4. Simulations

Jørgen Bølstad

STV4022, University of Oslo

1 Intro to Simulation

2 Basic Programming

3 Examples of Simulations

4 Summarizing Simulations

5 Bootstrapping

6 Conclusion

Simulation vs. Estimation

- A typical goal in statistics is to **estimate parameters** to describe data:
  - Specifying a probability model for the data
  - Estimating plausible parameter values in light of the data and the model
  - In short: **model + data → parameter estimates**
- Simulations turn to this logic on its head:
  - We specify a probability model and **generate data** from the model
  - In short: **model + parameter values → (fake) data**

Why Simulate?

1. **Explore** patterns of random variation
   - Testing and challenging our intuitions about how things work
2. Approximate the **sampling distribution** of the data and our estimators
   - **Bootstrapping:** Estimate uncertainty of estimates
   - **Monte Carlo studies**: Generating data with known parameters and assessing estimator properties (e.g. bias, variance, MSE)
3. Illustrate **uncertainty in predictions** from estimated models
4. **Assessing models** by checking if their predictions make sense
5. Other purposes (e.g. fitting Bayesian models)
   - (rstanarm does this, but we will not look at the details)

Generating Random Data

- A wide range of probability distributions are available in R
  - These are functions with different sets of parameters
- The functions to generate data start with d
  - `dnorm`, `dbinom`, `dunif`, etc.
- Recall: The Bernoulli distribution applies to a single binary trial
  - Its only parameter is the probability of success, p
  - It equals a Binomial distribution with a single trial (size $= 1$)

Simulating random data from the Bernoulli distribution

```
fakedat <- rbinom(30, size = 1, prob = .1) # 30 draws with p = .1
fakedat
```

```
##  [1] 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
```

```
sum(fakedat)
```

```
## [1] 4
```

Reproducible Random Numbers

- R relies on a **random number generator** (RNG) to generate data
  - The data are not completely random, but **pseudo random**:
  - The sequence of numbers is completely determined by an initial **seed**

Set the seed for reproducible random data

```
set.seed(1)         # Set seed to 1 before starting
rbinom(5, 1, .5)  # Take five draws from the Bernoulli distribution
```

```
## [1] 0 0 1 1 0
```

```
rbinom(5, 1, .5)  # Take five different draws
```

```
## [1] 1 1 1 1 0
```

```
set.seed(1)         # Re-set seed to 1
rbinom(5, 1, .5)  # Obtain the *same* draws as before
```

```
## [1] 0 0 1 1 0
```

```
rbinom(5, 1, .5)
```

```
## [1] 1 1 1 1 0
```

Functions

- **Functions** take input(s), perform some operation, and produce outputs
  - Inputs are in R referred to as **arguments**
  - Outputs are in R referred to as **value**
    - R functions can only **return** a single object

We can easily define our own functions

```
my_function <- function(x = 1) { # x=1 will be used if no x is provided
  y <- x^2
  return(y) # Specify what the function should return
}
my_function(2) # Test the function with x = 2
```

```
## [1] 4
```

Loops

- **Loops** repeat some operation for a specified number of times

Running our function for each integer from 1 to 5

```
for(i in 1:5) { # For each integer from 1 to 5, set i to this number
  print(my_function(i)) # Then perform operations on i
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```

- Functions in R are often **vectorized**, removing the need for loops
  - Vectorized operations are a lot faster then explicit loops in R

```
my_function(1:5)
```

```
## [1]  1  4  9 16 25
```

If ... Else Statements

- **If statements** execute operations only if some **condition** is true

Printing only numbers whose square is above or equal to 10

```
for(i in 1:5) {
  if (my_function(i) >= 10) {
    print(i)
  } else { # Note: You can drop the else-part when it is not needed
    print("Nope")
  }
}
```

```
## [1] "Nope"
## [1] "Nope"
## [1] "Nope"
## [1] 4
## [1] 5
```

Subsetting

- **Subsetting** means selecting some portion of the data
  - In base R, we use square brackets and add:
    - The indexes of the elements we want
    - Or a logical (TRUE/FALSE) vector of the same length as the data

Subsetting a (one-dimensional) vector

```
dat <- 0:8
dat[5:6]
```

```
## [1] 4 5
```

```
select <- dat > 5 # Create logical vector: Is dat above 5?
select
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

```
dat[select]
```

```
## [1] 6 7 8
```

Simulation of Discrete Probability Model

- The probability that a baby is girl is ca. 48.8%
- If 400 babies are born at a hospital in a year, how many will be girls?

