$$SE(\tilde{\psi}) = \sqrt{\frac{[f(D_1) - \tilde{\psi}]^2 + \cdots + [f(D_R) - \tilde{\psi}]^2}{R}},$$

...which is interpreted as *the standard deviation of a statistic under a large number of random samples* — an empirically obtained estimate of the standard error that does not require or assume an infinite number of samples.

The General Structure

1. **Generate** a dataset with n values according to some probability density function (e.g., normal, log-normal, binomial, χ^2, etc.).
2. **Analyse** the generated data by finding the mean of the sampled data, and store this value for later use.
3. Repeat steps 1 and 2 R times. Once complete, **summarise** the set of stored values with an appropriate statistic (e.g. mean, standard deviation).

Manipulate!

Once this structure is built, all sorts of things can be manipulated: generating distribution, sample size, number of replications, heterogeneity of variance, and so on.

Monte Carlo Designs…

In general, they have been used to:

- Evaluate the performance (e.g., Power/Type I error rates) of a new statistic or under various assumption violations
  - Examine the effects of skewness and kurtosis in linear mixed models (Arnau et al., 2013)

Monte Carlo Designs...

In general, they have been used to:

- Evaluate the performance (e.g., Power/Type I error rates) of a new statistic or under various assumption violations
  - Examine the effects of skewness and kurtosis in linear mixed models (Arnau et al., 2013)
- To see how well parameters are recovered in specific conditions
  - Investigate the behaviour of statistics and estimaters at various sample sizes (Schönbrodt and Perugini, 2013; Chalmers and Flora, 2014)
  - Determine the behavior of model fit statistics in complex multivariate systems of equations (Heene et al., 2012; Bollen et al., 2014)

Monte Carlo Designs...

In general, they have been used to:

- Evaluate the performance (e.g., Power/Type I error rates) of a new statistic or under various assumption violations
  - Examine the effects of skewness and kurtosis in linear mixed models (Arnau et al., 2013)
- To see how well parameters are recovered in specific conditions
  - Investigate the behaviour of statistics and estimaters at various sample sizes (Schönbrodt and Perugini, 2013; Chalmers and Flora, 2014)
  - Determine the behavior of model fit statistics in complex multivariate systems of equations (Heene et al., 2012; Bollen et al., 2014)
- Simulate 'realistic' data to address hard to study phenomena
  - To estimate if lower income areas have more pedestrian casualties (Noland et al., 2013)
  - Projections of teen pregnancy rates (Sayegh et al., 2010)

Origins

- Invented in the 1940s by Stanislaw Ulam, while working on nuclear weapons projects at Los Alamos National Laboratory.
- Was ill and ended up pondering the success rates of solitaire:

> *...what are the chances that a Canfield solitaire will come out successfully? After spending a lot of time trying to estimate them by pure combinatorial calculations, I wondered whether a more practical method... might... be to lay it out say one hundred times and... count the number of successful plays.*

- Due to the war effort, the project required a code name. Nicholas Metropolis suggested "Monte Carlo", after the casino in Monaco where Ulam's uncle gambled.

Within Psychology

MCSS are especially prevalent in the pages of *Multivariate Behavioral Research* and *Structural Equation Modeling* (*SEM*). In fact, these two journals have printed specific guides for researchers:

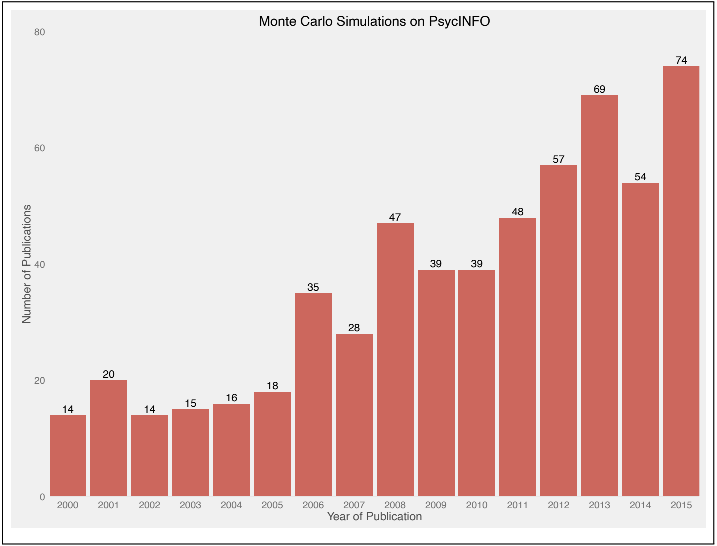
- Skrondal (2000) - *Design and analysis of Monte Carlo experiments: Attacking the conventional wisdom*
- Paxton, Curran, Bollen, et al. (2001) - *Monte Carlo experiments: Design and implementation*
- Boomsma (2013) - *Reporting Monte Carlo studies in Structural Equation Modeling*

Within Psychology

For SEM in particular, MCSS are an excellent approach for evaluating estimators and goodness-of-fit statistics under a variety of conditions, model complexity, and model misspecification (e.g., Kenny et al., 2015).

