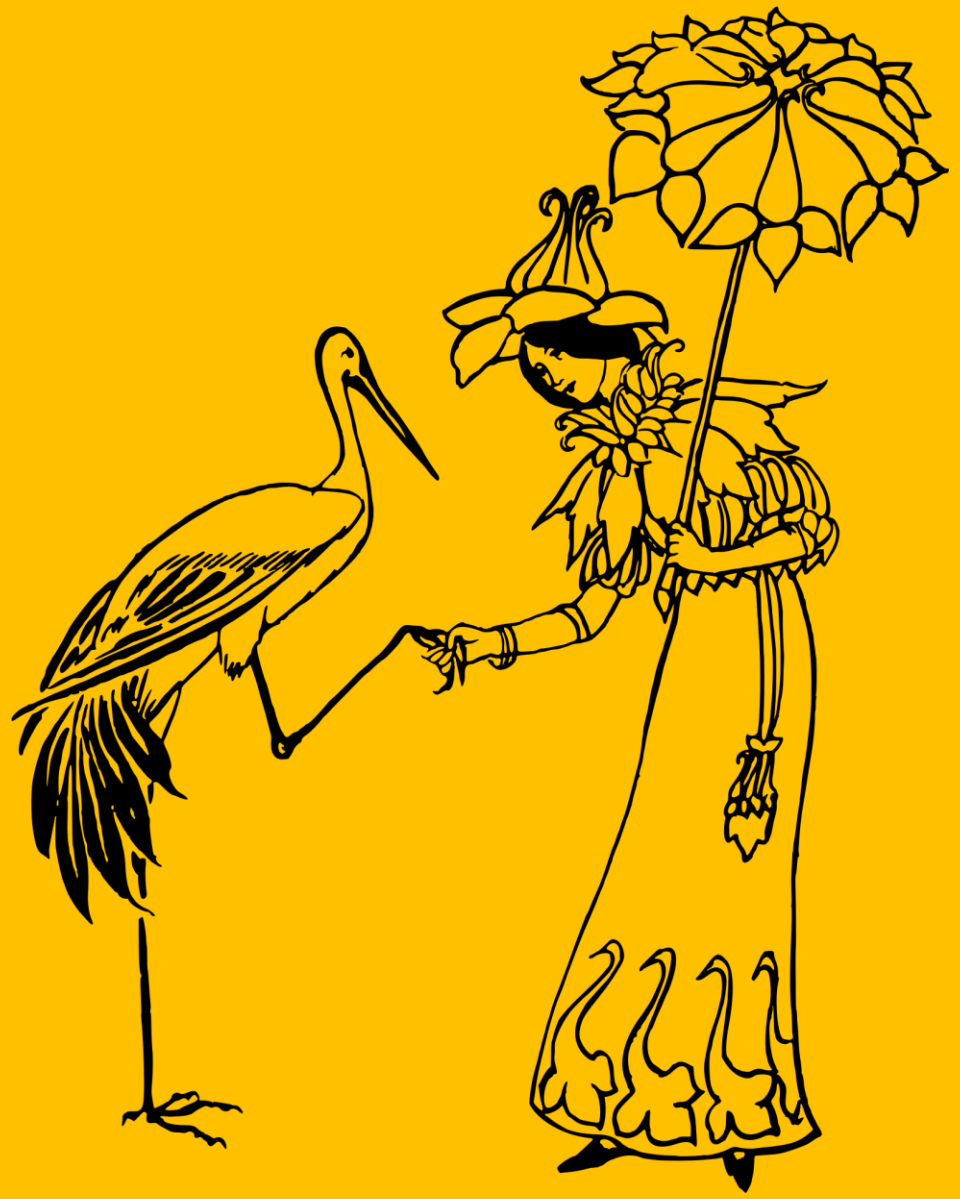
Yes, YOU can simulate!



https://openclipart.org/detail/280519/strange-meeting

Reproducible, Tidy Workflows with the Reimagined simpr Package

Ethan C. Brown, PhD

useR! 2024, Salzburg

The case for simpr

1. Do you want to understand the models you're using?
2. Are you planning a study, doing a power analysis, or trying to evaluate a statistical method?
3. Do you know that simulation can help but you're **overwhelmed by the complexity** of creating a simulation?

statisfactions.github.com/simpr

Understanding Your Models

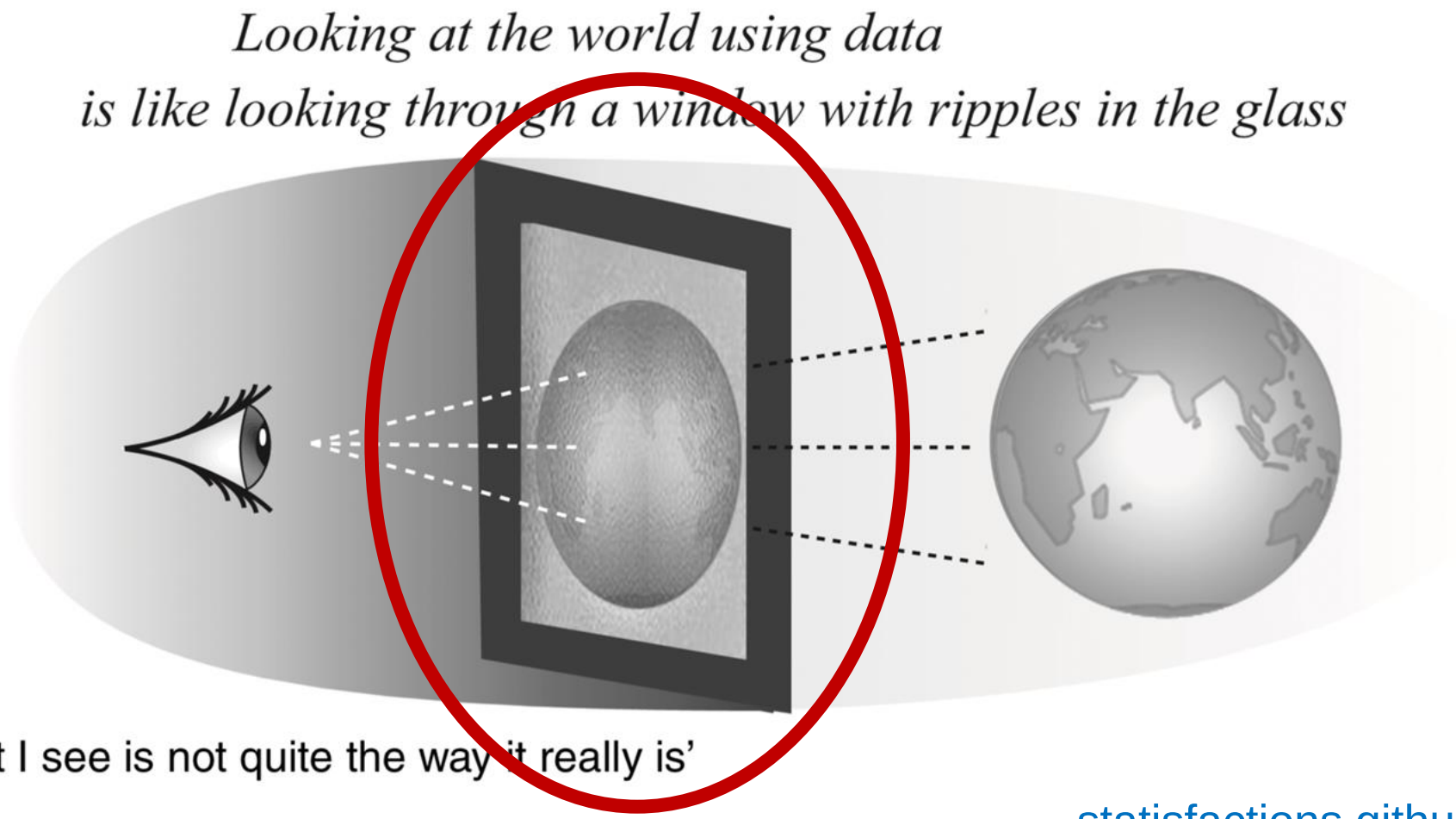
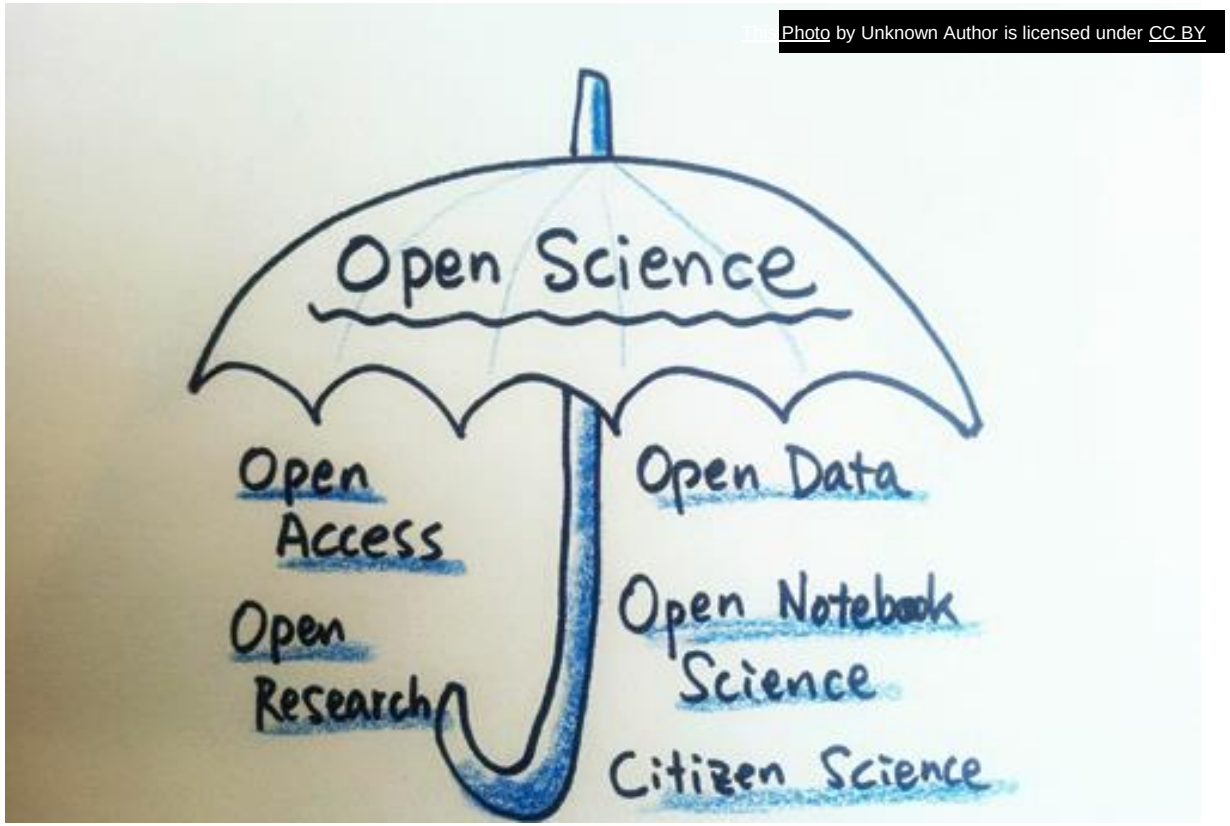


