COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

GR5206: lecture 10

Computational Statistics
And Introduction to Data Science

Thibault Vatter

Department of Statistics, Columbia University

11/08/2019

Today

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

- **Random number generation:**
  - Random numbers in R.
  - The linear congruential generator.
- **Simulation:**
  - Simulating random variables in R.
  - The inverse transforms method.
  - The acceptance-rejection method.
- **The bootstrap:**
  - How to approximate the sampling distribution.
- **Monte Carlo integration:**
  - How to use simulation to approximate integrals.
  - Variance reduction techniques:
    - Antithetic variates.
    - Control variates.
    - Importance sampling.

Outline

Outline

Random number generation

- How does R produce random numbers?
  - It doesn't!
  - It generate **pseudorandom numbers** that are indistinguishable from real random numbers.
    - (if you don't know how it started)

Linear Congruential Generator

- **Linear Congruential Generator (LCG):**
  - Produces a sequence of pseudorandom numbers based on the following recurrence:

$$X_n = (aX_{n-1} + c) \mod m$$

  - **Pseudorandom** because after a while, the sequence will repeat.
  - More sophisticated variants exist.

```
# Recall how modular arithmetic work
4 %% 4; 4 %% 3
#> [1] 0
#> [1] 1

# The recurrence relation
rand <- function(x, a = 5, c = 12, m = 16) (a * x + c) %% m
rand(10)
#> [1] 14
rand(rand(10))
#> [1] 2
```

Linear Congruential Generator

- A simple implementation

```
lcg_rand <- function(n, x = 10, a = 5, c = 12, m = 16) {
  rng <- vector(length = n)
  for (i in 1:n) {
    x <- rand(x, a, c, m)
    rng[i] <- x
  }
  return(rng)
}

lcg_rand(20)
#>  [1] 14  2  6 10 14  2  6 10 14  2  6 10 14  2  6 10 14  2  6 10
```

- Poor choice of parameters:
  - The generator shuffled some of the integers $0, 1, ..., m - 1 = 15$ into an "unpredictable" order.
  - Want the generator to shuffle all of these integers, but this generator only gives 4.

Linear Congruential Generator

- Better:

```
lcg_rand(20, a = 131, c = 7, m = 16)
#>  [1]  5  6  9  2 13 14  1 10  5  6  9  2 13 14  1 10  5  6  9  2
```

- Simulating uniform in $[0, 1]$
  - The X_n are between 0 and $m - 1$,.
  - The sequence repeats every m occurrences.
  - Dividing by m gives numbers in [0, 1).

```
lcg_unif <- function(n, x = as.numeric(Sys.time()) * 1000,
                     m = 2^32, a = 1103515245, c = 12345) {
  rng <- vector(length = n)
  for (i in 1:n) {
    x <- (a * x + c) %% m
    rng[i] <- x / m
  }
  return(rng)
}
lcg_unif(5)
#> [1] 0.0898 0.7930 0.0597 0.9220 0.3164
```

- ?Random to get more info!

Outline

Why do we care about simulation?

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- To **understand** a model:
  - Simulate model output.
    - Estimate accuracy and precision.
  - Simulate how a hypothesis test behaves under H_0 vs H_1.
    - Do the empirical results match the developed theory?
  - Simulate the sampling distribution of an estimator.
    - Assume some parametric model or use nonparametric methods such as the bootstrap.
- To **check** a model:
  - Simulated data from a stochastic model should resemble the real data.
- To **fit** a model:
  - Markov Chain Monte Carlo Methods (MCMC).

Sampling from known distributions

- How do we sample from a probability distribution?
- There are many ways...
- **Common distributions:** use built-in functions (normal, gamma, Poisson, binomial, etc...).
- **Uncommon distributions:** need to use simulation.
  - **Discrete distributions:** often can use `sample()`.
  - **Continuous distribution:**
    - Use the **inverse transform method** when the cdf is invertible in closed form.
    - Or the **acceptance-rejection method** otherwise.

The `sample()` function

- Use `sample()` to sample from
  - The discrete uniform distribution.
  - Uncommon discrete distributions (by specifying the probabilities).

```
sample(x, size, replace = FALSE, prob = NULL)
sample.int(n, size = n, replace = FALSE, prob = NULL)
```

- Problem: sample from the following discrete distribution.

x	1	2	3
$f(x)$	0.1	0.2	0.7

```
n <- 1000; p <- c(0.1, 0.2, 0.7)
x <- sample.int(3, size = n, prob = p, replace = TRUE)
head(x, 10)
#>  [1] 2 3 3 3 1 3 2 1 3 3
rbind(p, p.hat = table(x)/n)
#>            1     2     3
#> p      0.10 0.200 0.700
#> p.hat 0.11 0.196 0.694
```

The inverse transform method

Theorem

If X is a continuous random variable with cdf F and $U \sim U[0,1]$, then

- $F(X) \sim U[0,1]$,
- $F^{-1}(U)$ *is a continuous random variable with cdf* F.

- See next slide for the proof.
- The inverse transform sampling algorithm:
  1. Derive the inverse function F^{-1}.
     - Solve $F(x) = u$ for x to find $x = F^{-1}(u)$.
  2. Write a function to compute $x = F^{-1}(u)$.
  3. Generate a random value $u \sim U[0,1]$.
  4. Compute $x = F^{-1}(u)$.

Proof of the theorem

- Let $Z = F(X)$, then

$$\begin{aligned} P(Z \leq z) &= P(F(X) \leq z) \\ &= P(X \leq F^{-1}(z)) \\ &= F(F^{-1}(z)) \\ &= z. \end{aligned}$$

Thus,
$Z = F(X) \sim U[0, 1]$.

- Let $Y = F^{-1}(U)$, then

$$\begin{aligned} P(Y \leq y) &= P(F^{-1}(U) \leq y) \\ &= P(F(F^{-1}(U)) \leq F(y)) \\ &= P(U \leq F(y)) \\ &= F(y). \end{aligned}$$

Thus, $Y = F^{-1}(U)$ is a continuous random variable with cdf F.

An example

- Problem: sample exponential rvs (with $\lambda = 2$) using the inverse transform method.

1. The pdf $f(x) = \lambda e^{-\lambda t}$, so the cdf is

$$F(x) = \int_0^x f(t)dt = \int_0^x \lambda e^{-\lambda t} dt = 1 - e^{-\lambda x},$$

and solving $u = 1 - e^{-\lambda x}$ for u, its inverse is

$$F^{-1}(u) = -\frac{1}{\lambda} \log(1 - u).$$

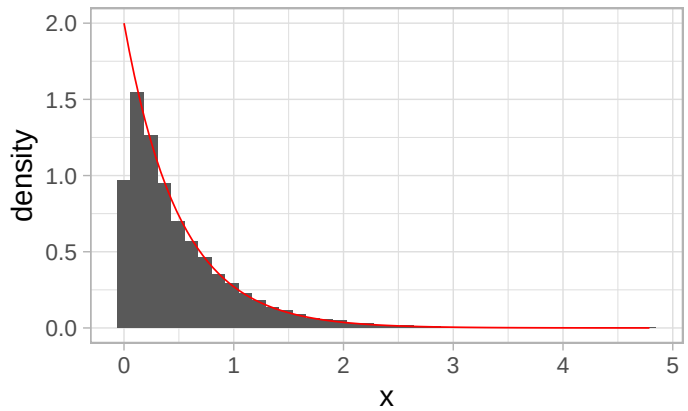
2. A simple function:

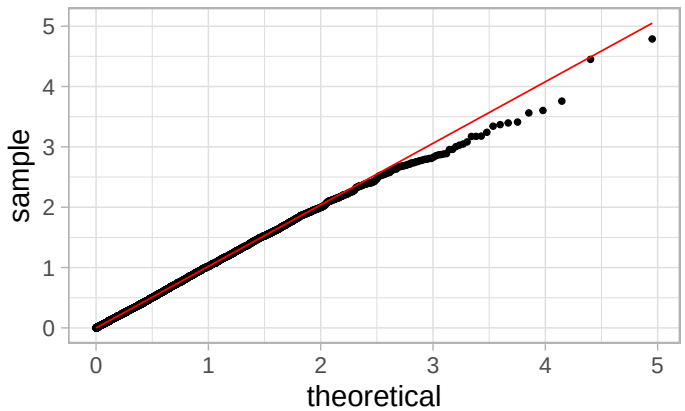
```
Finv <- function(u, lambda) -log(1 - u) / lambda
Finv(0.5, 1) == qexp(0.5, 1)
#> [1] TRUE
```

An example cont'd

3. Generate a random value $u \sim U[0, 1]$.
4. Compute $x = F^{-1}(u)$.

```
df <- tibble(u = runif(1e4), x = Finv(u, 2))
ks.test(df %>% pull(x), pexp, rate = 2)$p.value
#> [1] 0.402

ggplot(df, aes(x)) +
  geom_histogram(aes(y = ..density..), bins = 40) +
  stat_function(fun = function(x) dexp(x, rate = 2), col = "red")

ggplot(df, aes(sample = x)) +
geom_qq(distribution = qexp, dparams = list(rate = 2)) +
stat_qq_line(distribution = qexp, dparams = list(rate = 2), color = "red")
```





What if the F exists but not F^{-1}?