Paxton et al.: "...many topics in SEM would benefit from an empirical analysis through Monte Carlo methods" (2001, p. 288).

A search for **peer reviewed** articles using the query `all("Monte Carlo Simulation")` in **scholarly journals** on **PsycINFO**...



Conducting MCSS Research

Prep Work

1) Develop a theoretically derived research question and choose an appropriate software package.

Generate

2) Design specific experimental conditions and select values for the population parameters.

Analyse

3) Execute the simulation and repeat.
4) Troubleshoot and verify.

Summarise

5) Condense results from across iterations
6) Prepare results for communication

Conducting MCSS: An Introduction

Let's say Georgie is interested in the ability of a sample mean ($\overline{x}$) to recover μ and if the CLT approximation for the standard error is reasonable, given three different sample sizes.

Simulation Design

- Choice of generating distribution: *normal*
- Values of interest: *the mean*, *the standard error*
- Manipulation of interest: *sample size* (5, 30, 60)

Georgie's First Simulation: Setup

```
# Design
R <- 5000   # set 5,000 replications
mu <- 10    # set mu to 10
sigma <- 2   # set standard deviation to 2
N <- c(5, 30, 60)  # set 3 sample size conditions

# Results
res <- matrix(0, R, 3) # create a null matrix
                       # (with R rows, and 3 columns)
                       # to store output.
colnames(res) <- N # name columns (5, 30, 60)

head(res, n = 2)
```

```
##      5 30 60
## [1,] 0  0  0
## [2,] 0  0  0
```

Georgie's First Simulation: Replications

```
set.seed(77)  # Set seed to make analysis replicable
for(i in N){  # i = 5/30/60, across the 3 iterations
   for(r in 1:R){  # 1:R creates a vector 1,2,3,...,R
     dat <- rnorm(n = i, mean = mu, sd = sigma)
        # generate random data from a normal
        # distribution with set mean and sd
     res[r, as.character(i)] <- mean(dat)
        # return mean of dat and put it in res on row
        # r and in either column 5, 30, or 60.
   }
}
head(res, n = 2)
```

```
##             5       30       60
## [1,] 10.95739 10.11153 10.07469
## [2,] 10.64917 10.01001  9.90292
```

Georgie's First Simulation: Summarise

```r
# summarise by calculating mean for each column
apply(res, 2, mean)
```

```
##        5       30       60
## 10.00193 10.00089 10.00194
```

```r
# summarise by calculating s for each column
apply(res, 2, sd)
```

```
##         5        30        60
## 0.8892208 0.3684190 0.2575624
```

Georgie's First Simulation: Summarise

```r
# summarise by calculating mean for each column
apply(res, 2, mean)
```

```
##        5       30       60
## 10.00193 10.00089 10.00194
```

```r
# summarise by calculating s for each column
apply(res, 2, sd)
```

```
##         5         30         60
## 0.8892208 0.3684190 0.2575624
```

Georgie's Observations

- μ was recovered well regardless of n.
- Sampling variability of the estimates decreased as n increased.
- Empirical SEs can be compared against CLT ($\sigma/\sqrt{n}$):
  - 0.894, 0.365, and 0.258

Conducting MCSS: A WARNING

ABORT

While "for loops" are useful for introducing simulation designs they **should not** be used if at all possible:

- Setup mixes generate and summarise steps
- For loops become increasingly complex as the design expands (nested loops)
- Objects can be easily overwritten accidentally
- Design change might require overhaul of entire loop structure
- Deciphering and debugging for loops is hell

Conducting MCSS: What to look for in Software

What we want. . .

- An overarching philosophy for structuring MCSS that clearly delineates between generate, analyse, and summarise steps.
- A structure that can be expanded as needed for various designs.
- Convenience features, e.g.:
  - Resample non-convergent results
  - Support parallel computation
  - Save/restore results in case of power failures
  - Explicit tools for debugging

Conducting MCSS: My Recommendation

Highly recommended: `SimDesign` in `R` (Chalmers, 2018):

```
install.packages("SimDesign")
library(SimDesign)
```

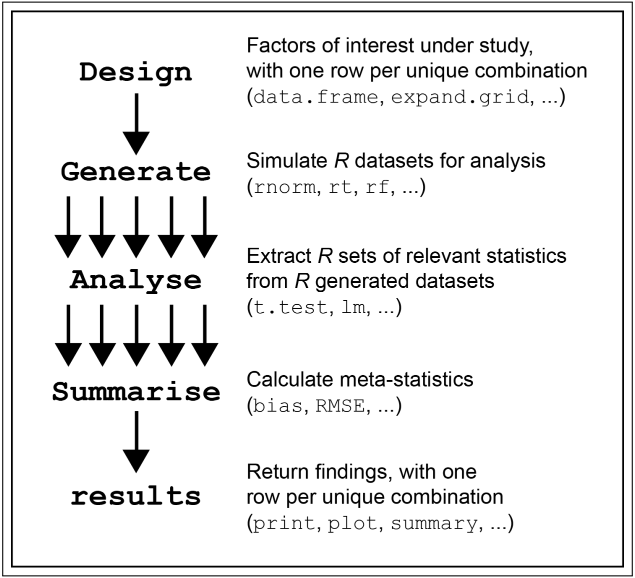
What does SimDesign provide?