Fig. 2. 'What I see is not quite the way it really is'

Wild et al. (2011) statisfactions.github.com/simpr



OPEN SCIENCE AND "WHAT IFS"

Will this design answer our research questions?

vs

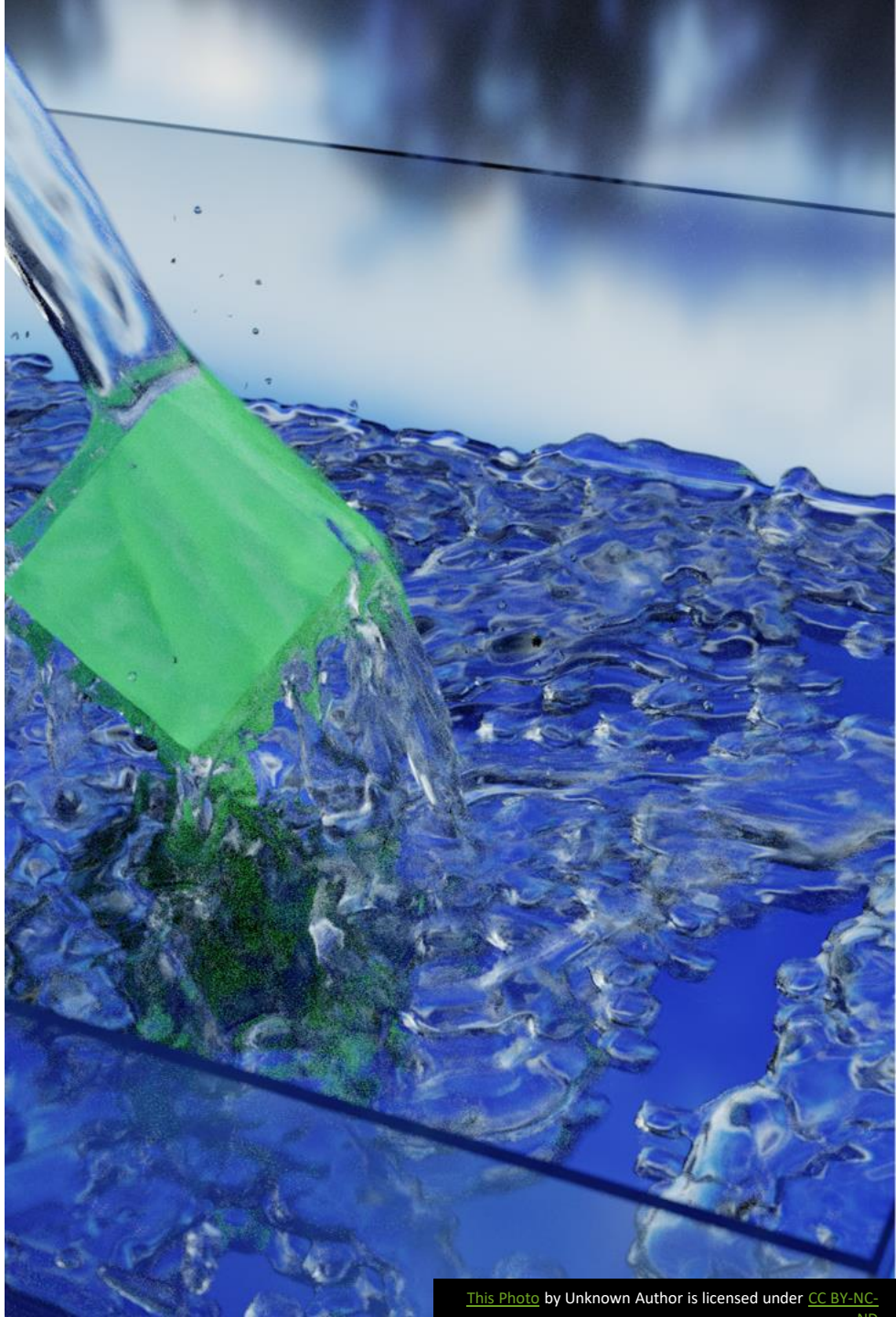
Under what conditions will this design answer our research questions?

- Transparently show the boundaries and assumptions of the design
- Support writing of preregistrations, prospectuses, and registered reports by a disciplined examination of possibilities
- Look at the ranges of possible quality (power, bias) based on what we assume is true about reality

statisfactions.github.com/simpr

SIMULATION-BASED DESIGN ANALYSIS

- Model the data-generating process, including sampling, assignment, measurement
- Generate simulated data based on the data-generating process
- Fit the proposed model and evaluate whatever outcomes we're interested in (power, bias, convergence…).



Is this your feeling when doing a simulation?