- Let $f(x) = x\sin(x)/2\pi$ for $x \in [-\pi, pi]$, hence

$$F(x) = \int_{\pi}^{x} f(x)dx = (\sin(x) - x\cos(x) + \pi)/2\pi.$$

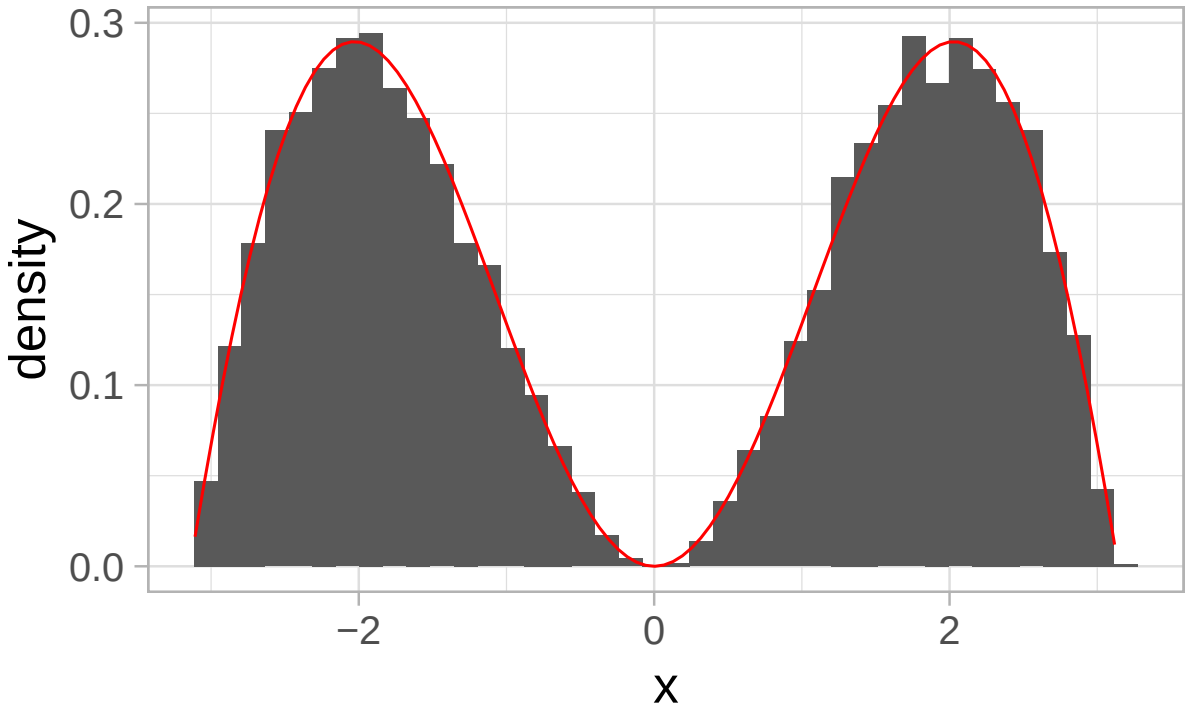
- F^{-1} does not admit a closed form expression... so all is lost?
- No! One can use numerical inversion!

```
F_not_nice <- function(x) (sin(x) - x * cos(x) + pi) / (2 * pi)
Finv <- function(u) uniroot(function(x) F_not_nice(x) - u,
                            lower = -pi, upper = pi)$root
Finv(F_not_nice(0))
#> [1] 0
Finv(F_not_nice(1))
#> [1] 1
```

What if the F exists but not F^{-1}?

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

```
df <- tibble(u = runif(1e4),
             x = map_dbl(u, Finv))

ggplot(df, aes(x)) +
  geom_histogram(aes(y = ..density..), bins = 40) +
  stat_function(fun = function(x) x * sin(x) / (2 * pi), col = "red")
```

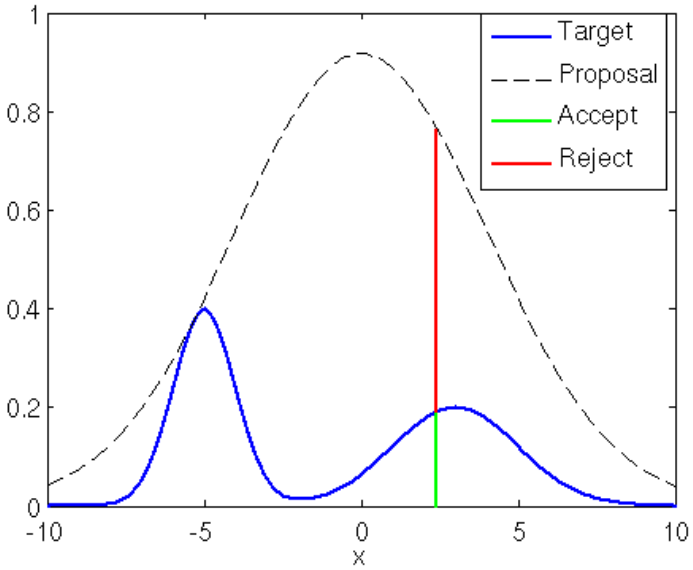


The acceptance-rejection method

- The inverse transform method requires a nice invertible cdf.
- What if we only have the pdf?
- **Rejection sampling** (or acceptance-rejection sampling) obtains draws exactly from the target distribution.
- How?
  - By sampling candidates from an easier distribution.
  - An then correcting the sample by randomly rejecting some candidates.

The theory

- Let
  - $X \sim f$ where f is the target density,
  - $Y \sim g$ where g is a density satisfying $g(x) > 0$ if $f(x) > 0$.
  - $1 < M < \infty$ be such that $f(x) \leq Mg(x)$ for all values of x.
    - I.e., M is an upper bound on the likelihood ratio $f(x)/g(x)$.
    - $Mg(x)$ is called an **envelope** of $f(x)$.
- Then
  1. Draw $Y \sim g$.
  2. Draw $U \sim U[0, 1]$.
  3. If $U < f(Y)/Mg(Y)$, keep Y.
  4. Otherwise, repeat until a value is accepted.
  5. Repeat 1-4 until you have the desired sample size.

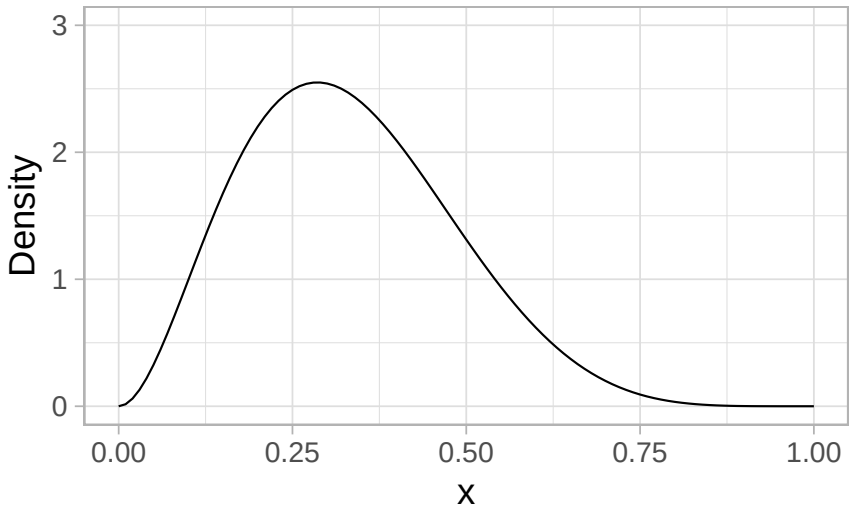


An example

- Suppose f is the beta density with parameters $a = 3$ and $b = 6$.