Core elements of SimDesign make explicit reference to the generate-analyse-summarise paradigm:

```
Design <- createDesign(...)

Generate <- function(...) ...
Analyse <- function(...) ...
Summarise <- function(...) ...

results <- runSimulation(...)
```

This structure can be applied to any simulation study, regardless of its complexity!

The `SimDesign` Skeleton



The Helper: `SimFunctions()`

```r
SimFunctions("MySim", comments = TRUE) # creates .R script w/comments
```

```r
#-------------------------------------------------------------------

library(SimDesign)

### Define design conditions
Design <- createDesign(condition1 = NA,
                       condition2 = NA)

#-------------------------------------------------------------------

### Define essential simulation functions

Generate <- function(condition, fixed_objects = NULL) {
    # Define data generation code ...

    # Return a vector, matrix, data.frame, or list
    dat <- data.frame()
    dat
}
```

`MySim.R`, continued:

```
Analyse <- function(condition, dat, fixed_objects = NULL) {
    # Run statistical analyses of interest ...

    # Return a named vector or list
    ret <- c(stat1 = NaN, stat2 = NaN)
    ret
}

Summarise <- function(condition, results, fixed_objects = NULL) {
    # Summarise the simulation results ...

    # Return a named vector of results
    ret <- c(bias = NaN, RMSE = NaN)
    ret
}

#-------------------------------------------------------------------

### Run the simulation

res <- runSimulation(design=Design, replications=1000, generate=Generate,
                     analyse=Analyse, summarise=Summarise)
res
```

It is. . . by Design

The "design" of a simulation study is typically a (fully-crossed) set of factors. `SimDesign` uses a `tibble` to store this:

```
Design <- createDesign(sample_size = c(5, 30, 60))
Design
```

```
## # A tibble: 3 x 1
##   sample_size
##         <dbl>
## 1           5
## 2          30
## 3          60
```

Benefits:

- `Design` will be accessed sequentially (top to bottom), so it is easy to see what parameters are being passed and when.
- Rows of `Design` can be filtered, just as you would subset any other data.
- Columns can be added to incorporate other factors!

createDesign()

Add another variable to create fully-crossed design object:

```r
Design <- createDesign(sample_size = c(30, 60, 120),
                       distribution = c('norm', 'chi'))
Design
```

```
## # A tibble: 6 x 2
##   sample_size distribution
##         <dbl> <chr>
## 1          30 norm
## 2          60 norm
## 3         120 norm
## 4          30 chi
## 5          60 chi
## 6         120 chi
```

createDesign()

Use `subset` argument to remove unwanted rows:

```
Design <- createDesign(sample_size = c(30, 60, 120),
                       distribution = c('norm', 'chi'),
                       subset = !(sample_size == 60))
Design
```

```
## # A tibble: 4 x 2
##   sample_size distribution
## *       <dbl> <chr>
## 1          30 norm
## 2         120 norm
## 3          30 chi
## 4         120 chi
```

Generate This!

`Generate()` is a function that has only 1 required input: `condition` (a single row from `Design`) and uses parameters from that row to prepare a single dataset:

```
Generate <- function(condition, fixed_objects = NULL) {
  dat <- rnorm(n = condition$sample_size, mean = 10, sd = 2)
  dat
}
```

- Note the use of `condition$` to access variables from `Design`.
- Use `if()` statements if needed (e.g., for generating distribution).

Analyse That!

The purpose of `Analyse()` is to calculate and store all statistics of interest from each iteration.

For example, if we are only interested in the mean:

```
Analyse <- function(condition, dat, fixed_objects = NULL) {
  ret <- mean(dat)
  ret
}
```

This code will be called R times for each row of the `Design` matrix and can be used to return multiple values, if needed.

Then Summarise!

`Summarise()` is where we compute meta-statistics such as means, standard deviations, degree of bias, root mean-square error (RMSE), detection rates, and so on.

```
Summarise <- function(condition, results, fixed_objects = NULL) {
  c_mean <- mean(results)
  c_se <- sd(results)
  ret <- c(mu = c_mean, se = c_se)  # create a named vector
  ret
}
```

For each row of the design matrix, `SimDesign` will return the mean and standard error of the R replications as well as the number of replications, computation time, and a summary of any warnings that occurred.

runSimulation()

The final step is to pass the objects to `runSimulation()`:

```
results <- runSimulation(design=Design, replications = 5000,
    generate=Generate, analyse=Analyse, summarise=Summarise)
```

- Useful optional arguments:
  - `seed`: Set a random value seed for reproducability.
  - `save`: Save results to an external file.
  - `parallel`/`ncores`: Use parallel processing.
  - `debug`: Set to jump inside a running simulation (via `browser()`). Options include: `error`, `all`, `generate`, `analyse`, `summarise`.

. . . But what about the results?

