Simulating Data

What are simulation studies?

- Use of artificial data to investigate problems related to:
  - study design
  - analytic approach
  - implications of research findings

Why learn to simulate data

- Very useful tool in the toolbox
  - study design, what-if analysis, utility analysis
- Gives you a new (and better) understanding of statistics

Random Number Generators (RNG)

- Functions that generate (pseudo-) random numbers
- Typically based on a probability distribution
- A mathematical function that quantifies the probability of some random variable, X given a set of parameters

Random Number Generators (RNG)

- `rnorm` function
- draws n random samples from a normal distribution
  - with some mean and standard deviation
- has 3 arguments
  - a sample size `n`, a `mean` and `sd` of the distribution

```
y <- rnorm(n = 100, mean = 10, sd = 2.5)
m <- mean(y)
s <- sd(y)

print(c("sample mean" = m, "sample sd" = s))
```

```
sample mean   sample sd
   9.454534    2.238740
```

Random Number Generators (RNG)

- Give us a unique set of numbers each time
- Sometimes we want replicability
  - demonstrations
  - replicating results of a simulation
- "Set the seed"
  - `set.seed()`

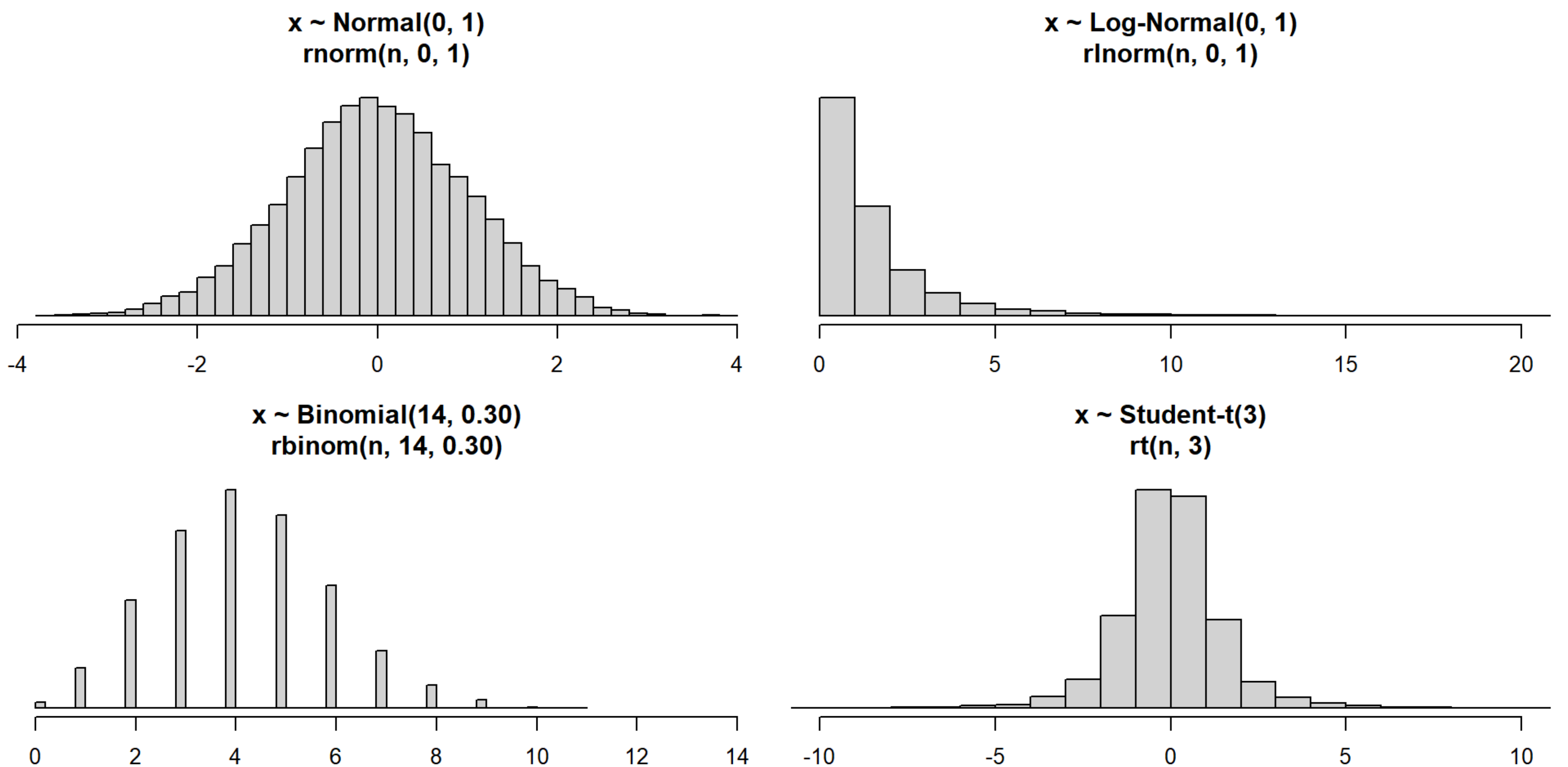
Random Number Generators (RNG)

```r
set.seed(1)
y1 <- rnorm(n = 100, mean = 0, sd = 1)
mean_1 <- mean(y1)

set.seed(1)
y2 <- rnorm(n = 100, mean = 0, sd = 1)
mean_2 <- mean(y2)

print(c("mean 1" = mean_1, "mean 2" = mean_2))
```

```
   mean 1    mean 2 
0.1088874 0.1088874 
```