$$f(x) = \Gamma(a+b)/(\Gamma(a)\Gamma(b))x^{a-1}(1-x)^{b-1}$$

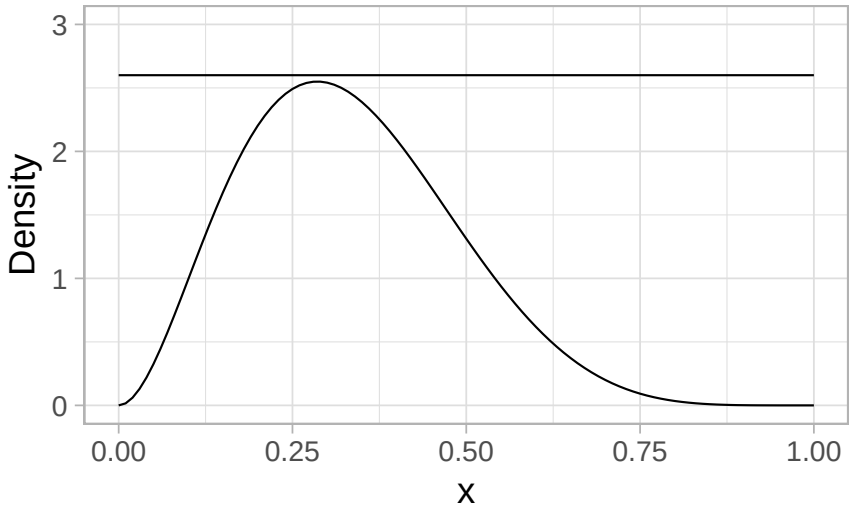
```
f <-  function(x) dbeta(x, shape1 = 3, shape2 = 6)
ggplot(tibble(x = seq(0, 1, 1e-2)), aes(x = x)) +
  stat_function(fun = f) +
  ylab("Density") +
  ylim(c(0, 3))
```



An example

- Suppose f is the beta density with parameters 3 and 6.
  - Why not use g as the uniform and $M = 2.6$?

```
M <- 2.6
ggplot(tibble(x = seq(0, 1, 1e-2)), aes(x = x)) +
  stat_function(fun = f) +
  stat_function(fun = function(x) M * dunif(x)) +
  ylab("Density") +
  ylim(c(0, 3))
```

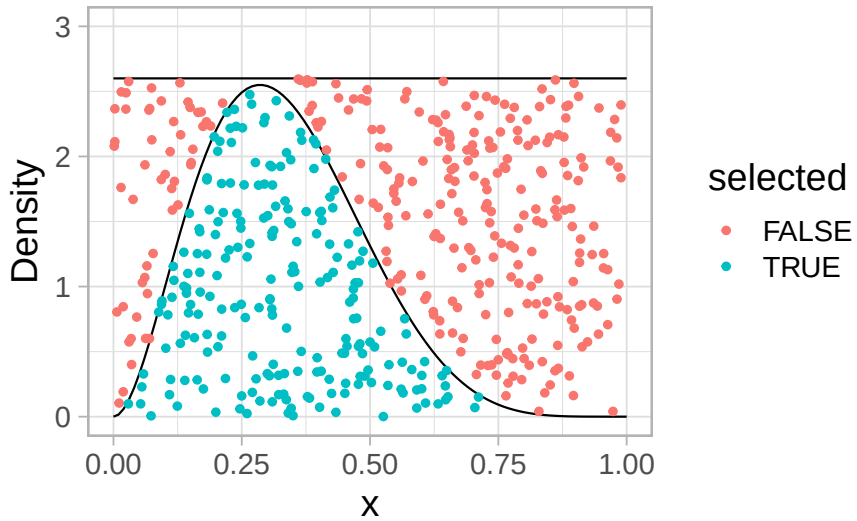


An example

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- Suppose f is the beta density with parameters 3 and 6.
  - Why not use g as the uniform and $M = 2.6$?

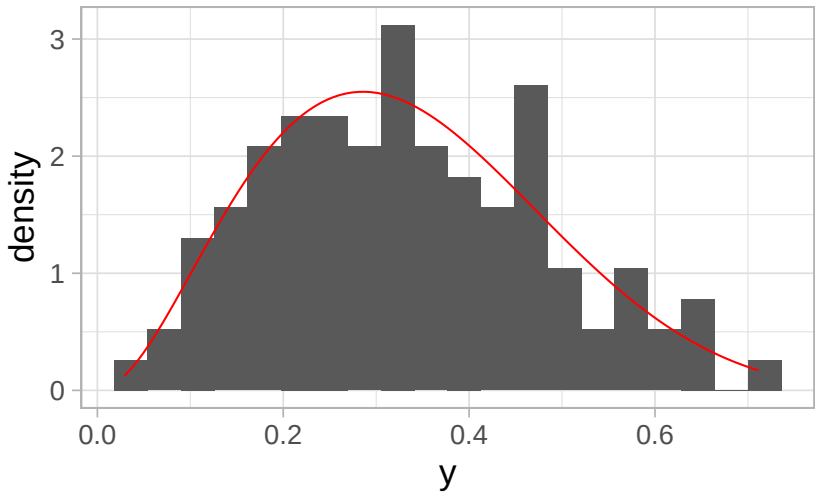
```
df <- tibble(y = runif(5e2, 0, 1), u = runif(5e2, 0, 1),
             loglik_ratio = dbeta(y, 3, 6) / M,
             selected = u < loglik_ratio)

ggplot(tibble(x = seq(0, 1, 1e-2)), aes(x = x)) +
  stat_function(fun = f) +
  stat_function(fun = function(x) 2.6 * dunif(x)) +
  geom_point(aes(x = y, y = M * u, color = selected), data = df) +
  ylab("Density") + ylim(c(0, 3))
```

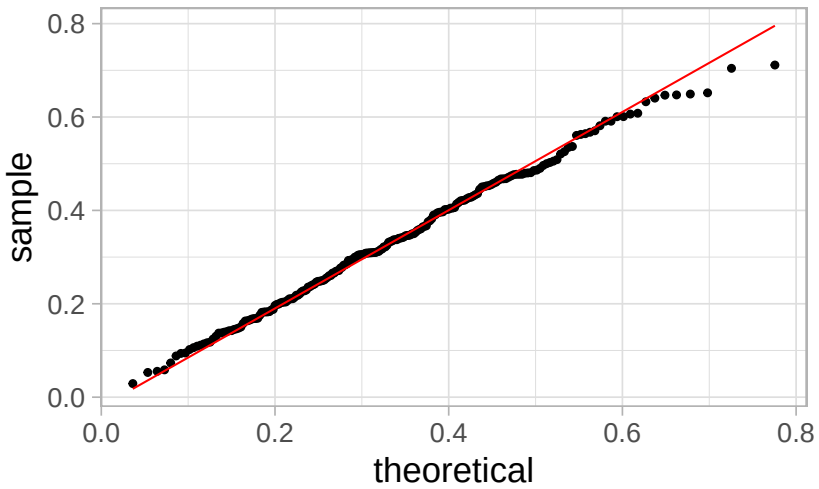


An example

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

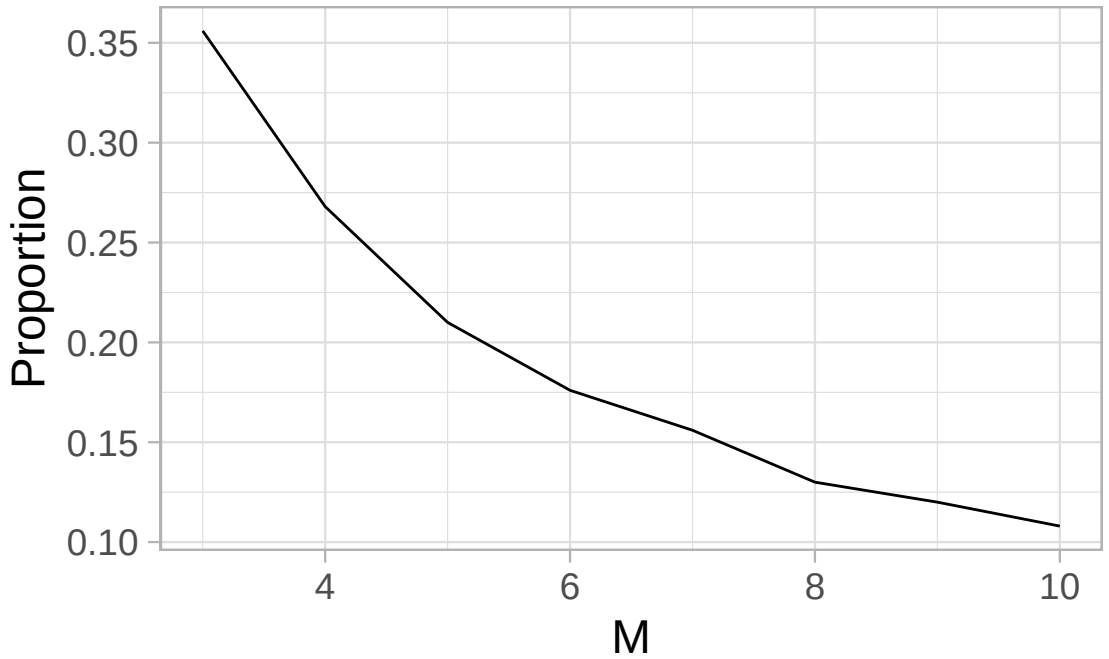
```
df_selected <- df %>%
  filter(selected)

# Proportion of selected samples?
NROW(df_selected) / NROW(df)
#> [1] 0.428
```