Power Analysis

Some notes from *Greg Snow* (tweaked by Ben Bolker) on power analysis (for a LMM rather than a GLMM, but the general principles are the same). Here is some code to get you started (based on some assumptions that may be way off):

```
library(lme4)

sim1 <- function(bSex=0, bFreq=0, bSF=0, b0=1000, Vsubj=1, Vword=1, Verror=1) {
    Subject <- rep( 1:60, each=50 )
    Word <- rep( 1:50, 60 )
    Sex <- rep(c('M','F'), each=50*30)
    ## or use expand.grid(), although it won't work perfectly for this case:
    ## expand.grid(Word=1:50,Subject=1:30,Sex=c('M','F')) would give
    ## subjects 1 to 30 in EACH sex rather subjects 1 to 60 of which
    ## half are each sex

    #  assume frequency is constant across word, random from 1-100
    tmp <- sample( 1:100, 50, replace=TRUE )
    Frequency <- tmp[Word]

    # random effects per subject
    S.re <- rnorm(60, 0, sqrt(Vsubj))

    # random effects per word
    W.re <- rnorm(50, 0, sqrt(Vword))

    # epsilons
    eps <- rnorm(50*60, 0, sqrt(Verror))

    # put it all together
    # or use model.matrix() for more complex problems
    ReactionTime <- b0 + bSex*(Sex=='M') + bFreq*Frequency + bSF*(Sex=='M')*Frequency +
        S.re[Subject] + W.re[Word] + eps

    # put into a data frame
    mydata <- data.frame( Subject = paste('s',Subject, sep=''),
                    Word = paste('w', Word, sep=''), Sex=Sex, Frequency=Frequency,
                    ReactionTime = ReactionTime)

    # analyze looking at interaction term with LR test
    fit1 <- lmer( ReactionTime ~ (Sex*Frequency) + (1|Subject) + (1|Word), data=mydata)
    fit2 <- lmer( ReactionTime ~ Sex + Frequency + (1|Subject) + (1|Word), data=mydata)
    anova(fit2,fit1)[2,"Pr(>Chisq)"]
}
```

Set random number seed for reproducibility:

Source: http://glmm.wikidot.com/power-analysis

statisfactions.github.com/simpr

Challenges of simulation

- Require understanding underlying statistical model
- Simulations make just as many assumptions as analytical methods, and may not be reasonable
- Custom simulations often difficult to code
- Simulations easy to mislead by missing fundamental features of the process

IS 100 STUDENTS ENOUGH?

Study: *Pre-post comparison of the "Triglicious" intervention*

Research question: *Did Triglicious improve students' math scores?*

Method: *Paired t-test*

statisfactions.github.com/simpr

Simulations in base R

```r
## Set up parameters
ns = c(100, 150, 200)
mean_diffs = c(10, 20, 30)
sds = c(50, 100)
reps = 10
```

statisfactions.github.com/simpr

Simulations in base R

```r
## Set up parameters
ns = c(100, 150, 200)
mean_diffs = c(10, 20, 30)
sds = c(50, 100)
reps = 10
## Bring together into data frame
results_template = expand.grid(n = ns,
                               mean_diff = mean_diffs,
                               sd = sds, p.value = NA_real_)

base_r_sim = results_template[rep(1:nrow(results_template), each = reps),]
```

statisfactions.github.com/simpr

Simulations in base R

```r
## Set up parameters
ns = c(100, 150, 200)
mean_diffs = c(10, 20, 30)
sds = c(50, 100)
reps = 10
## Bring together into data frame
results_template = expand.grid(n = ns,
                               mean_diff = mean_diffs,
                               sd = sds, p.value = NA_real_)

base_r_sim = results_template[rep(1:nrow(results_template), each = reps),]

## Loop over rows of the data frame and calculate the p-value
for(i in 1:nrow(results_template)) {
  params = results_rep[i,]
  pre = rnorm(params$n, 0, params$sd)
  post = pre + rnorm(params$n, params$mean_diff, params$sd)

  base_r_sim$p.value[i] = t.test(pre, post)$p.value
}
```

statisfactions.github.com/simpr

```
##         n mean_diff  sd      p.value
## 1     100         10  50 5.977957e-01
## 1.1   100         10  50 2.245753e-01
## 1.2   100         10  50 4.449589e-01
## 1.3   100         10  50 6.099931e-01
## 1.4   100         10  50 3.258414e-01
## 1.5   100         10  50 1.647985e-01
## 1.6   100         10  50 6.690890e-01
## 1.7   100         10  50 1.333951e-02
## 1.8   100         10  50 9.811185e-02
## 1.9   100         10  50 1.576709e-01
                  …
## 18    200         30 100 4.655720e-03
## 18.1  200         30 100 9.155604e-03
## 18.2  200         30 100 7.904275e-02
## 18.3  200         30 100 6.980651e-02
## 18.4  200         30 100 4.588371e-02
## 18.5  200         30 100 6.476568e-02
## 18.6  200         30 100 6.849110e-03
## 18.7  200         30 100 3.409384e-03
## 18.8  200         30 100 2.076897e-03
## 18.9  200         30 100 1.731442e-03
```

Disadvantages of this approach

Already good solutions (DeclareDesign, MonteCarlo, …)

- Most important pieces (model specification, definitions) are hidden
- What if there's an error?
- What if we want parallel processing?

Not yet implemented…

- **What if we want this to be easier?**

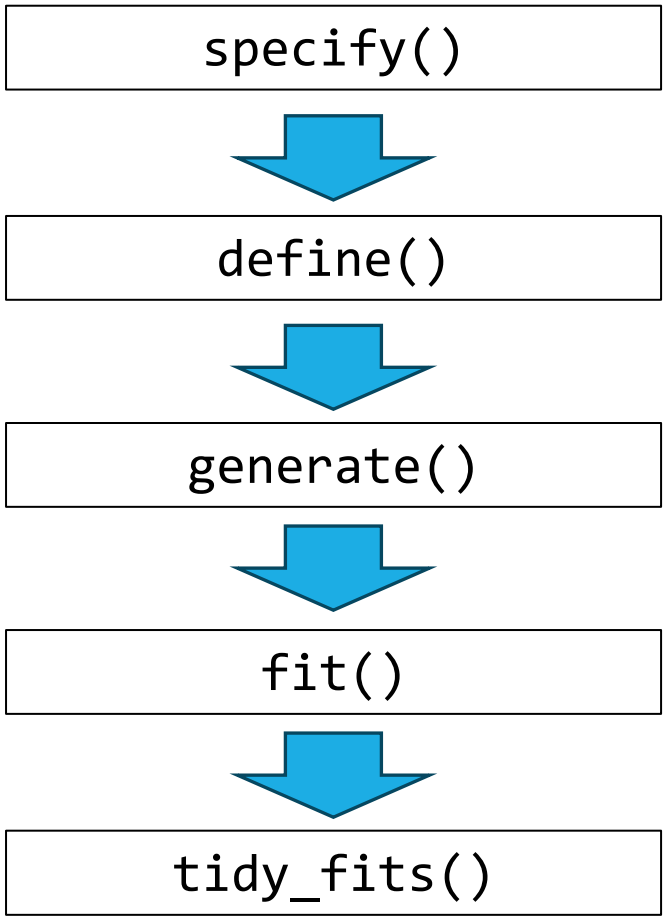
simpr does the housekeeping. *You* focus on your model!

statisfactions.github.com/simpr

INTRODUCING SIMPR

What is simpr?