Random Number Generators (RNG)

- Many types of distributions available in R



Distribution	R function	Common Uses
Normal	`rnorm`	noise/error, performance scores
Log-Normal	`rlnorm`	response times, anything that is strictly positive
Exponential	`rexp`	short response times, distances
Gamma	`rgamma`	response times, anything that is strictly positive

Distribution	R function	Common Uses
Binomial/Bernoulli	`rbinom`	number of successes out of k attempts
Poisson	`rpois`	count data, number of events in a given time
Beta	`rbeta`	proportions or percentages
Multinomial	`sample`	categorical data, simulating group assignments

A general framework for simulations in R

- Define your question or goal
  - power analysis, what-if analysis
- Define the generative model and generator
- Define the simulation and simulator
- Define the simulation conditions
  - e.g. sample sizes, test lengths, number of predictors
- Define the summary statistics

Example 1: Power Analysis

Define the goal

- What sample size do I need for a 1-sample t-test?
  - Population Mean: $\mu = 0$
  - Power: $1 - \beta = 0.80$ (20% false negative rate)
  - Observed Standard Deviation: $s = 1$
  - Minimum Effect Size: $d = \frac{\bar{x}-\mu}{s} = \frac{\bar{x}-0}{1} = 0.20$

Define the generative model

- How are the data going to be created?
  - What are the variables involved?
  - What distributions should they be sampled from?
  - Are the parameters fixed (i.e. chosen by you beforehand)?
  - Are the parameters randomly generated as well?
- What needs to be returned by the generator?
  - Just the data?
  - the parameters too?

Define the generative model

$$\mu = 0$$

$$s = 1$$

$$\mu_x = \frac{d - \mu}{s} = 0.20$$

$$x \sim \mathrm{Normal}(\mu_x, s)$$

Define the generative model

$$\mu = 0$$

$$s = 1$$

$$\mu_x = \frac{d - \mu}{s} = 0.20$$

$$x \sim \text{Normal}(\mu_x, s)$$

```
n <- 100

d <- 0.20
mu <- 0
s <- 1
mu_x <- (d - mu) / s

x <- rnorm(n, mu_x, s)

data <- data.frame(x = x)
```

Define the generative model

```r
generator <- function(n_persons) {
  d <- 0.20
  mu <- 0
  s <- 1
  mu_x <- (d - mu) / s

  x <- rnorm(n_persons, mu_x, s)

  data <- data.frame(x = x)

  return(data)
}
```

Define simulation conditions

- What conditions should affect our simulation?
    - sample size, test-length, effect sizes, careless responder %, etc.
- Sample sizes: from 5 to 50
- Simulation iterations: 1,000

```
simulation_conditions <- data.frame(n_persons = 5:500)

n_iters <- 1000
```

Define the simulation

- What sort of analysis are you investigating?
- t-test

```r
simulator <- function(conditions) {
  n_persons <- conditions[["n_persons"]]
  data <- generator(n_persons)

  fit <- t.test(x = data$x, mu=0)
  p_val <- fit$p.value

  output <- data.frame(n_persons=n_persons, p_val=p_val)

  return(output)
}
```

Define the summary statistics

- Simulations generate a lot of data
- We are typically interested in facts about the data and parameters rather than the data and parameters themselves

Common summary statistics

- Mean Square Error (MSE) of Prediction
  - lower = better prediction
  - $$\text{MSE} = \frac{1}{n}\sum(b - \hat{b})^2$$
- Mean Bias
  - how far is the estimate from the true value on average?
  - $$\text{Bias} = \frac{1}{n}\sum b - \hat{b}$$