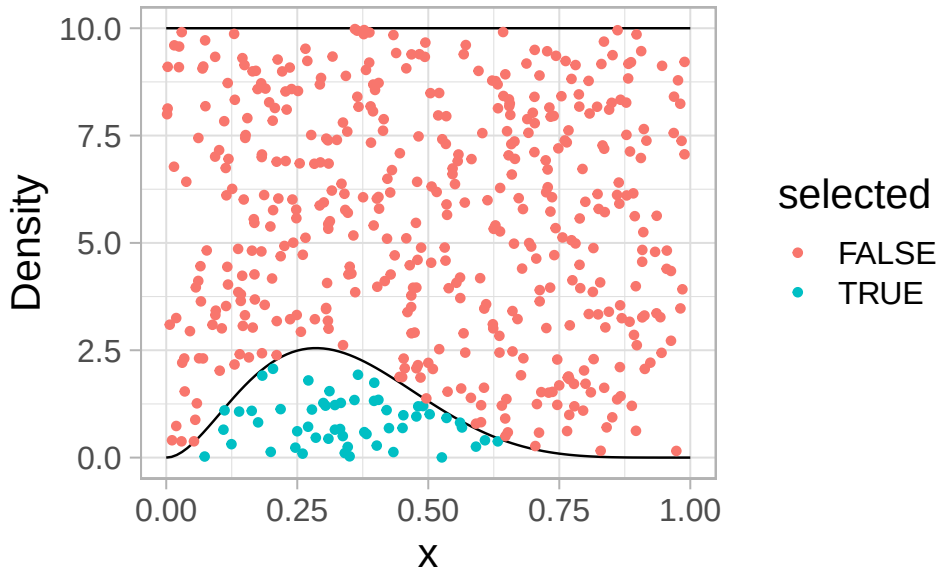
The effect of M


```
proportion_selected <- function(M) {
  df %>%
    mutate(loglik_ratio = dbeta(y, 3, 6) / M,
           selected = u < loglik_ratio) %>%
    filter(selected) %>%
  NROW() / NROW(df)
}
tibble(M = 3:10, Proportion = map_dbl(M, proportion_selected)) %>%
  ggplot(aes(M, Proportion)) + geom_line()
```



What's happening?


```
M <- 10
df <- df %>%
    mutate(loglik_ratio = dbeta(y, 3, 6) / M,
           selected = u < loglik_ratio)

ggplot(tibble(x = seq(0, 1, 1e-2)), aes(x = x)) +
  stat_function(fun = f) +
  stat_function(fun = function(x) M * dunif(x)) +
  geom_point(aes(x = y, y = M * u, color = selected), data = df) +
  ylab("Density") + ylim(c(0, M))
```



The theory again

- Let
  - $X \sim f$ where f is the target density,
  - $Y \sim g$ where g is a density satisfying $g(x) > 0$ if $f(x) > 0$.
  - $1 < M < \infty$ be such that $f(x) \leq Mg(x)$ for all values of x.
    - I.e., M is an upper bound on the likelihood ratio $f(x)/g(x)$.
    - $Mg(x)$ is called an **envelope** of $f(x)$.
- Then
  1. Draw $Y \sim g$.
  2. Draw $U \sim U[0, 1]$.
  3. If $U < f(Y)/Mg(Y)$, keep Y.
  4. Otherwise, repeat until a value is accepted.
  5. Repeat 1-4 until you have the desired sample size.
- The **unconditional acceptance probability:**

$$\mathbb{P}\left(U \leq \frac{f(Y)}{Mg(Y)}\right) = \frac{1}{M}$$

Envelope

- Remember:
  - The envelope is $Mg(x)$.
  - The unconditional acceptance probability is $1/M$.
- A good g:
  - Is easy to sample from.
  - Is "close" to f.
    - If g is "close" to f, then M can be close to 1, resulting in fewer rejected draws!
- A simple approach to finding an envelope:
  - Set $M = \max_x\{f(x)\}$.
  - Use a uniform distribution as g.

Outline

Confidence Intervals

- Let $\mathbf{X} = (X_1, \ldots, X_n)$ be random variables with joint distribution depending on θ.
  - Let $\theta \in \Theta \subset \mathbb{R}$ and two statistics $L(\mathbf{X}) < U(\mathbf{X})$. The random interval $[L(\mathbf{X}), U(\mathbf{X})]$ is called a α **confidence interval** for θ if

$$P[L(\mathbf{X}) \leq \theta \leq U(\mathbf{X})] \geq \alpha.$$

  - Let $\theta \in \Theta \subset \mathbb{R}^p$ and $R(\mathbf{X})$ be a subset of Θ depending on $\mathbf{X}$. Then $R(\mathbf{X})$ is called a α **confidence region** for θ if

$$P[\theta \in R(\mathbf{X})] \geq \alpha.$$

- The number α is called the **coverage probability** or **confidence level**.
  - This is NOT the probability that a fixed parameter lies in a given interval!

Pivotal method

- Let
  - $\mathbf{X} = (X_1, \ldots, X_n)$ have a joint distribution depending on a real-valued parameter θ,
  - and $g(\mathbf{X}; \theta)$ be a **pivot**.
    - I.e. a random variable whose distribution does not depend on θ.
- Then
  - Find constants $a < b$ such that $P[a \leq g(\mathbf{X}; \theta) \leq b] = \alpha$
  - And (try to) manipulate $\{a \leq g(\mathbf{X}; \theta) \leq b\}$ to get $L(\mathbf{X})$ and $U(\mathbf{X})$ such that $P[L(\mathbf{X}) \leq \theta \leq U(\mathbf{X})] \geq \alpha$.
- For instance:
  - If $X_1, \ldots, X_n \overset{iid}{\sim} N(\mu, \sigma^2)$, a well known pivot is the t-statistic

$$\sqrt{n}\frac{\overline{X}_n - \mu}{s} \sim t_{n-1} \text{ with } s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2}.$$

  - Hence, an α CI is $[\overline{X}_n - e, \overline{X}_n + e]$ with $e = T_{n-1}^{-1}((1+\alpha)/2)\frac{s}{\sqrt{n}}$ and T_{n-1}^{-1} the inverse cdf of t_{n-1}.