MCSS Presentation, An Example

Even results from fairly simple MCSS produce a large amount of output, which are often presented in *very* long tables. Ramsey & Ramsey (2009) in the *British Journal of Mathematical and Statistical Psychology* had a straight-forward design:

- Goal: compare the performance of 10 pairwise multiple comparison procedures (MCPs) in an ANOVA framework
- Design:
  1. degree of heteroskedasticity (c, equal variance, and multiplied by 2, 4, and 10)
  2. number of groups (k, from 4 to 8)
  3. sample size per group (n, from 2 to 500)
- Primary output: Type I error rates from full true null models.

What might be included in a publication?

Table 1. Type I errors for 10 MCPs at $\alpha = .05$, $k = 4$, equal n_i, variance multiplier c and a true full null hypothesis

n	T3	C	GH	GF*	PF*	GH9	GH8	GH7	GH6	GH5
(a) $c = 1$										
2	.0791	.0117	.0445	.0048	.0039	.0368	.0313	.0260	.0197	.0158
3	.0397	.0135	.0512	.0137	.0171	.0439	.0385	.0331	.0272	.0216
4	.0391	.0157	.0514	.0186	.0256	.0450	.0388	.0335	.0283	.0234
5	.0383	.0173	.0499	.0216	.0311	.0446	.0379	.0328	.0265	.0218
6	.0426	.0231	.0536	.0286	.0371	.0472	.0426	.0375	.0309	.0253
7	.0388	.0239	.0497	.0259	.0351	.0434	.0386	.0338	.0295	.0252
8	.0446	.0301	.0534	.0305	.0373	.0479	.0443	.0391	.0342	.0292
9	.0410	.0263	.0511	.0301	.0390	.0460	.0407	.0360	.0290	.0238
10	.0471	.0343	.0562	.0363	.0441	.0518	.0468	.0410	.0353	.0291
11	.0417	.0313	.0531	.0326	.0423	.0464	.0412	.0368	.0311	.0256
12	.0423	.0333	.0518	.0342	.0433	.0470	.0422	.0376	.0317	.0261
13	.0446	.0371	.0545	.0367	.0440	.0488	.0439	.0390	.0344	.0295
14	.0439	.0367	.0525	.0338	.0436	.0474	.0437	.0381	.0333	.0274
21	.0430	.0401	.0517	.0387	.0479	.0473	.0417	.0369	.0321	.0269
24	.0422	.0406	.0510	.0383	.0475	.0469	.0414	.0367	.0323	.0284
28	.0405	.0400	.0496	.0371	.0467	.0448	.0397	.0354	.0301	.0249
29	.0402	.0391	.0472	.0356	.0426	.0436	.0387	.0333	.0288	.0236
30	.0416	.0408	.0510	.0398	.0498	.0456	.0402	.0361	.0305	.0248
31	.0436	.0432	.0525	.0406	.0478	.0477	.0418	.0369	.0311	.0257
32	.0427	.0425	.0513	.0388	.0467	.0474	.0411	.0365	.0320	.0269
33	.0365	.0364	.0457	.0333	.0425	.0394	.0352	.0312	.0260	.0214
34	.0467	.0469	.0546	.0430	.0517	.0506	.0460	.0413	.0360	.0301
35	.0409	.0413	.0488	.0361	.0434	.0436	.0397	.0344	.0287	.0234
36	.0431	.0440	.0527	.0402	.0480	.0463	.0418	.0367	.0322	.0264
37	.0412	.0420	.0495	.0377	.0442	.0448	.0395	.0359	.0310	.0249
38	.0423	.0439	.0521	.0375	.0459	.0465	.0408	.0351	.0295	.0246
39	.0446	.0461	.0545	.0403	.0485	.0485	.0435	.0381	.0325	.0274
40	.0405	.0425	.0487	.0383	.0467	.0445	.0394	.0353	.0290	.0245
50	.0496	.0520	.0581	.0457	.0528	.0534	.0481	.0416	.0356	.0303
60	.0417	.0448	.0506	.0391	.0460	.0452	.0402	.0358	.0314	.0259
70	.0401	.0438	.0482	.0384	.0462	.0437	.0392	.0340	.0284	.0236
80	.0437	.0491	.0534	.0440	.0517	.0486	.0430	.0376	.0328	.0277
100	.0431	.0484	.0517	.0415	.0484	.0469	.0419	.0371	.0324	.0269
200	.0421	.0498	.0515	.0419	.0500	.0462	.0411	.0363	.0318	.0283
300	.0430	.0489	.0499	.0423	.0500	.0464	.0421	.0374	.0317	.0265
400	.0392	.0463	.0464	.0395	.0464	.0426	.0382	.0346	.0300	.0251
500	.0450	.0534	.0537	.0452	.0517	.0480	.0439	.0394	.0344	.0296
Max	.0791	.0534	.0581	.0457	.0528	.0534	.0481	.0416	.0360	.0303
(b) $c = 2$										
2	.0898	.0109	.0512	.0073	.0056	.0407	.0321	.0253	.0209	.0167
3	.0451	.0158	.0585	.0156	.0199	.0515	.0430	.0367	.0310	.0259
4	.0478	.0225	.0598	.0240	.0324	.0531	.0469	.0402	.0357	.0292
5	.0467	.0247	.0586	.0267	.0418	.0519	.0460	.0405	.0345	.0282
6	.0448	.0275	.0563	.0283	.0440	.0502	.0444	.0384	.0324	.0270
7	.0433	.0288	.0543	.0293	.0438	.0491	.0431	.0382	.0329	.0271
14	.0383	.0348	.0449	.0261	.0443	.0420	.0379	.0335	.0280	.0226
21	.0427	.0424	.0525	.0298	.0485	.0475	.0418	.0370	.0312	.0263