- provides a grammar and a workflow for simulating data
- Targeted to intermediate R users, especially those who use the 'tidyverse' ecosystem
- Makes simulation code substantially easier



Source: https://www.travelpostersonline.com/vietnam-propaganda-poster-tomorrow-starts-today-7292-p.asp

statisfactions.github.com/simpr

THE SIMPR WORKFLOW

The simpr workflow, inspired by the infer package, distills a simulation study into five primary steps:

- specify() your data-generating process
- define() parameters that you want to systematically vary across your simulation design (e.g. *n*, effect size)
- generate() the simulation data
- fit() models to your data (e.g. lm())
- tidy_fits() for consolidating results using broom::tidy(), such as computing power or Type I Error rates

statisfactions.github.com/simpr

```r
## Specify pre and post scores that differ by a given amount
specify(pre = ~ rnorm(n, 0, sd),
                        post = ~ pre + rnorm(n, mean_diff, sd)) %>%
```

```r
## Specify pre and post scores that differ by a given amount
specify(pre = ~ rnorm(n, 0, sd),
                        post = ~ pre + rnorm(n, mean_diff, sd)) %>%
  ## Define parameters that can be varied
  define(n = 100,
         mean_diff = 10,
         sd = 50) %>%
```

```r
## Specify pre and post scores that differ by a given amount
specify(pre = ~ rnorm(n, 0, sd),
                        post = ~ pre + rnorm(n, mean_diff, sd)) %>%
  ## Define parameters that can be varied
  define(n = 100,
         mean_diff = 10,
         sd = 50) %>%
  ## Generate datasets
  generate(100) %>%
```

```r
## Specify pre and post scores that differ by a given amount
specify(pre = ~ rnorm(n, 0, sd),
                        post = ~ pre + rnorm(n, mean_diff, sd)) %>%
  ## Define parameters that can be varied
  define(n = 100,
         mean_diff = 10,
         sd = 50) %>%
  ## Generate datasets
  generate(100) %>%
  ## Fit datasets
  fit(t = ~t.test(post, pre, paired = TRUE)) %>%
```

```r
## Specify pre and post scores that differ by a given amount
specify(pre = ~ rnorm(n, 0, sd),
                        post = ~ pre + rnorm(n, mean_diff, sd)) %>%
  ## Define parameters that can be varied
  define(n = 100,
         mean_diff = 10,
         sd = 50) %>%
  ## Generate datasets
  generate(100) %>%
  ## Fit datasets
  fit(t = ~t.test(post, pre, paired = TRUE)) %>%
  ## Collect results
  tidy_fits()
```

statisfactions.github.com/simpr

And, that's all!

```
## # A tibble: 100 × 14
##    .sim_id     n mean_diff    sd   rep Source estimate statistic    p.value
##      <int> <dbl>     <dbl> <dbl> <int> <chr>     <dbl>     <dbl>      <dbl>
##  1       1   100        10    50     1 t          12.1      2.26 0.0258
##  2       2   100        10    50     2 t          14.5      3.00 0.00337
##  3       3   100        10    50     3 t          13.5      3.02 0.00326
##  4       4   100        10    50     4 t          11.9      2.18 0.0319
##  5       5   100        10    50     5 t          24.2      4.71 0.00000811
##  6       6   100        10    50     6 t          17.7      3.79 0.000256
##  7       7   100        10    50     7 t          11.7      2.32 0.0222
##  8       8   100        10    50     8 t          11.1      2.35 0.0206
##  9       9   100        10    50     9 t          15.0      2.83 0.00569
## 10      10   100        10    50    10 t           5.11     1.02 0.312
## # ℹ 90 more rows
## # ℹ 5 more variables: parameter <dbl>, conf.low <dbl>, conf.high <dbl>,
## #   method <chr>, alternative <chr>
```

statisfactions.github.com/simpr

VARYING PARAMETERS

```
## Specify pre and post scores that differ by a given amount
sim_vary = specify(pre = ~ rnorm(n, 0, sd),
                   post = ~ pre + rnorm(n, mean_diff, sd)) %>%
  ## Define parameters and vary them
  define(n = c(100, 150, 200),
         mean_diff = c(10, 20, 30),
         sd = c(50, 100)) %>%
  ## Generate datasets
  generate(1000, .progress = TRUE) %>%
  ## Fit datasets
  fit(t = ~t.test(post, pre, paired = TRUE)) %>%
  ## Collect results
  tidy_fits()
```

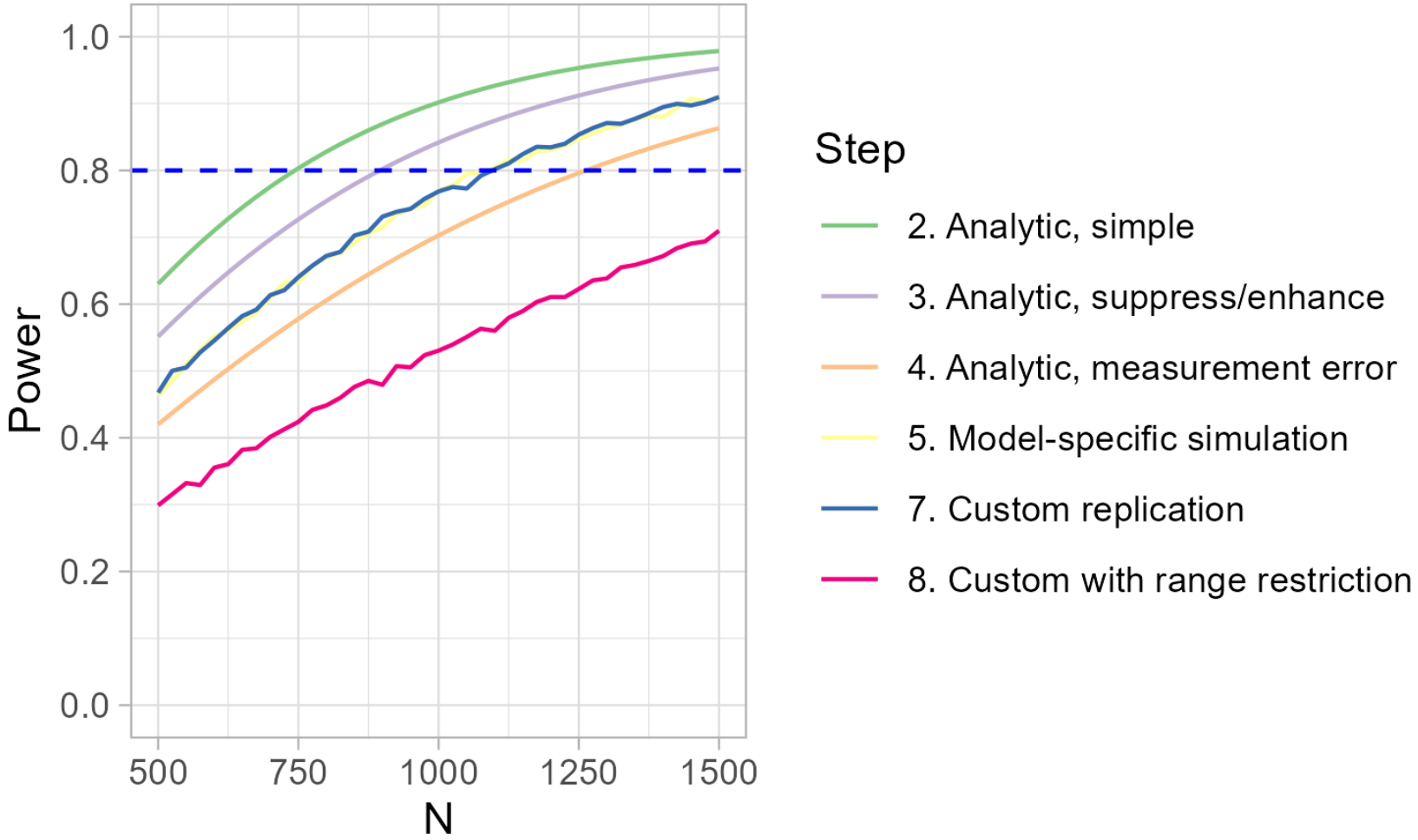
statisfactions.github.com/simpr

SUMMARIZING RESULTS WITH THE TIDYVERSE

```
sim_vary %>%
  group_by(n, mean_diff, sd) %>%
  summarize(Power = mean(p.value > 0.05))
```

```
## # A tibble: 18 × 4
## # Groups:   n, mean_diff [9]
##       n mean_diff    sd Power
##   <dbl>     <dbl> <dbl> <dbl>
##  1   100        10    50 0.507
##  2   100        10   100 0.831
##  3   100        20    50 0.026
##  4   100        20   100 0.482
##  5   100        30    50 0
##  6   100        30   100 0.16
##  7   150        10    50 0.336
##  8   150        10   100 0.791
##  9   150        20    50 0.002
## 10   150        20   100 0.303
## 11   150        30    50 0
## 12   150        30   100 0.041
## 13   200        10    50 0.212
## 14   200        10   100 0.684
## 15   200        20    50 0.001
## 16   200        20   100 0.219
## 17   200        30    50 0
## 18   200        30   100 0.016
```