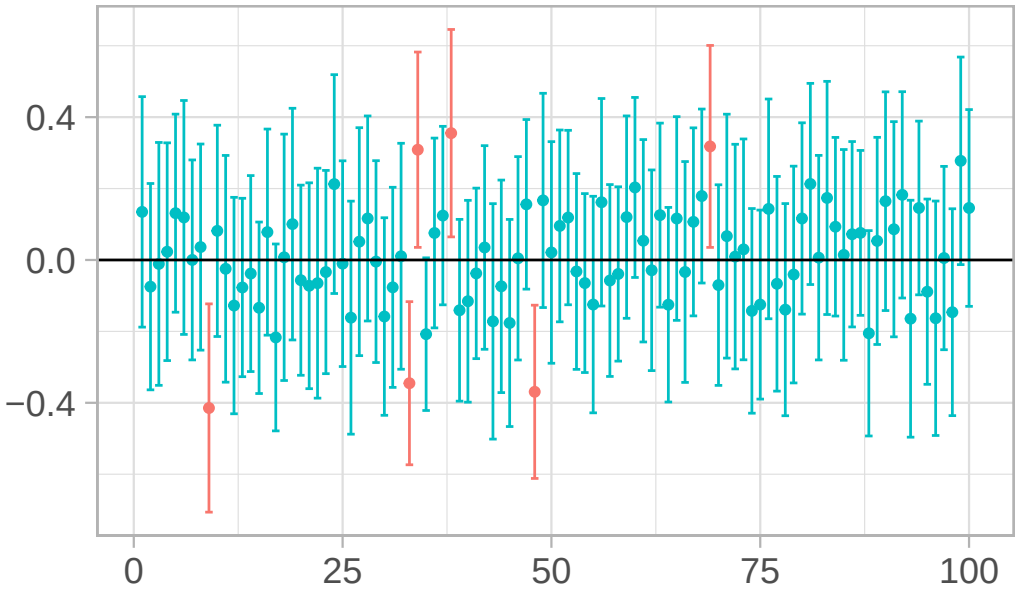
In R

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

```
normal_ci <- function(x, alpha = 0.95) {
  m <- mean(x)
  s <- sd(x)
  n <- length(x)
  tibble(m = m,
         lower = m + qt((1 - alpha)/2, n - 1) * s/sqrt(n),
         upper = m + qt((1 + alpha)/2, n - 1) * s/sqrt(n))
}

normal_ci(rnorm(10))
#> # A tibble: 1 x 3
#>       m  lower upper
#>   <dbl>  <dbl> <dbl>
#> 1 0.146 -0.672 0.965
normal_ci(rnorm(50))
#> # A tibble: 1 x 3
#>         m  lower   upper
#>     <dbl>  <dbl>   <dbl>
#> 1 -0.357 -0.636 -0.0781
```

In R


```
(simulation <- tibble(experiment_number = 1:100,
                      ci = map(experiment_number,
                               ~ normal_ci(rnorm(50)))) %>%
  unnest(ci)) %>% print(n = 2)
#> # A tibble: 100 x 4
#>   experiment_number       m  lower upper
#>               <int>   <dbl>  <dbl> <dbl>
#> 1                 1  0.135  -0.188 0.457
#> 2                 2 -0.0748 -0.364 0.214
#> # ... with 98 more rows
```



The bootstrap

- What if you can't find a pivot or can't invert it?
- One way would be to repeat an experiment over and over again, to find an approximation to the sampling distribution.
  - Often too expensive or time-consuming.
- Bradley Efron's Idea (1979):
  - **The bootstrap!**
  - Use computers to **simulate** replication.
    - Instead of obtaining new/independent datasets from the population, repeatedly obtain datasets from the sample itself.

To get a bootstrap estimate

- Resample from the original data n times **with replacement**.
- Use the new dataset to compute a new estimate.
- Repeat this to create B new datasets, and B new estimates.

Formally

- Let $X_1, \ldots, X_n \overset{iid}{\sim} F_\theta$ and let $\hat{\theta} = T(X_1, \ldots, X_n)$ be an estimator of θ_0.
- The procedure for $b = 1, \ldots, B$:
  1. Create a bootstrap sample $X_1^{(b)}, \ldots, X_n^{(b)}$, where each $X_i^{(b)}$ is obtained by drawing with replacement from $X_1, \ldots, X_n$.
  2. Compute a bootstrap estimates of θ as

$$\hat{\theta}^{(b)} = T(X_1^{(b)}, \ldots, X_n^{(b)}).$$

- Under some technical conditions, we have that

$$\hat{\theta}^{(b)} - \hat{\theta} \dot{\sim} \hat{\theta} - \theta_0,$$

  where $\dot{\sim}$ denotes "is approximately distributed as".
- In other words, the collection $(\hat{\theta}^{(b)} - \hat{\theta})_{b=1}^{B}$ approximates the sampling distribution of $\hat{\theta} - \theta_0$.

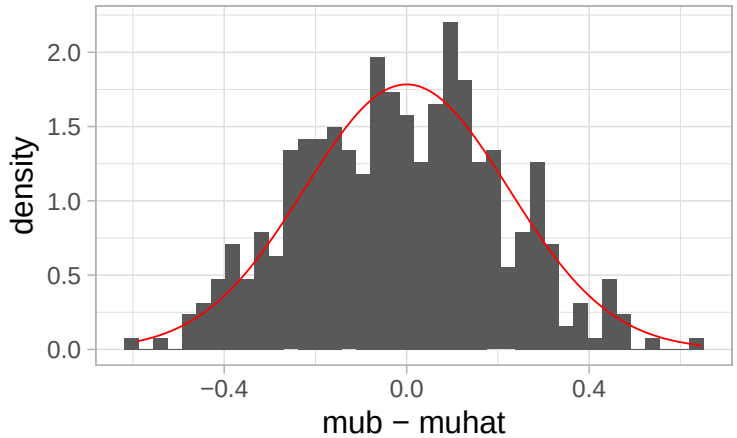
An example: Gaussian rvs

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- Let $X_1, \ldots, X_n \sim N(\mu, 1)$ and $\hat{\mu} = \overline{X}_n$ with $n = 20$.

```
n <- 20
x <- rnorm(n)
(muhat <- mean(x))
#> [1] -0.00178

boot_one <- function()
  sample(x, n, replace = TRUE)

B <- 400
xB <- tibble(b = 1:B,
             xb = map(b, ~boot_one()),
             mub = map_dbl(xb, mean))
xB %>% print(n = 2)
#> # A tibble: 400 x 3
#>       b xb              mub
#>   <int> <list>        <dbl>
#> 1     1 <dbl [20]> -0.353
#> 2     2 <dbl [20]> -0.399
#> # ... with 398 more rows
```

- Recall that $\hat{\mu} - \mu \sim N(0, 1/n)$ (the red curve).



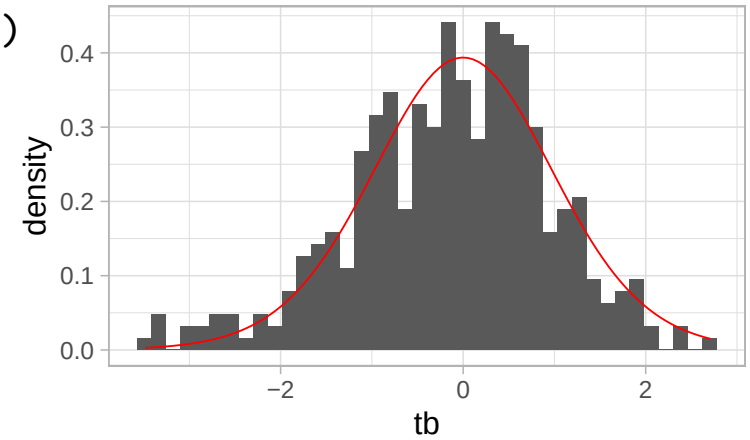
Gaussian rvs cont'd

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- Now, let $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$ and $\hat{\mu} = \overline{X}_n$ and σ^2 unknown.

```
tstat <- function(x, mu = 0)
  sqrt(length(x)) * (mean(x) - mu) / sd(x)

xB <- xB %>%
  mutate(tb = map_dbl(xb, tstat, mu = muhat))
xB %>% print(n = 2)
#> # A tibble: 400 x 4
#>       b xb                mub     tb
#>   <int> <list>          <dbl> <dbl>
#> 1     1 <dbl [20]> -0.353 -1.20
#> 2     2 <dbl [20]> -0.399 -1.98
#> # ... with 398 more rows
```

- Recall that $\sqrt{n}(\hat{\mu} - \mu)/s \sim t_{n-1}$ (the red curve).



Bootstraped confidence intervals

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- An intuitive way to bootstrap confidence intervals:
  - View $\hat{\theta}^{(1)}, \ldots, \ldots, \hat{\theta}^{(B)}$ as i.i.d. random sample.
  - Recall that $P(a \leq \hat{\theta}^{(b)} - \hat{\theta} \leq b) \approx P(a \leq \hat{\theta} - \theta_0 \leq b)$.
  - Use the quantiles of the distribution.
    - Denote as $\hat{\theta}^B_\alpha$ the α quantile of $\{\hat{\theta}^{(b)}\}_{b=1}^B$.
    - I.e., $P\left(\hat{\theta}^B_{(1-\alpha)/2} \leq \hat{\theta}^{(b)} - \hat{\theta} \leq \hat{\theta}^B_{(1+\alpha)/2}\right) \approx \alpha$

Definition (Basic bootstrap)

Use

- $L(\mathbf{X}) = 2\hat{\theta} - \hat{\theta}^B_{(1+\alpha)/2}$ and $U(\mathbf{X}) = 2\hat{\theta} - \hat{\theta}^B_{(1-\alpha)/2}$.