Simulate a single instance (one particular year)

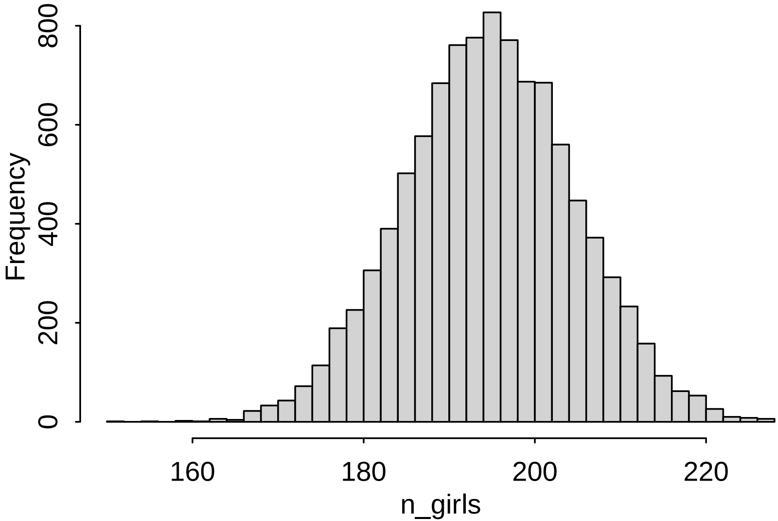
```
n_girls <- rbinom(n = 1, size = 400, prob = 0.488)
print(n_girls) # This is what *could* happen in *one* instance
```

```
## [1] 198
```

Repeat simulation 1000 times

```
n_sims <- 10000
n_girls <- rep(NA, n_sims)
for (s in 1:n_sims) {
  n_girls[s] <- rbinom(1, 400, 0.488)
}
```

Histogram of the Results

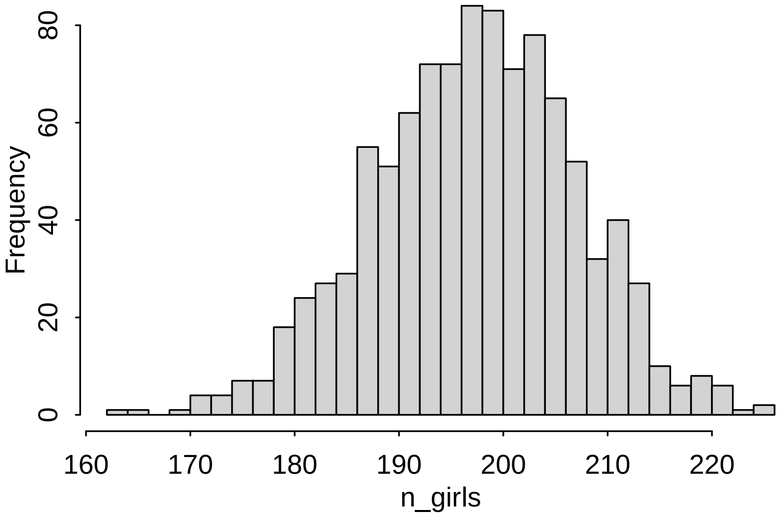


- Accounting for fraternal and identical twins
  - Each having a 49.5% change of being girls

Accounting for twins and repeating 1000 times

```
n_girls <- rep(NA, n_sims)
for (s in 1:n_sims){
  birth_type <- sample(c("fraternal twin","identical twin",
                          "single birth"),
    size=400, replace=TRUE, prob=c(1/125, 1/300, 1 - 1/125 - 1/300))
  girls <- rep(NA, 400)
  for (i in 1:400){
    if (birth_type[i]=="single birth"){
      girls[i] <- rbinom(1, 1, 0.488)}
    else if (birth_type[i]=="identical twin"){
      girls[i] <- 2*rbinom(1, 1, 0.495)}
    else if (birth_type[i]=="fraternal twin"){
      girls[i] <- rbinom(1, 2, 0.495)}
  }
  n_girls[s] <- sum(girls)
}
```

Histogram of the Results



Summarizing a Set of Simulations

- The `mean` or `median` summarize the **location** of a distribution
- **Variation** is traditionally summarized by the variance or std. dev. (SD)
  - In R, we obtain these using the functions `var` and `sd`

Using the mean and SD to summarize a distribution

```
mean(n_girls)
```

```
## [1] 197.778
```

```
sd(n_girls)
```

```
## [1] 9.797383
```

The Median Absolute Deviation vs. SD

- Gelman et al. suggest using the **median absolute deviation** (MAD)
  - This is the median absolute deviation *from the median* of a variable
  - The median makes this measure more stable than the normal SD
  - They multiply MAD by 1.483, which yields the SD for the normal dist.
  - They refer to the resulting measure as **MAD SD**, in R: `mad`
  - You can think of the MAD SD reported by `rstanarm` as a standard error

Using the median and MAD SD to summarize a distribution

```
median(n_girls)
```

```
## [1] 198
```

```
mad(n_girls)
```

```
## [1] 10.3782
```

Summarizing by Uncertainty Intervals

- We can also summarize distributions in terms of intervals
- This can be useful both for parameter estimates and predictions

Using the quantile function to summarize a distribution

```r
# An interval containing 95% of the values:
quantile(n_girls, probs = c(.025, .975))
```

```
##    2.5%   97.5%
## 178.975 216.000
```

```r
# An interval containing 50% of the values:
quantile(n_girls, probs = c(.25, .75))
```

```
## 25% 75%
## 191 204
```