Table 1. *(Continued)*

n	T3	C	GH	GF*	PF*	GH9	GH8	GH7	GH6	GH5
25	.0391	.0394	.0456	.0267	.0430	.0411	.0380	.0337	.0295	.0250
26	.0418	.0437	.0506	.0299	.0465	.0457	.0413	.0365	.0315	.0249
27	.0424	.0435	.0527	.0320	.0472	.0470	.0413	.0354	.0317	.0262
28	.0393	.0408	.0465	.0269	.0439	.0434	.0382	.0336	.0289	.0235
29	.0398	.0409	.0481	.0285	.0447	.0434	.0388	.0348	.0296	.0246
30	.0395	.0410	.0470	.0269	.0426	.0419	.0383	.0333	.0287	.0238
31	.0427	.0434	.0497	.0299	.0466	.0451	.0417	.0364	.0326	.0279
32	.0393	.0418	.0490	.0280	.0457	.0436	.0387	.0339	.0297	.0247
33	.0402	.0422	.0479	.0286	.0462	.0440	.0393	.0353	.0298	.0249
34	.0396	.0414	.0471	.0256	.0429	.0430	.0378	.0341	.0290	.0249
35	.0425	.0450	.0498	.0303	.0490	.0457	.0411	.0362	.0313	.0266
Max	.0898	.0450	.0598	.0320	.0490	.0531	.0469	.0405	.0357	.0292
(c) $c = 4$										
2	.1195	.0179	.0689	.0133	.0120	.0566	.0464	.0387	.0322	.0267
3	.0618	.0291	.0742	.0241	.0306	.0670	.0600	.0520	.0457	.0397
4	.0513	.0299	.0623	.0238	.0334	.0567	.0504	.0440	.0385	.0328
5	.0502	.0341	.0609	.0246	.0378	.0550	.0494	.0437	.0382	.0311
6	.0436	.0317	.0547	.0203	.0364	.0489	.0432	.0355	.0308	.0270
7	.0397	.0330	.0489	.0219	.0390	.0440	.0396	.0350	.0302	.0254
8	.0402	.0336	.0507	.0213	.0379	.0450	.0401	.0350	.0296	.0247
9	.0419	.0378	.0516	.0242	.0410	.0469	.0416	.0365	.0309	.0261
10	.0390	.0371	.0494	.0233	.0427	.0440	.0389	.0341	.0289	.0248
11	.0404	.0379	.0479	.0205	.0369	.0443	.0400	.0350	.0297	.0253
12	.0388	.0377	.0467	.0210	.0403	.0419	.0384	.0342	.0291	.0239
13	.0388	.0387	.0480	.0220	.0423	.0436	.0386	.0349	.0298	.0246
14	.0395	.0395	.0482	.0206	.0400	.0434	.0391	.0350	.0303	.0247
15	.0365	.0370	.0454	.0208	.0413	.0403	.0361	.0307	.0267	.0218
16	.0335	.0351	.0414	.0186	.0354	.0378	.0328	.0290	.0253	.0212
17	.0359	.0386	.0443	.0189	.0361	.0405	.0356	.0311	.0269	.0229
18	.0380	.0398	.0460	.0201	.0409	.0414	.0378	.0322	.0279	.0236
21	.0392	.0419	.0473	.0203	.0417	.0423	.0388	.0341	.0280	.0229
28	.0337	.0373	.0424	.0187	.0381	.0376	.0325	.0273	.0228	.0188
35	.0366	.0396	.0422	.0196	.0398	.0391	.0357	.0316	.0273	.0228
Max	.1195	.0450	.0742	.0320	.0490	.0670	.0600	.0520	.0457	.0397
(d) $c = 10$										
2	.1555	.0351	.0964	.0268	.0266	.0823	.0708	.0595	.0520	.0431
3	.0686	.0434	.0831	.0285	.0359	.0747	.0660	.0602	.0523	.0440
4	.0530	.0389	.0641	.0250	.0381	.0581	.0516	.0460	.0400	.0328
5	.0476	.0406	.0578	.0224	.0346	.0527	.0466	.0419	.0367	.0306
6	.0417	.0381	.0525	.0215	.0343	.0473	.0414	.0362	.0309	.0260
7	.0394	.0385	.0492	.0181	.0319	.0435	.0390	.0343	.0293	.0243
8	.0371	.0370	.0469	.0184	.0340	.0416	.0371	.0323	.0280	.0241
9	.0327	.0351	.0439	.0180	.0359	.0382	.0325	.0287	.0250	.0207
10	.0358	.0379	.0441	.0173	.0346	.0401	.0357	.0314	.0266	.0226
11	.0338	.0371	.0431	.0180	.0364	.0375	.0337	.0304	.0266	.0226
12	.0350	.0379	.0430	.0179	.0358	.0389	.0350	.0310	.0263	.0226
13	.0327	.0368	.0414	.0172	.0363	.0376	.0327	.0288	.0247	.0206
14	.0357	.0390	.0433	.0193	.0349	.0390	.0356	.0300	.0256	.0218
21	.0319	.0369	.0398	.0152	.0354	.0363	.0312	.0283	.0242	.0205

. . . and it is still going!