EXAMPLE: POWER ANALYSIS OF REGRESSION INTERACTIONS



SIMPR PHILOSOPHY

- Custom functions not required
- Minimize overhead code
- **Get people simulating as fast as possible!**

statisfactions.github.com/simpr

ADVANCED FEATURES OF SIMPR

- More sophisticated specifications
- Data munging capabilities
- Sensibly handle errors
- Reproducible workflows
- Easy-to-use parallel processing

statisfactions.github.com/simpr

DATA MUNGING CAPABILITIES

```
range_sim = specify(pre = ~ rnorm(n, 0, sd),
                                post = ~ pre + rnorm(n, mean_diff, sd)) %>%
  ## Define parameters and vary them
  define(n = c(100, 150, 200),
         mean_diff = c(10, 20, 30),
         sd = c(50, 100)) %>%
  ## Generate datasets
  generate(1000, .progress = TRUE) %>%
  ## Apply tidyverse functions to every simulation dataset
  per_sim() %>%
  ## Mutate to add a range restriction
  mutate(across(everything(), case_when(
    pre > 100 ~ 100,
    pre < -100 ~ -100,
    .default ~ pre))) %>%
  ## Fit datasets
  fit(t = ~t.test(post, pre, paired = TRUE)) %>%
  ## Collect results
  tidy_fits()
```

statisfactions.github.com/simpr

ERROR HANDLING

- Can change error handling to keep going with simulation, stop simulation, or to skip warnings
- Debug and recovery options to enter into simulation during error

REPRODUCIBLE WORKFLOWS

- Same seed, same results
- Can regenerate *just a specific subset* to see what happened in that particular dataset or fit
- Useful in debugging and diagnosing unexpected results, etc.

statisfactions.github.com/simpr

REPRODUCIBLE WORKFLOWS

Filtering Full Simulation

```
set.seed(500)

specify(a = ~ runif(6)) %>%
  generate(3) %>%
  filter(.sim_id == 3)
```

```
## full tibble
## -------------------------
## # A tibble: 1 × 3
##   .sim_id   rep sim
##     <int> <int> <list>
## 1       3     3 <tibble [6 × 1]>
##
## sim[[1]]
## -------------------------
## # A tibble: 6 × 1
##       a
##   <dbl>
## 1 0.371
## 2 0.959
## 3 0.633
## 4 0.177
## 5 0.803
## 6 0.133
```

Simulate Subset Only

```
set.seed(500)

specify(a = ~ runif(6)) %>%
  generate(3, .sim_id == 3)
```

```
## full tibble
## -------------------------
## # A tibble: 1 × 3
##   .sim_id   rep sim
##     <int> <int> <list>
## 1       3     3 <tibble [6 × 1]>
##
## sim[[1]]
## -------------------------
## # A tibble: 6 × 1
##       a
##   <dbl>
## 1 0.371
## 2 0.959
## 3 0.633
## 4 0.177
## 5 0.803
## 6 0.133
```

statisfactions.github.com/simpr

PARALLEL PROCESSING

```
library(simpr)
library(tidyverse)
library(future)

plan(multisession, workers = 6) # or however many cores are reasonable to use

## Specify pre and post scores that differ by a given amount
sim_vary = specify(pre = ~ rnorm(n, 0, sd),
                   post = ~ pre + rnorm(n, mean_diff, sd)) %>%
  ## Define parameters and vary them
  define(n = c(100, 150, 200),
         mean_diff = c(10, 20, 30),
         sd = c(50, 100)) %>%
  ## Fit datasets
  fit(t = ~t.test(post, pre, paired = TRUE)) %>%
  ## Collect results
  tidy_fits() %>%
  ## Generate datasets (and perform post-processing)
  generate(1000, .progress = TRUE)
```

statisfactions.github.com/simpr

Pros and Cons of simpr vs. other solutions

- ✓ tidyverse friendly
- ✓ Beginner friendly
- ✓ Reproducibility, error handling built in
- ✓ General-purpose, customizable and can handle arbitrary R code

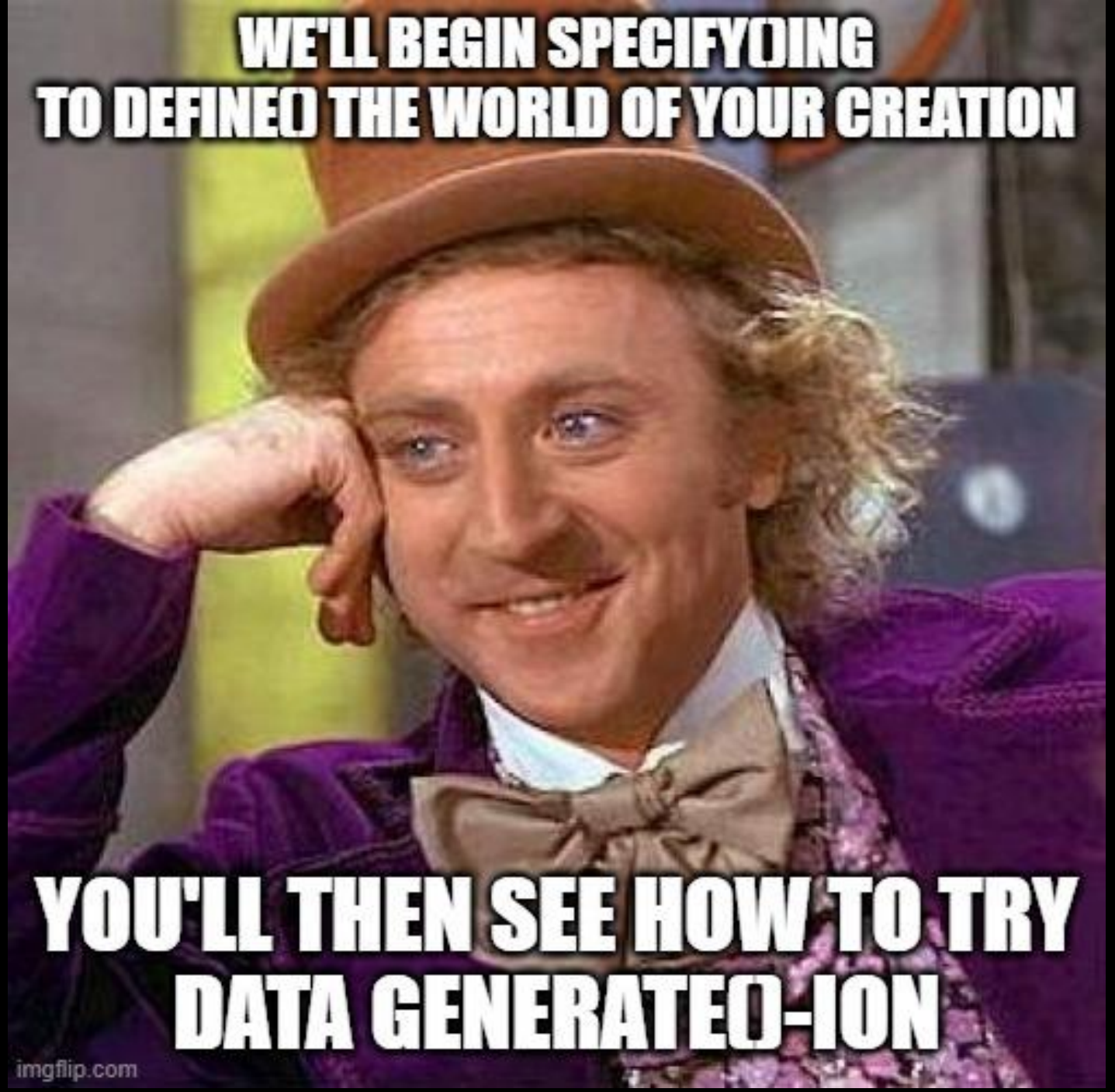
- × Likely not as fast/optimized as some alternatives
- × Not as customizable/powerful as DeclareDesign
- × Not specifically set up for any particular application (no MC errors, plots, reports, specific models…)