Bootstrapping

- **Bootstrapping** helps us assess the uncertainty in estimates
  - Useful if we lack measures of uncertainty
    - Which may happen for complicated frequentist analyses
    - In contrast, a fully Bayesian analysis *always* estimates uncertainty
- Bootstrapping is randomly **resampling the data with replacement**
  - Creates new datasets where each datapoint can appear several times
- This is a way to approximate some aspect of the sampling distribution
  - Illustrates what kind of data might be expected if they were recollected
- Our estimator can be applied to each new dataset
  - The distribution of estimates can be used to assess uncertainty
  - It can be used as an approximate sampling distribution for the estimator

Example: Bootstrapping a Ratio of Medians

- Estimating the ratio of women's earnings to men's earnings
  - Data: Survey of 1816 US respondents, 1990 (Gelman et. al ch. 5)

Median of women's earnings, divided by the median of men's earnings

```
earn <- earnings$earn
male <- earnings$male
print(median(earn[male==0]) / median(earn[male==1]))
```

```
## [1] 0.6
```

- Point estimate: The median earnings of women is 60% that of men
  - *What is the standard error of this estimate? We don't know!*

Example: Simple bootstrapping code

A set of bootstrap simulations

```
boot_ratio <- function(data){
  n <- nrow(data)
  boot <- sample(n, replace=TRUE)
  earn_boot <- data$earn[boot]
  male_boot <- data$male[boot]
  ratio_boot <- median(earn_boot[male_boot==0]) /
                  median(earn_boot[male_boot==1])
  return(ratio_boot)
}
n_sims <- 10000
output <- replicate(n_sims, boot_ratio(data=earnings))
```

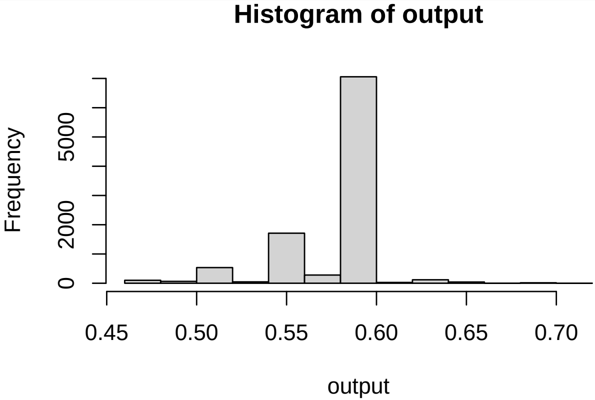
Example: Bootstrapping results

- The standard error of our estimated ratio of earnings is .03:

```
round(sd(output), 2)
```

```
## [1] 0.03
```

```
hist(output)
```



Example: What If We Compared Means Instead?

Bootstrap simulations for a ratio of means

```
boot_ratio <- function(data){
  n <- nrow(data)
  boot <- sample(n, replace=TRUE)
  earn_boot <- data$earn[boot]
  male_boot <- data$male[boot]
  ratio_boot <- mean(earn_boot[male_boot==0]) / # Now using MEAN
                  mean(earn_boot[male_boot==1])
  return(ratio_boot)
}
n_sims <- 10000
output_means <- replicate(n_sims, boot_ratio(data=earnings))
```

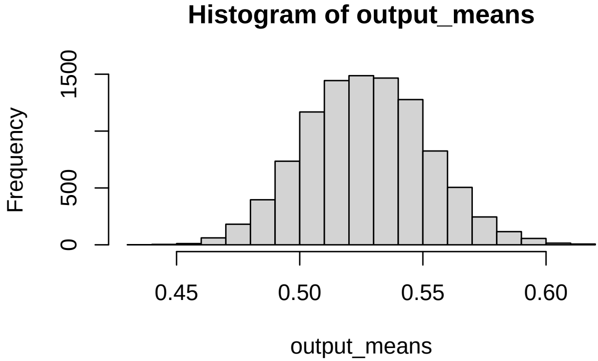
Example: What If We Compared Means Instead?

- The sampling distribution is now a **normal distribution**
  - We could have calculated the standard error analytically

```
round(sd(output_means), 2)
```

```
## [1] 0.02
```

```
hist(output_means)
```



Challenges and Limitations of Bootstrapping

- Bootstrapping is easy for data consisting of a simple random sample:
  - Just resample units with replacement
- For **other datastructures**, we face hard choices:
  - Time series: Simple resampling will likely be meaningless
  - Multilevel data: With multiple obs. per cluster, what do we resample?
    - E.g.: If survey respondents answer questions about several parties, do we resample individuals or parties within individuals?
    - The answers depend on what uncertainty we are trying to approximate
- The validity of the bootstrap **depends on the data** and analysis:
  - If the data contains no black respondents voting for a Republican, the bootstrap of a traditional analysis would suggest the probability of a black person voting Republican is zero and that the uncertainty in this estimate is also zero. This is clearly wrong.

Final Comments

- Simulations are useful for a wide range of tasks in statistics:
    - Assessing models
    - Assessing uncertainty in estimates
    - Assessing uncertainty in predictions
    - And more!
- If you if are unsure if you have understood some statistical concept, or want to know how your model handles a certain situation:
    - Simulations can often provide an answer