Table 1. (*Continued*)

n	T3	C	GH	GF*	PF*	GH9	GH8	GH7	GH6	GH5
27	.0315	.0369	.0397	.0161	.0352	.0347	.0312	.0277	.0230	.0196
Max	.1555	.0434	.0964	.0285	.0381	.0823	.0708	.0602	.0523	.0440
Max	.1555	.0534	.0964	.0457	.0528	.0823	.0708	.0602	.0523	.0440

Note. T3, Dunnett; C, Cochran; GH, Games–Howell; GF*, Games–Howell and Brown–Forsythe *F*; PF*, Peritz–Brown–Forsythe *F*; GH9, GH applied at 0.9α; GH8, GH at 0.8α; GH7, GH at 0.7α; GH6, GH at 0.6α; GH5, GH at 0.5α.

Even so, this table ignores:

- Many of the sample size comparisons
  - none of the (many) sample size conditions that pertain to unequal groups
- The number of groups factor. . .
  - this entire table only refers to $k = 4$!

Observations

MCSS Results

Output takes the form of **multi-dimensional tables** with dimensions pertaining to the results for one or more outcome measures (e.g., Type I error rate) for a particular set of design variables or conditions (e.g., sample size/generating distribution).

However, methods for conveying MCSS findings has typically been given little attention.

For instance, Paxton et al. (2001) state that results can be presented "descriptively, graphically, and inferentially" but provide little detail on how to do so.

MCSS Presentation

"...reading results from Monte Carlo studies in whatever form should be a revelatory task, not a baffling puzzlement."

–Boomsma, 2013, p. 534.

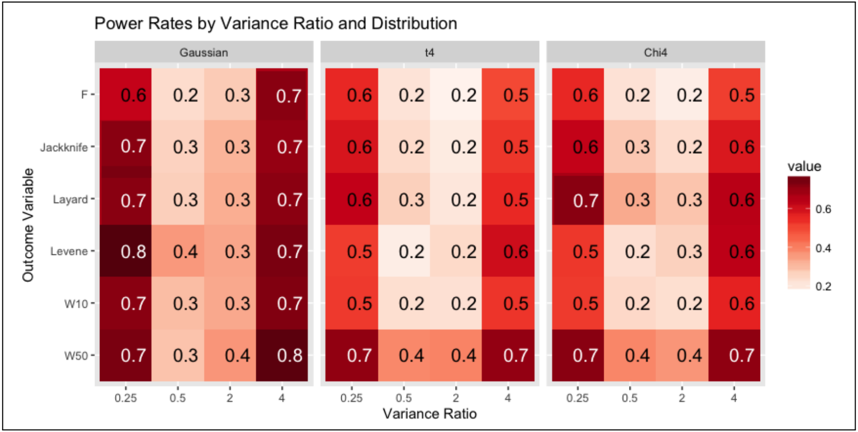
Issues with tabular displays

- Results nearly unreadable, except for looking up particular combinations of factors
- Many comparisons get hidden from view, especially for complex simulation designs with many factors
- Wearisome – patterns are difficult to discern at a glance

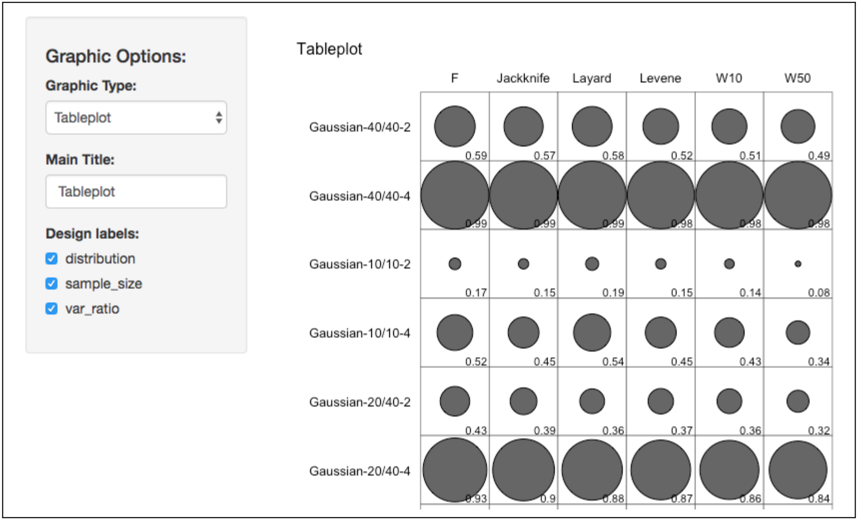
How can this situation be improved?