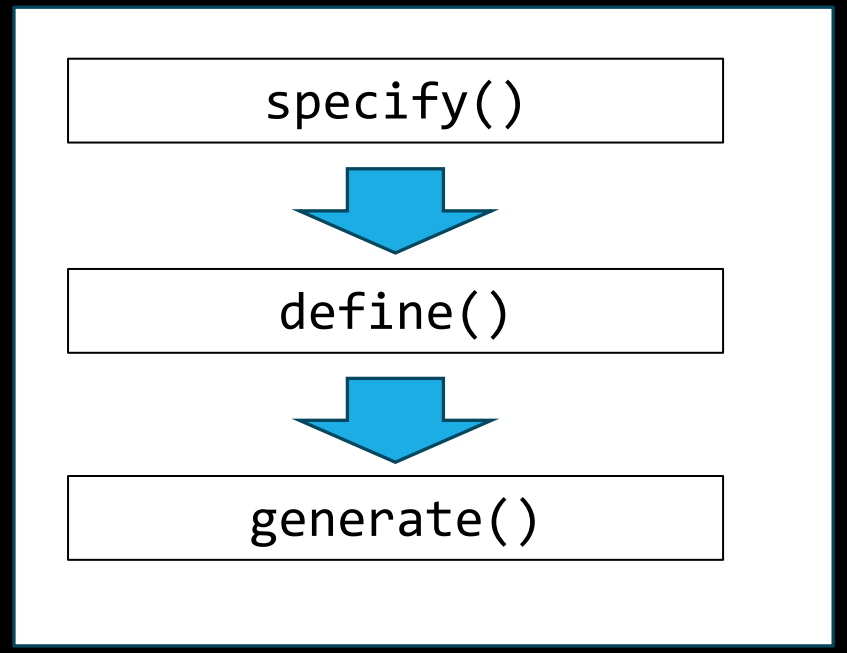
statisfactions.github.com/simpr

Future Directions

- “Chunking”/options for autosaving intermediate results to disk
- Model-specific support for common design analyses (use DesignLibrary?)
- Test coverage! 🐜🐜🐜

statisfactions.github.com/simpr


WE'LL BEGIN SPECIFY()ING
TO DEFINE() THE WORLD OF YOUR CREATION
YOU'LL THEN SEE HOW TO TRY
DATA GENERATE()-ION
imgflip.com

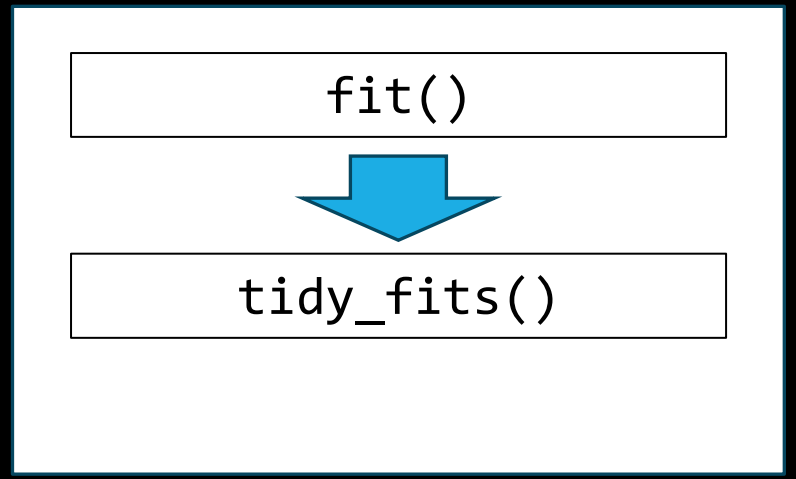

specify()
define()
generate()


```r
library(simpr)
library(tidyverse)
library(future)

plan(multisession, workers = 6) # or however
many cores are reasonable to use
```

```r
set.seed(500)

specify(a = ~ runif(6)) %>%
  generate(3, .sim_id == 3)
```



THERE IS NO WAY, I KNOW TO COMPARE
WITH SIMPR SIMULATION
CODING THERE, YOU'LL BE FREE
IF YOU TRULY WISH TO BE

- **Algorithm Optimization**
  - CAISEr determines the sample size required for comparing the performance of a set of *k* algorithms on a given problem instance when power inaccuracy is predefined
- **ANCOVA**
  - pwr2ppl computes power for one or two factor ANCOVA with a single covariate
- **ANOVA:**
  - General options include pwr (one-way ANOVA), pwr2 & pwr2ppl (up to two-way ANOVA) and WebPower (up to two-way and repeated measures). The easypower package, based on the pwr package, simplifies the user input for factorial ANOVA.
  - BUCSS allows any number of factors, using uncertainty and publication bias correction.
  - powerbydesign provides functions for bootstrapping the power of ANOVA designs based on estimated means and standard deviations of the conditions
  - powerAnalysis only should be used with balanced one-way analysis of variance tests.
  - powerMediation performs power calculation for interaction effect in 2x2 two-way ANOVA.
  - Superpower uses simulations and analytic power solutions for ANOVA designs of up to three factors to calculate power and average observed effect sizes.
- **Bayesian:**
  - bayescount provides analysis and power calculations for count data.
  - BayesESS determines effective sample size of a parametric prior distribution in Bayesian conjugate models (beta-binomial, gamma-exponential, gamma-Poisson, dirichlet-multinomial, normal-normal, inverse chi-squared-normal, inverse-gamma-normal), Bayesian linear and logistic regression models. Bayesian continual reassessment method (CRM), and Bayesian time to event model.
  - BayesianPower determine the required samp… factor
  - SampleSizeMeans calculates sample size re… proportions, using three different Bayesian … sample sizes for the Average Length Criteric… both the fully Bayesian and the mixed Bayes…
- **Beta Distribution:**
  - PASSED and BetaPASS helps find the power… outcome/response/dependent variable is a p… functions to plot power curves demonstrate…
- **Bioequivalence Study**
  - Power2Stage contains functions to obtain th… via simulations.
  - PowerTOST calculates power and sample siz…
- **Case-Control study:**
  - CoRpower calculates power for assessment … in clinical efficacy trials. The methods differ … and in biomarker response subgroups, whic… of efficacy/protection.
  - epiR calculates sample size, power, and det…
  - samplesizelogisticcasecontrol determines s…
- **Chi-squared test:**
  - Options include powerAnalysis, pwr, pwr2ppl, bimetallic, ssd.
- **Cochran-Mantel-Haenszel Test:**
  - samplesizeCMH calculates the power and sample size for Cochran-Mantel-Haenszel tests, with several helper functions for working with probability, odds, relative risk, and odds ratio values.
- **Competing Risks Analysis:**
  - powerCompRisk is power analysis tool for jointly testing the cause-1 cause-specific hazard and the any-cause hazard with competing risks data.
- **Complex Surveys**

- **Diagnostic Test**
  - MKpower computes sample size, power, delta, or significance level of a diagnostic test for an expected sensitivity or specificity.
- **Dirichlet-Multinomial distribution:**
  - HMP uses the Dirichlet-Multinomial distribution to provide several functions for formal hypothesis testing, power and sample size calculations.
- **Factorial Design**
  - BDEsize calculates the sample size required to detect a certain standardized effect size, under a significance level (two-level fractional factorial, randomized complete block design, full factorial design, and split-plot design). This package also provides three graphs; detectable standardized effect size vs power, sample size vs detectable standardized effect size, and sample size vs power, which show the mutual relationship between the sample size, power and the detectable standardized effect size.
  - H2x2Factorial estimates the required number of clusters or the achieved power level under different types of hypothesis tests in a hierarchical 2x2 factorial trial with unequal cluster sizes and a continuous outcome.
- **Fisher's Test**
  - Power calculations for differences between binomial proportions can be achieved with Exact for unconditional exact tests with 2x2 contingency tables. MIDN computes the exact sample sizes required based on the Boschloo's technique and Fisher-Boschloo test mid-N estimates.
  - ssanv provides a calibrated power calculation by leveraging the uncertainty in either nonadherence or parameter estimation.