Common summary statistics

- Coverage
  - do 95% confidence intervals contain the data generating parameter?
  - $\frac{1}{n}\sum_i^n \delta(-1.98 \cdot \mathrm{SE} < b < 1.98 \cdot \mathrm{SE})$
- Power: $1 - \beta \approx \frac{1}{n}\sum_i^n \delta(p < \alpha)$

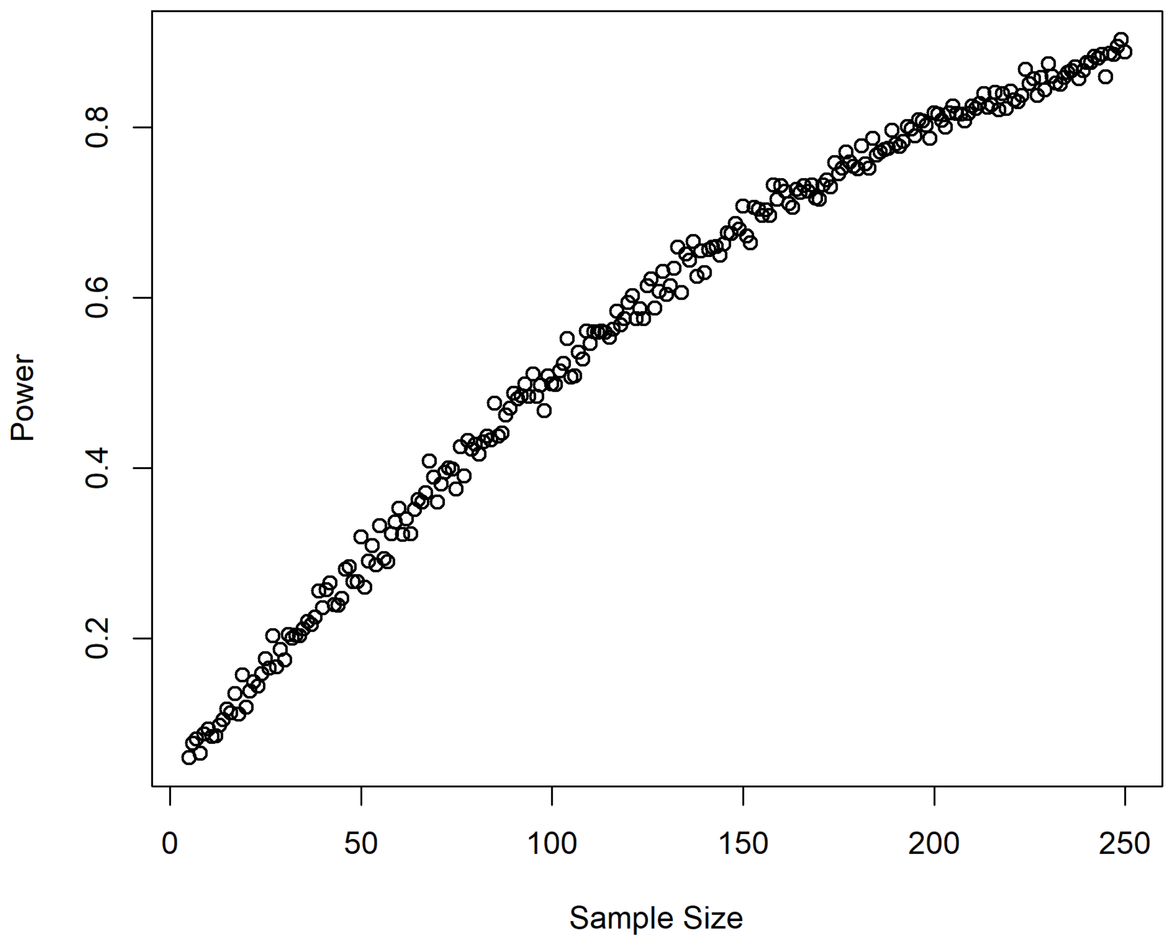
```
calculate_power <- function(p_values, alpha=0.05) {
  mean(p_values < alpha)
}
```