- Then $P(\theta_0 \in [L(\mathbf{X}), U(\mathbf{X})]) \approx \alpha$.
- E.g., for a 95% confidence interval, use
  - $L(\mathbf{X}) = 2\hat{\theta} - \hat{\theta}^B_{0.975}$ and $U(\mathbf{X}) = 2\hat{\theta} - \hat{\theta}^B_{0.025}$.

Bootstraped confidence intervals

- Why does it work?

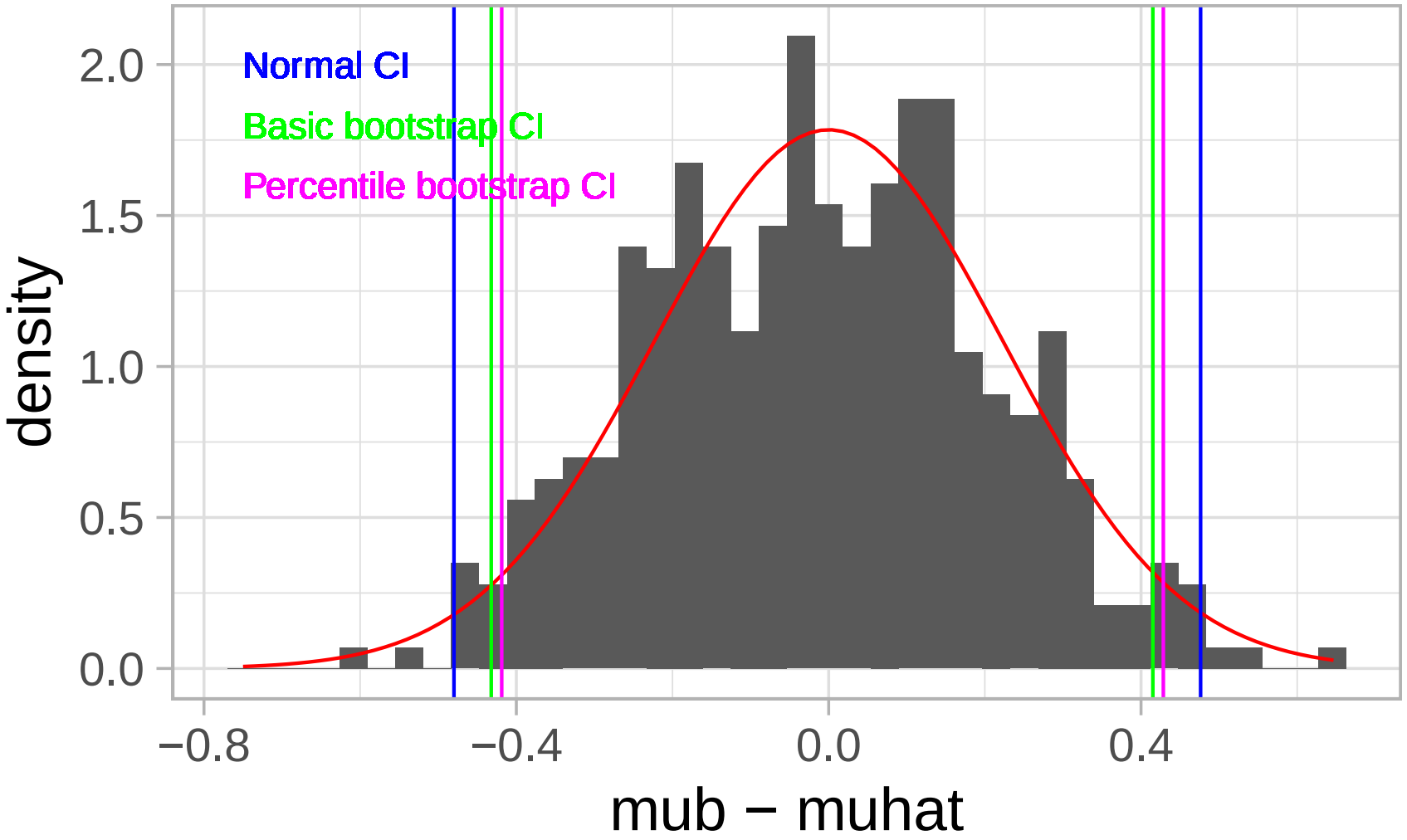
$$
\begin{aligned}
P\left(L(\mathbf{X}) \leq \theta_0 \leq U(\mathbf{X})\right) &= P\left(2\hat{\theta} - \hat{\theta}^B_{(1+\alpha)/2} \leq \theta_0 \leq 2\hat{\theta} - \hat{\theta}^B_{(1-\alpha)/2}\right) \\
&= P\left(\hat{\theta} - \hat{\theta}^B_{(1+\alpha)/2} \leq \theta_0 - \hat{\theta} \leq \hat{\theta} - \hat{\theta}^B_{(1-\alpha)/2}\right) \\
&= P\left(\hat{\theta}^B_{(1-\alpha)/2} - \hat{\theta} \leq \hat{\theta} - \theta_0 \leq \hat{\theta}^B_{(1+\alpha)/2} - \hat{\theta}\right) \\
&\approx P\left(\hat{\theta}^B_{(1-\alpha)/2} - \hat{\theta} \leq \hat{\theta}^{(b)} - \hat{\theta} \leq \hat{\theta}^B_{(1+\alpha)/2} - \hat{\theta}\right) \\
&= P\left(\hat{\theta}^B_{(1-\alpha)/2} \leq \hat{\theta}^{(b)} \leq \hat{\theta}^B_{(1+\alpha)/2}\right) \approx \alpha
\end{aligned}
$$

- Other types of bootstrap
  - Percentile bootstrap
    - Use $L(\mathbf{X}) = \hat{\theta}^B_{(1-\alpha)/2}$ and $U(\mathbf{X}) = \hat{\theta}^B_{(1+\alpha)/2}$.
  - Studentized bootstrap
  - Bias-corrected bootstrap

Gaussian rvs cont'd


```
ci_basic_bootstrap <- function(thetab, thetahat, alpha = 0.95) {
  qb <- quantile(thetab, c((1 - alpha) / 2, (1 + alpha) / 2))
  tibble(lower = 2 * thetahat - qb[2],
         upper = 2 * thetahat - qb[1])
}
ci_basic_bootstrap(xB %>% pull(mub), muhat)
#> # A tibble: 1 x 2
#>    lower upper
#>    <dbl> <dbl>
#> 1 -0.432 0.415

ci_percentile_bootstrap <- function(thetab, thetahat, alpha = 0.95) {
  qb <- quantile(thetab, c((1 - alpha) / 2, (1 + alpha) / 2))
  tibble(lower = qb[1],
         upper = qb[2])
}
ci_percentile_bootstrap(xB %>% pull(mub), muhat)
#> # A tibble: 1 x 2
#>    lower upper
#>    <dbl> <dbl>
#> 1 -0.419 0.429
```

Gaussian rvs cont'd



Hypothesis testing

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- Consider $X_1, \ldots, X_n$ i.i.d. random variables associated with a statistical model $(E, (F_\theta)_{\theta \in \Theta})$.
- Let Θ_0 and Θ_1 be disjoint sets $(\Theta_0 \cap \Theta_1 = \emptyset)$ of Θ.
  - $H_0 : \theta \in \Theta_0$ is the **null hypothesis**.
  - $H_1 : \theta \in \Theta_1$ is the **alternative hypothesis**.
- Goal: decide whether to reject H_0 by seeking evidence against.
- How?
  - Define a rejection region: $R = \{\mathbf{X} : H_0 \text{ is rejected}\}$.
  - Often $R = \{\mathbf{X} : T_n(\mathbf{X}) > t\}$ for some statistic T_n and critical value t.
- Two types of potential errors:
  - Type I:
    - Reject H_0 when it is actually true.
    - $\alpha = P_\theta(R),\ \theta \in \Theta_0$.
  - Type II:
    - Fail to reject H_0 when it is not true.
    - $\beta = P_\theta(R),\ \theta \in \Theta \setminus \Theta_0$.