- **Hierarchical Data**
  - HierO calculates statistical power for given type I error (alpha), effect size (Delta) and non-centrality parameter (ncpar) of a non-central chi-square distribution.
- **High Dimensional Classification Study**
  - HDDesign determines the sample size requirement to achieve the target probability of correct classification (PCC) for studies employing high-dimensional features.
- **Human Microbiome Experiment**
  - HMP uses the Dirichlet-Multinomial distribution to provide several functions for formal hypothesis testing, power and sample size calculations for human microbiome experiments

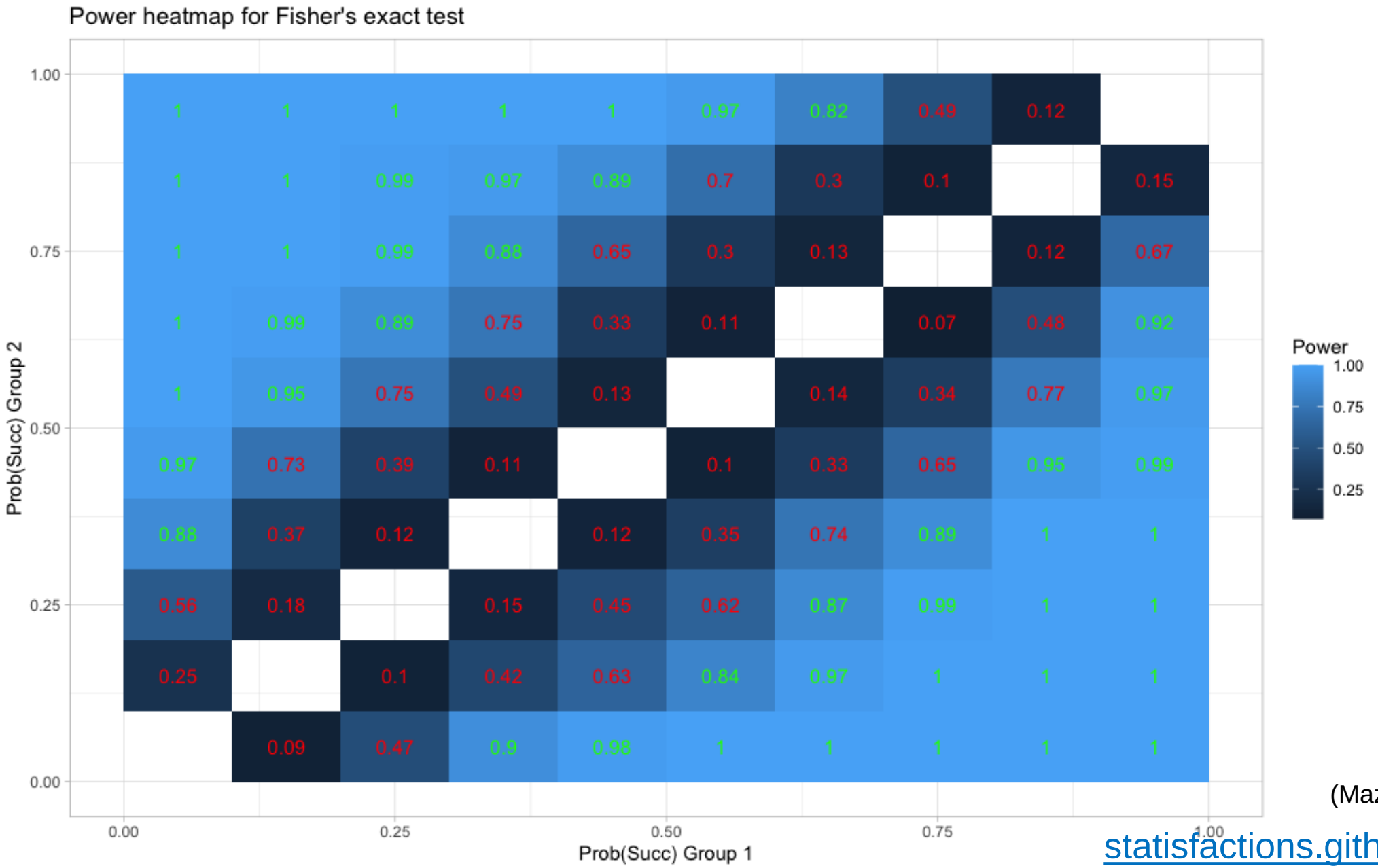
- **Intraclass Correlation**
  - ICC.Sample.Size calculates power for given value of p, the null hypothesis p0, number of raters (k), num… and alpha. Can also be used to calculate the effect of increasing N at given intervals to a maximum N, or… increase in sample size to obtain increasing power with a given maximum N.
- **Interobserver Agreement Studies**
  - kappaSize providesbasic tools for sample size estimation in studies of interob-server/interrater agreeme… functions for both the power-based and confidence interval-based methods, with binary or multinomial … through six raters.
- **Likelihood Ratio test**
  - asypow calculates power utilizing asymptotic likelihood ratio methods
- **Linear Regression**
  - pwr, powerMediation and WebPower provide analytical power calculations.
  - simr and simpr provided simulated power calculations.
  - BayesESS provided simulated power calculations for Bayesian models
- **Local Average Treatment Effect (LATE)**
  - powerLATE is an implementation of the generalized power analysis for the local average treatment effect… uses standardized effect sizes to place a conservative bound on the power under minimal assumptions. … to recover power, sample size requirements, or minimum detectable effect sizes. Package also allows us… bounds, and …
- **MANOVA**
  - pwr2ppl supports power for one factor MANOVA with up to 2 levels and 4 measures.
- **Mcnemar Test**
  - MIDN computes the exact sample sizes required in the randomized UMPU test and its conservative nonr… counterpart for attaining prespecified power. However, in contrast to the parallel group setting, the midp… between these two numbers shall now be used as a nearly exact value of the number of pairs to be obse… test based on the score-statistic corrected for possible exceedances of the nominal significance level.
- **Mediation Analysis (see also Structural Equation Modeling)**
  - pwr2ppl and WebPower both support power analysis of mediation. WebPower additionally supports stati…

Relatedly: Message me if you're interested in helping with a Power Analysis Task view! 😟😟😟😟😟