Shaded Tables 1

Shaded Tables 2



Interactive Exploration



Final Example - Type I Error Rates and Power

A quick study of Type I error (and power) rates for the independent groups t-test under violations of homogeneity of variance:

```r
library(SimDesign)
Design <- expand.grid(sample_size = c(30, 60, 120),
                      group_size_ratio = c(1, 2),
                      sd_ratio = c(1/4, 1, 4),
                      mean_diff = c(0, 0.5))
head(Design)
```

```
##   sample_size group_size_ratio sd_ratio mean_diff
## 1          30                1     0.25         0
## 2          60                1     0.25         0
## 3         120                1     0.25         0
## 4          30                2     0.25         0
## 5          60                2     0.25         0
## 6         120                2     0.25         0
```

Final Example - Type I Error Rates and Power

```
Generate <- function(condition, fixed_objects = NULL){
  # Attach() makes the variables in condition directly accesible
  Attach(condition)
  N1 <- sample_size / (group_size_ratio + 1)
  N2 <- sample_size - N1
  group1 <- rnorm(N1)
  group2 <- rnorm(N2,
                  mean=mean_diff,
                  sd=sd_ratio)
  dat <- data.frame(group = c(rep('g1', N1),
                              rep('g2', N2)),
                    DV = c(group1, group2))
  dat
}

Analyse <- function(condition, dat, fixed_objects = NULL){
  welch <- t.test(DV ~ group, dat)
  ind <- t.test(DV ~ group, dat, var.equal=TRUE)
  ret <- c(welch=welch$p.value, independent=ind$p.value)
  ret
}
```

Final Example - Type I Error Rates and Power

```
Summarise <- function(condition, results, fixed_objects = NULL){
  ret <- EDR(results, alpha = .05)
  ret
}
```

NOTE: `EDR()` is a `SimDesign` function for detection-based statistical tools; e.g., if $f(D)$ returns a p-value, then an estimate of the "true detection rate" (aka *empirical detection rate*) is approximated by:

$$\tilde{\rho} = \frac{I_\alpha[f(D_1)] + \cdots + I_\alpha[f(D_R)]}{R},$$

where I_α is an indicator function that returns 1 if the p-value from $f(D)$ is less than α and 0 otherwise. `EDR()` averages the values to obtain a proportion.

Final Example - Type I Error Rates and Power

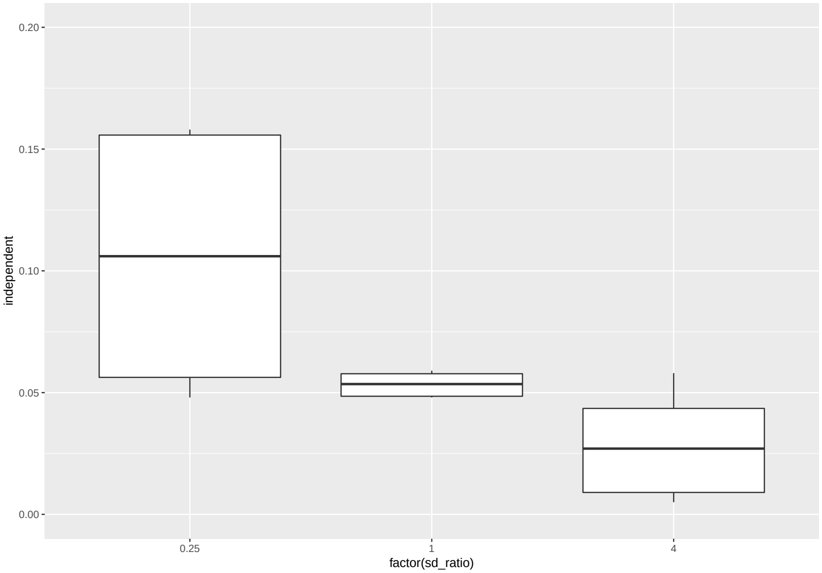
```r
results <- runSimulation(design = Design, replications = 1000,
                         parallel = TRUE, generate = Generate,
                         analyse = Analyse, summarise = Summarise)
head(results)
```

```
## # A tibble: 6 x 10
##   sample_size group_size_ratio sd_ratio mean_diff welch independent
##         <dbl>            <dbl>    <dbl>     <dbl> <dbl>       <dbl>
## 1          30                1     0.25         0 0.05         0.06
## 2          60                1     0.25         0 0.041        0.048
## 3         120                1     0.25         0 0.054        0.055
## 4          30                2     0.25         0 0.056        0.157
## 5          60                2     0.25         0 0.049        0.158
## 6         120                2     0.25         0 0.044        0.152
## # ... with 4 more variables: REPLICATIONS <int>, SIM_TIME <dbl>,
## #   COMPLETED <chr>, SEED <int>
```