P-values

Definition

The **p-value** of a test statistics T_n is the probability under H_0 of obtaining the observed value of T_n or a more extreme one, that is

$$p_{value} = P_{H_0}[T_n \geq t_{obs}],$$

where t_{obs} is the observed value of T_n.

- The smaller the p-values, the more evidence to reject H_0.
- Common in practice to use 0.05 or 0.01 as tresholds for indication of "statistically significance".
- P-values DO NOT
  - Give you the probability that the null hypothesis is true!
  - Indicate which alternative is best supported by the data!

Gaussian rvs cont'd

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- Let $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$ and $\hat{\mu} = \overline{X}_n$ and σ^2 unknown.
  - $T_n = \sqrt{n}(\overline{X}_n - \mu)/s \sim t_{n-1}$
  - So the p-value is $2(1 - P(t_{n-1} \leq |T_n|)$

```
2 * (1 - pt(abs(tstat(x)), n - 1))
#> [1] 0.994
t.test(x)
#>
#>  One Sample t-test
#>
#> data:  x
#> t = -0.008, df = 19, p-value = 1
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  -0.480  0.476
#> sample estimates:
#> mean of x
#>  -0.00178
```

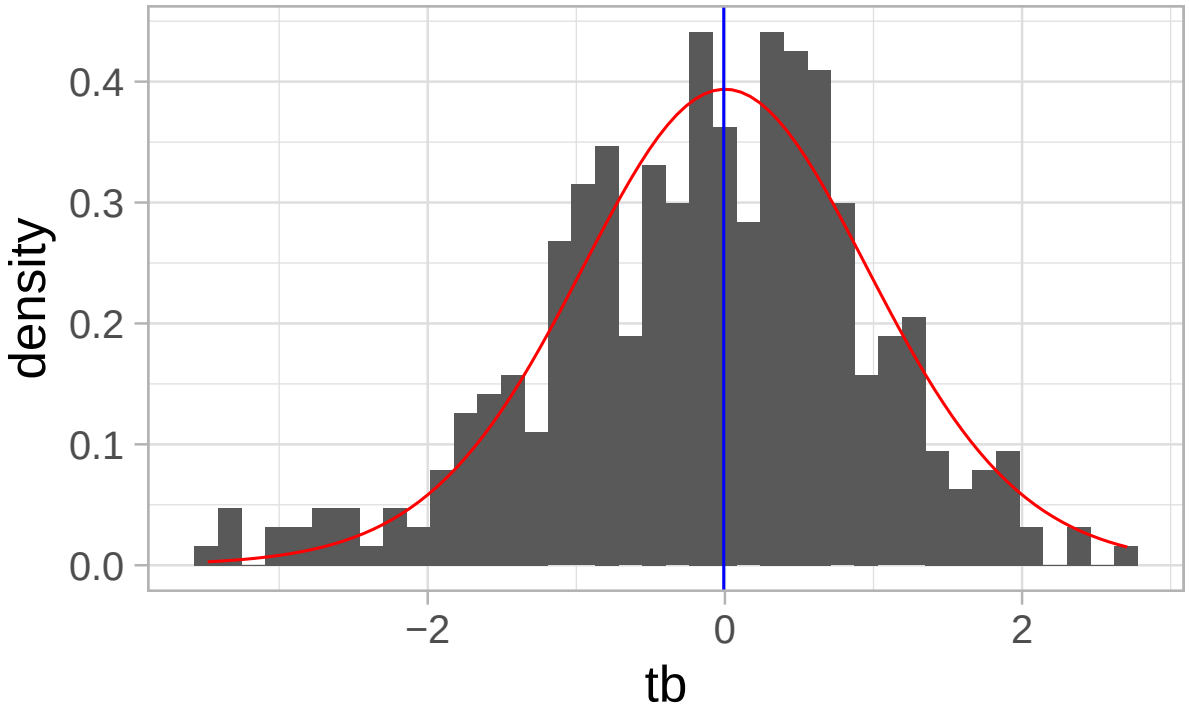
Bootstraped hypothesis test

COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

- What if we didn't know the distribution of T_n?
- We know that
$$\hat{\theta}^{(b)} - \hat{\theta} \overset{\cdot}{\sim} \hat{\theta} - \theta_0.$$
- So
$$T_n^{(b)} \overset{\cdot}{\sim} T_n.$$
- In other words, the distribution of $(T_n^{(b)})_{b=1}^B$ approximates that of T_n!
- Use the empirical cumulative distribution to obtain p-values.
- This works in most situations!

Gaussian rvs cont'd


```
tb <- xB %>%
  pull(tb)
2 * (1 - pt(abs(tstat(x)), n - 1))
#> [1] 0.994
2 * (1 - ecdf(tb)(abs(tstat(x))))
#> [1] 0.955
```



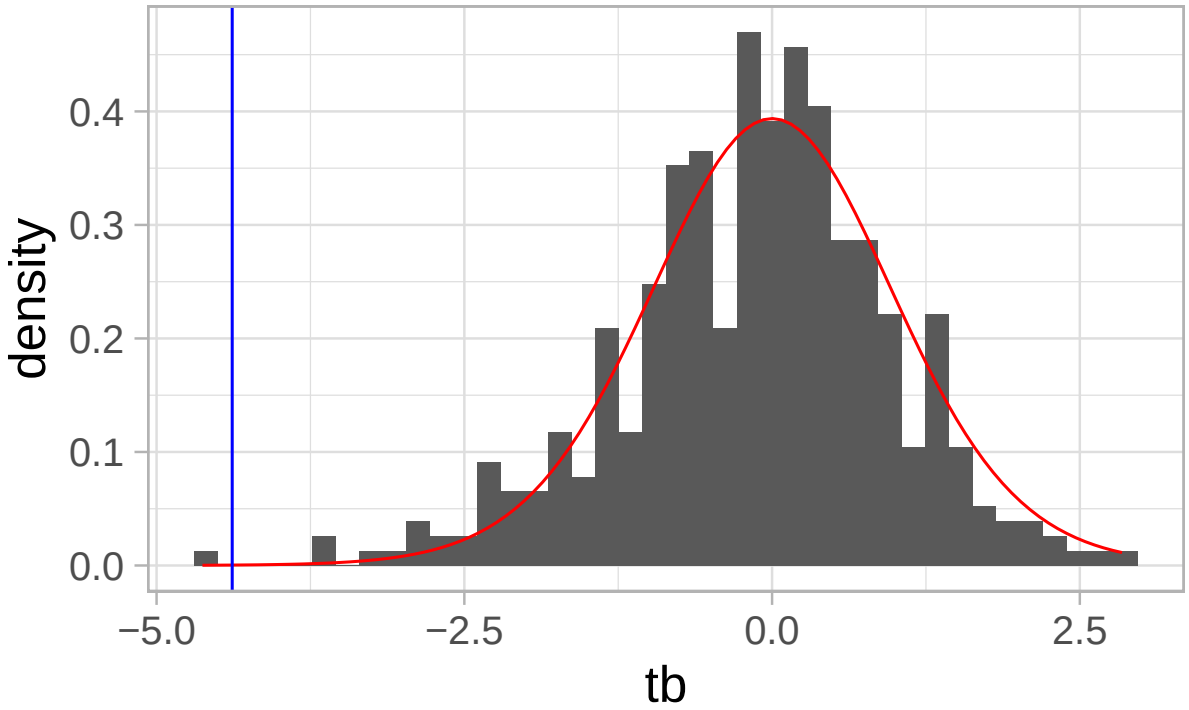
So how do we sample under the null?

- Let's test whether $\mu = \mu_0$ with $\mu_0 = 1$.
- But doing `sample(x, n, replace = TRUE)` is approximating the true distribution.
- So how do we resample under the null?

```
mu0 <- 1
x0 <- x - muhat + mu0
xB <- tibble(b = 1:B,
             xb = map(b, ~sample(x0, n, replace = TRUE)),
             tb = map_dbl(xb, tstat, mu = mu0))
xB %>% print(n = 2)
#> # A tibble: 400 x 3
#>       b xb               tb
#>   <int> <list>        <dbl>
#> 1     1 <dbl [20]>   1.38
#> 2     2 <dbl [20]> -0.520
#> # ... with 398 more rows
```

So how do we sample under the null?


```
tb <- xB %>%
  pull(tb)
2 * (1 - pt(abs(tstat(x, mu = mu0)), n - 1))
#> [1] 0.000318
2 * (1 - ecdf(tb)(abs(tstat(x, mu = mu0))))
#> [1] 0
```



Outline

Numerical integration

- Often we need to solve integrals,

$$I = \int f(x)dx,$$

  but doing so can be hard.
- Even when we know the function f, finding a closed-form antiderivative may be difficult or even impossible.
- In these cases, we'd like to find good ways to approximate the value of the integral.
- Such approximations are generally referred to as **numerical integration**.