Roll it into a simulation

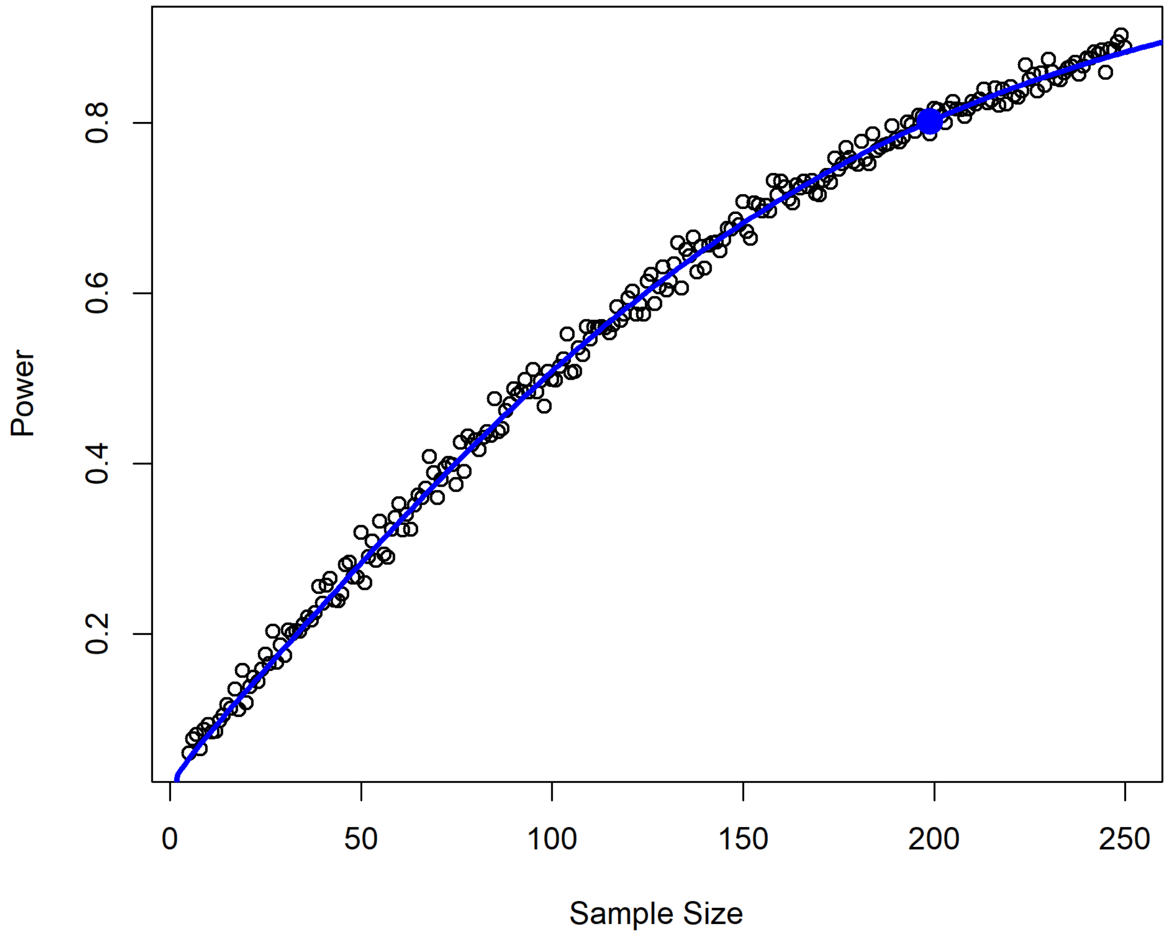
- Create a for-loop

```
# create an empty data.frame to store simulated p-values
results <- data.frame()
n_conditions <- nrow(simulation_conditions)

# loop over conditions
for (c in 1:n_conditions) {
  conditions <- simulation_conditions[c,,drop=FALSE]
  # for some number of iterations
  for (i in 1:n_iters) {
    new_result <- simulator(conditions)

    results <- rbind(results, new_result)

    # print progress every 100 iterations
    if (i %% 100 == 0) {
      cat("Sample Size: ", conditions$n_persons, "; Iteration: ", i
    }
  }
}
```

Review Results



Review Results



Caveats

- Using RNGs introduces some variability
  - solutions are not exact
  - need lots of iterations
- Can be computationally intensive
  - takes time and computing power
  - running simulations across cores in parallel helps
- Answers are only as thorough as your simulation conditions
  - Do you need to simulate all possible scenarios?

Resources

- Hallgren, K. A. (2013). Conducting Simulation Studies in the R Programming Environment. *Tutorials in Quantitative Methods for Psychology*, *9*(2), 43–60. https://doi.org/10.20982/tqmp.09.2.p043
- Morris, T. P., White, I. R., & Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, *38*(11), 2074–2102. https://doi.org/10.1002/sim.8086

https://distribution-explorer.github.io/index.html

https://github.com/tylerjamesryan/Simulation