Github/X: @statisfactions

https://github.com/statisfactions/ctv-power

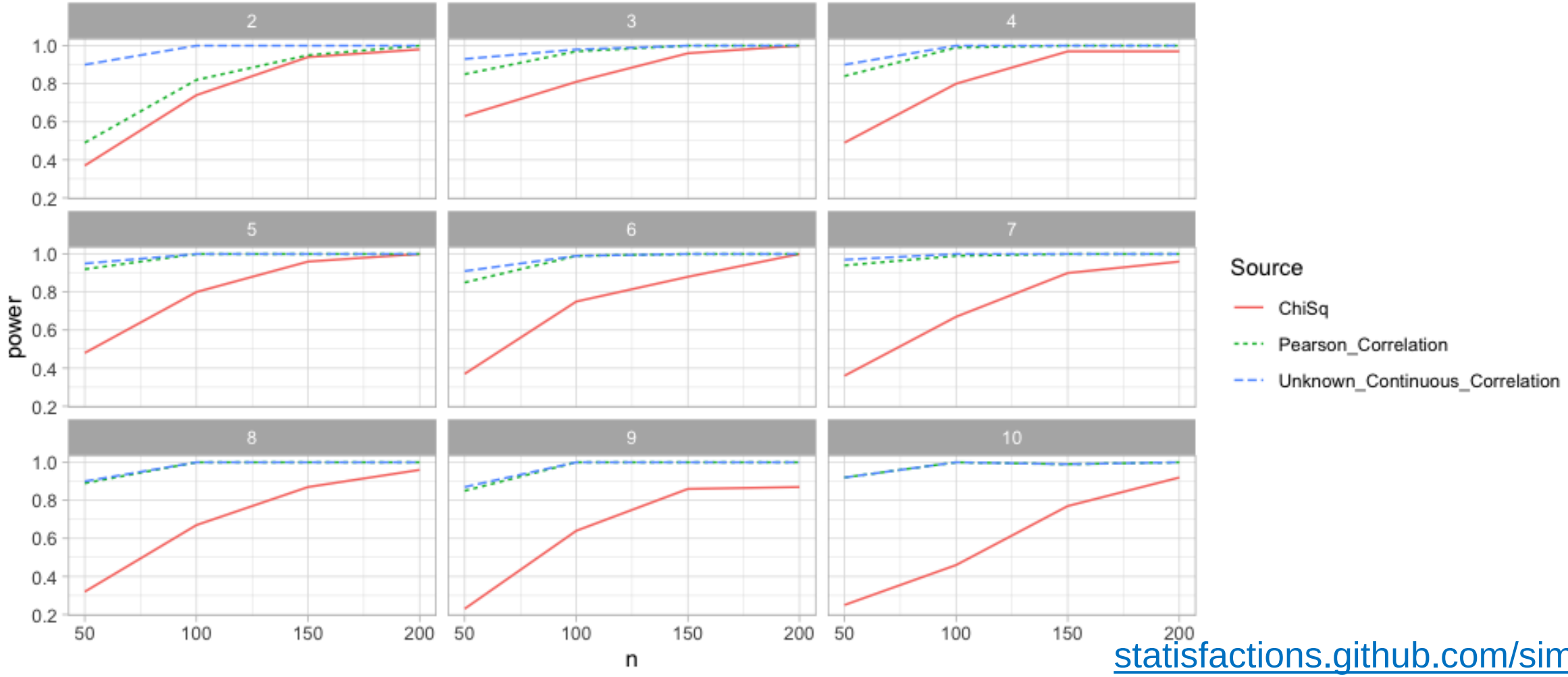
STUDY PLANNING: POWER FOR A FISHER EXACT TEST



(Mazzocco et al., 2020)

statisfactions.github.com/simpr

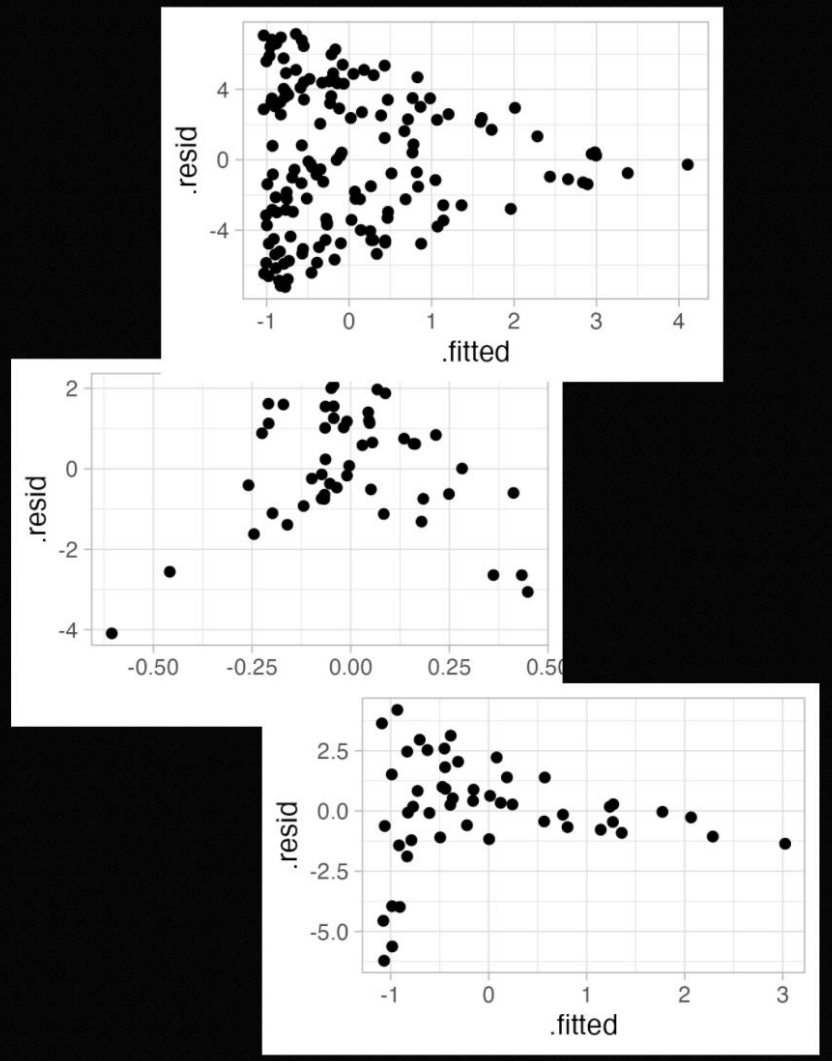
METHODOLOGICAL RESEARCH: EFFECT OF THE NUMBER OF CATEGORIES ON POWER

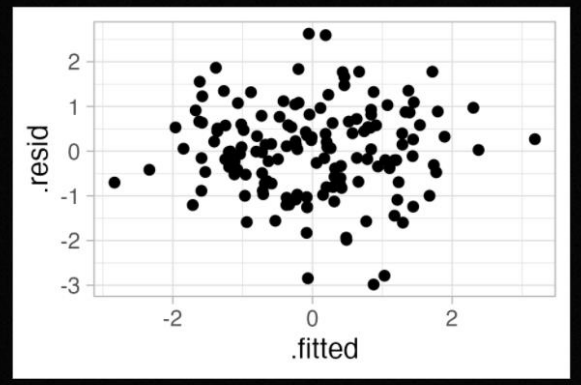


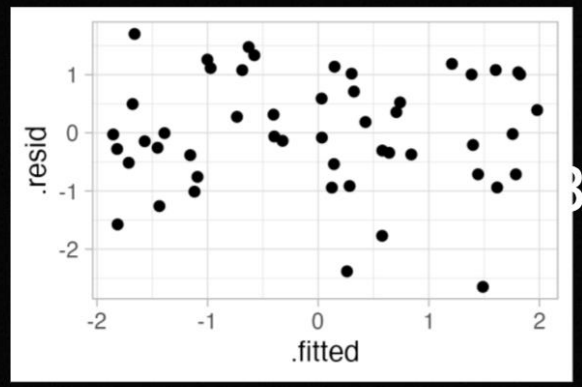
statisfactions.github.com/simpr

Linearity 🤔

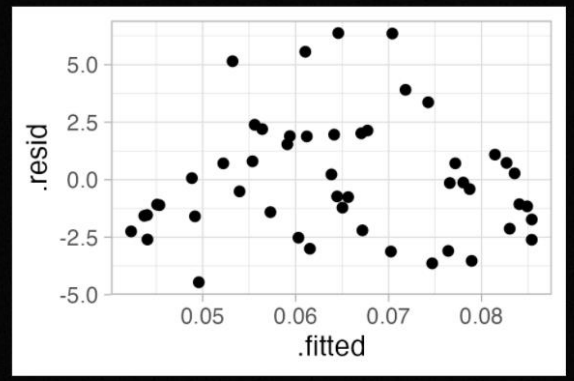
Worst







Best



GENERATING EXAMPLES FOR TEACHING: REGRESSION ASSUMPTIONS

statisfactions.github.com