Common techniques

- There are many methods of numerical integration.
- Quadrature-based integration:
  - Riemann rule
  - Trapezoid rule
  - Simpson's rule
  - Newton-Cotes (a generalization of the above three)
- . . . but works mainly for one-dimensional integrals!
- Today: Monte Carlo integration!

Law of large numbers

- Let $X_1, X_2, \ldots, X_n$ are iid with pdf f.
- By the law of large numbers

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) \xrightarrow{P} \mathbb{E}[g(X)] = \int g(x)f(x)dx.$$

The Monte Carlo principle

- To estimate $\int g(x)dx$:
  - Draw from f.
  - Take the sample mean of $g(x)/f(x)$.
  - Hence, $I_n = \frac{1}{n}\sum_{i=1}^{n} g(X_i)/f(X_i) \xrightarrow{P} I = \int g(x)dx$.

An example

- Estimate the integral

$$I = \int_{-\infty}^{\infty} g(x)dx$$

  for $g(x) = x^2 e^{-x^2}$ using MC techniques.
- We know that this equals $\sqrt{\pi}/2$ (how?), but let's still perform the exercise:
  - Draw standard normal rvs.
  - Take the sample mean of $g(x)/f(x)$ where $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$.

```
g_over_f <- function(x) sqrt(2 * pi) * x^2 * exp(-(1 / 2) * x^2)
n <- 1e4
x <- rnorm(n)
mean(g_over_f(x))
#> [1] 0.891
sqrt(pi) / 2
#> [1] 0.886
```

A variance estimate

- By the Central Limit Theorem,

$$I_n \xrightarrow{d} N\left(I, \frac{\sigma^2}{n}\right),$$

where $\sigma^2 = \text{var}\left[g(X)/f(X)\right]$.

```
(se <- sqrt(var(x) / n))
#> [1] 0.01
c(mean(g_over_f(x)) - 1.96 * se,
  mean(g_over_f(x)) + 1.96 * se)
#> [1] 0.871 0.911
sqrt(pi) / 2
#> [1] 0.886
```

- In other words:
  - The Monte Carlo approximation is unbiased.
  - The root mean square error is $\propto n^{-1/2}$.
    - Means that if we just keep taking Monte Carlo draws, the error can get as small as needed, even if g or f are very complicated.

Variance reduction

- Procedures used to increase the precision of the estimates.
- How?
  - Antithetic variates.
  - Control variates.
  - Importance sampling.

Antithetic variates

- The idea: for every sample path $X_1, \ldots, X_n$, consider also an estimator built on the antithetic path $-X_1, \ldots, -X_n$.
- How does that help?
  - Let $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ be two samples with the same pdf f and define

$$I_{n,x} = \frac{1}{n} \sum_{i=1}^{n} g(X_i)/f(X_i),$$

$$I_{n,y} = \frac{1}{n} \sum_{i=1}^{n} g(Y_i)/f(Y_i),$$

$$I_n = \frac{I_{n,x} + I_{n,y}}{2}.$$

  - We have that $E[I_n] = I$, that is I_n is an unbiased estimate for I.
  - Furthermore

$$\text{var}[I_n] = \frac{\text{var}[I_{n,x}] + \text{var}[I_{n,y}] + \text{cov}[I_{n,x}, I_{n,y}]}{4},$$

    so variance is reduced if $\text{cov}[I_{n,x}, I_{n,y}]$ is negative.

An example

- Let's use MC to estimate the integral

$$I = \int_0^1 1/(1+x)dx.$$

- We know that this equals log(2) but let's still do it:
  - Draw $U(0,1)$ rvs, so $f(x) = 1$.
  - Notice that if $X_i \sim U(0,1)$, then so does $1 - X_i$.
  - Take the sample mean of $(g(x) + g(1-x))/2$.

```
g <- function(x) 1 / (1 + x)
n <- 1e3
x <- runif(n)
```

```
mean(g(x))
#> [1] 0.695
mean((g(x) + g(1 - x)) / 2)
#> [1] 0.693
log(2)
#> [1] 0.693
```

```
var(g(x)) / n
#> [1] 1.94e-05
var((g(x) + g(1 - x)) / 2) / n
#> [1] 5.9e-07
cor(g(x), g(1 - x))
#> [1] -0.939
```

Control variates

- The idea: exploit information about the errors in estimates of known quantities to reduce the error of the estimate of I.
- How does that work?
  - Suppose we know another statistic $T_n = T(X_1, \ldots, X_n)$ is such that $E[T_n] = \tau$ with τ known.
  - Then
$$I_n^c = I_n + c(T_n - \tau)$$
is unbiased for any value of c because $E(T_n - \tau) = 0$.
  - Furthermore, we have
$$\text{var}[I_n^c] = \text{var}[I_n] + c^2\text{var}[T_n] + 2c\,\text{cov}[I_n, T_n],$$
which is minimized at $c = -\text{cov}[I_n, T_n]/\text{var}[T_n]$, resulting in
$$\text{var}[I_n^c] = \text{var}[I_n] - \frac{\text{cov}[I_n, T_n]^2}{\text{var}[T_n]} = (1 - \text{cor}[I_n, T_n]^2)\text{var}[I_n].$$
  - Larger $|\text{cor}[I_n, T_n]|$ imply greater variance reduction!

An example

- Let's use MC to estimate again the integral

$$I = \int_0^1 1/(1+x)dx.$$

  - Let's define h(x) = 1 + x.
  - Hence, $\int_0^1 h(x)dx = 3/2$
  - Take the sample mean of $g(x) + c(h(x) - 3/2)$.

```
h <- function(x) 1 + x
c <- -cov(g(x), h(x)) / var(h(x))
```

```
mean(g(x))
#> [1] 0.695
mean(g(x) + c * (h(x) - 3/2))
#> [1] 0.693
log(2)
#> [1] 0.693
```

```
var(g(x)) / n
#> [1] 1.94e-05
var(g(x) + c * (h(x) - 3/2)) / n
#> [1] 6.15e-07
cor(g(x), h(x))
#> [1] -0.984
```

Importance sampling

- Essentially: how to choose f?
- In principle, any f which is supported on the same set as g could be used for Monte Carlo.
- In practice, we want f to be **easy to simulate** and s.t. g/f
  - **Has low variance**
    - It generally improves efficiency to have the shape of $f(x)$ follow that of $g(x)$ s.t. σ^2 is small.
  - **Takes a simple form**
    - It is often worth looking carefully at the integrand to see if a probability density can be factored out of it.

An example

- Goal: estimate $I = P(Y \geq 3)$ for $Y \sim N(0,1)$, that is

$$I = \int_3^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = \int_{-\infty}^\infty g(x) dx,$$

  where $g(x) = \mathbb{1}(x \geq 3) \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$.
- A naive MC estimator would use standard normal rvs and

$$I_{n,1} = \frac{1}{n} \sum_{i=1}^n \frac{g(X_i)}{f(X_i)} = \frac{1}{n} \sum_{i=1}^n \mathbb{1}(X_i \geq 3).$$

- Clearly, this works, but what is the problem here?
  - Very few observations will be larger than 3...
- What if we used rvs that are $N(1,4)$ instead?
  - We have $g(x)/f(x) = 2\mathbb{1}(x \geq 3) e^{-x^2/2 + (x-1)^2/8}$, so

$$I_{n,2} = \frac{1}{n} \sum_{i=1}^n 2\mathbb{1}(X_i \geq 3) e^{-X_i^2/2 + (X_i-1)^2/8}$$

Example cont'd


```r
n <- 1e4
x1 <- rnorm(n)
x2 <- 1 + sqrt(4) * x1
g_over_f1 <- function(x) x >= 3
g_over_f2 <- function(x) 2 * (x >= 3) * exp(-x^2/2 + (x - 1)^2 / 8)

mean(g_over_f1(x1))
#> [1] 0.0015
mean(g_over_f2(x2))
#> [1] 0.00131
1 - pnorm(3)
#> [1] 0.00135

sqrt(var(g_over_f1(x1)) / n)
#> [1] 0.000387
sqrt(var(g_over_f2(x2)) / n)
#> [1] 5.11e-05
```