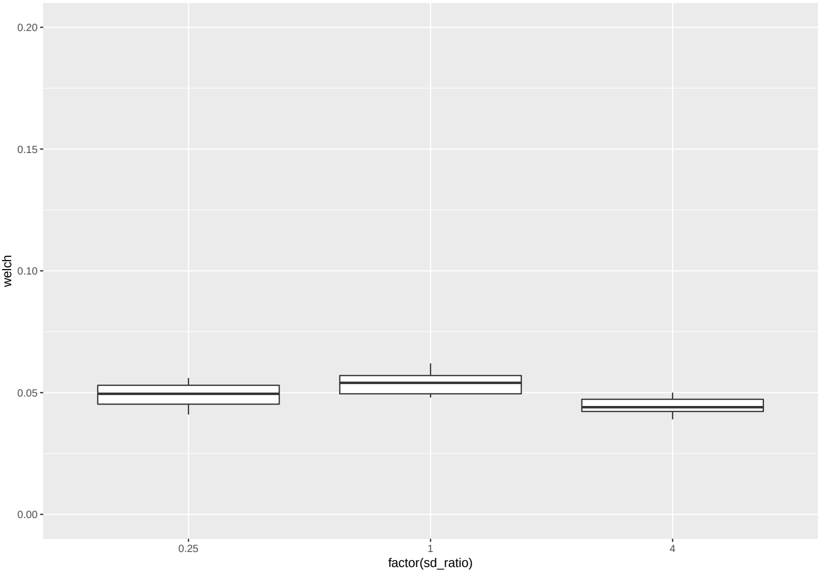
Final Example - Type I Error Rates and Power

```r
TypeI <- subset(results, mean_difference == 0)
Power <- subset(results, mean_difference != 0)

library(ggplot2)
ggplot(TypeI,
       aes(factor(standard_deviation_ratio), independent) +
  geom_boxplot() + ylim(c(0,.2))

ggplot(TypeI,
       aes(factor(standard_deviation_ratio), welch) +
  geom_boxplot() + ylim(c(0,.2))
```

Final Example - Type I Error Rates and Power



Final Example - Type I Error Rates and Power



Conclusion

- The theory of simulation studies is reasonable but dependent on the appropriate choice of parameters by the researcher.
- Many topics are amenable to MCSS designs!
- MCSS are fairly easy to implement, especially when one is able to harness the power of `R` and `SimDesign` (see Sigal and Chalmers, 2016).
- Presenting results from MCSS experiments via tables is the classic approach...
  - ...however, there is definitely room for improvement.

Arnau, J., Bendayan, R., Blanca, M. J., and Bono, R. (2013). The effects of skewness and kurtosis on the robustness of linear mixed models. *Behavioural Research*, 45(3):873–879.

Bollen, K. A., Harden, J. J., Ray, S., and Zavisca, J. (2014). BIC and alternative Bayesian Information Criteria in the selection of Structural Equation Models. *Structural Equation Modeling*, 21:1–19.

Boomsma, A. (2013). Reporting Monte Carlo studies in Structural Equation Modeling. *Structural Equation Modeling*, 20:518–540.

Chalmers, P. (2018). *SimDesign: Structure for Organizing Monte Carlo Simulation Designs*. R package version 1.11, https://CRAN.R-project.org/package=SimDesign.

Chalmers, R. P. and Flora, D. B. (2014). Maximum-likelihood estimation of noncompensatory IRT models with the MH-RM algorithm. *Applied Psychological Measurement*, 38(5):339–358.

Heene, M., Hilbert, S., Freudenthaler, H. H., and Bühner, M. (2012). Sensitivity of SEM fit indexes with respect to violations of uncorrelated errors. *Structural Equation Modeling*, 19:36–50.

Kenny, D. A., Kaniskan, B., and McCoach, D. B. (2015). The performance of RMSEA in models with small degrees of freedom. *Sociological Methods & Research*, 44(3):486–507.

Mooney, C. Z. (1997). *Monte Carlo Simulations*. Sage, Thousand Oaks, CA.

Noland, R. B., Klein, N. J., and Tulach, N. K. (2013). Do lower income areas have more pedestrian casualties? *Accident Analysis and Prevention*, 59:337–45.

Paxton, P., Curran, P. J., Bollen, K. A., Kirby, J., and Chen, F. (2001). Monte Carlo experiments: Design and implementation. *Structural Equation Modeling*, 8(2):287–312.

Ramsey, P. H. and Ramsey, P. P. (2009). Power and Type I errors for pairwise comparisons of means in the unequal variances case. *British Journal of Mathematical and Statistical Psychology*, 62:263–281.

Sayegh, M. A., Castrucci, B. C., Lewis, K., and Hobbs-Lopez, A. (2010). Teen pregnancy in Texas: 2005 to 2015. *Maternal and Child Health Journal*, 14(1):94–101.

Schönbrodt, F. D. and Perugini, M. (2013). At what sample size do correlations stabilize? *Journal of Research in Personality*, 47(5):609–612.

Sigal, M. J. and Chalmers, R. P. (2016). Play it again: Teaching statistics with Monte Carlo simulation. *Journal of Statistics Education*, 24(3):136–156.

Skrondal, A. (2000). Design and analysis of Monte Carlo experiments: Attacking the conventional wisdom. *Multivariate Behavioral Research*, 35(2):137—167.