GOV 2001/ 1002/ E-2001 Section 1[1]
Monte Carlo Simulation

Anton Strezhnev

Harvard University

January 27, 2016

[1] These notes and accompanying code draw on the notes from TF's from previous years.

OUTLINE

LOGISTICS

Course Website: j.mp/G2001
lecture notes, videos, announcements

Canvas: problem sets, discussion board

Perusall: readings **Learning Catalytics:** in class activities

LOGISTICS

Reading Assignment- Unifying Political Methodology chs 1-3. and Intro to Monte Carlo Simulation. Due Monday at 2pm.

Problem Set 1- Due by 6pm next Wednesday on Canvas.

Discussion board- Post and get to know each other! (Change notifications)

booc.io- Browse the overall structure of the course and review connections between material.

OUTLINE

WHY MONTE CARLO SIMULATION?

WHY MONTE CARLO SIMULATION?

- Quantities can be hard to calculate analytically

Why Monte Carlo Simulation?

- Quantities can be hard to calculate analytically
  - If $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{Normal}(\mu, \sigma^2)$, What's the expectation of

$$\sum_{i=1}^{N} X_i$$

Why Monte Carlo Simulation?

- Quantities can be hard to calculate analytically
  - If $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{Normal}(\mu, \sigma^2)$, What's the expectation of

$$\sum_{i=1}^{N} X_i$$

  - That's a random sum (number of terms added is random)!

Why Monte Carlo Simulation?

- Quantities can be hard to calculate analytically
  - If $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{Normal}(\mu, \sigma^2)$, What's the expectation of

$$\sum_{i=1}^{N} X_i$$

  - That's a random sum (number of terms added is random)! There's an analytical solution (Wald's equation) that relies on Law of Iterated Expectations.

Why Monte Carlo Simulation?

- Quantities can be hard to calculate analytically
  - If $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{Normal}(\mu, \sigma^2)$, What's the expectation of

$$\sum_{i=1}^{N} X_i$$

  - That's a random sum (number of terms added is random)! There's an analytical solution (Wald's equation) that relies on Law of Iterated Expectations.
  - Or we could simulate it...
- Extremely common in modern day social science for...
  - Simulating interpretable results from complex models -

Why Monte Carlo Simulation?

- Quantities can be hard to calculate analytically
  - If $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{Normal}(\mu, \sigma^2)$, What's the expectation of

$$\sum_{i=1}^{N} X_i$$

  - That's a random sum (number of terms added is random)! There's an analytical solution (Wald's equation) that relies on Law of Iterated Expectations.
  - Or we could simulate it...
- Extremely common in modern day social science for...
  - Simulating interpretable results from complex models - e.g. FiveThirtyEight's election forecasts.
  - Comparing models and estimators –

WHY MONTE CARLO SIMULATION?

- Quantities can be hard to calculate analytically
  - If $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{Normal}(\mu, \sigma^2)$, What's the expectation of

$$\sum_{i=1}^{N} X_i$$

  - That's a random sum (number of terms added is random)! There's an analytical solution (Wald's equation) that relies on Law of Iterated Expectations.
  - Or we could simulate it...
- Extremely common in modern day social science for...
  - Simulating interpretable results from complex models - e.g. FiveThirtyEight's election forecasts.
  - Comparing models and estimators – showing that one method is better than the other in lieu of proofs.

Why Monte Carlo Simulation?

- Quantities can be hard to calculate analytically
  - If $N \sim \text{Poisson}(\lambda)$ and $X_i \sim \text{Normal}(\mu, \sigma^2)$, What's the expectation of

$$\sum_{i=1}^{N} X_i$$

  - That's a random sum (number of terms added is random)! There's an analytical solution (Wald's equation) that relies on Law of Iterated Expectations.
  - Or we could simulate it...
- Extremely common in modern day social science for...
  - Simulating interpretable results from complex models - e.g. FiveThirtyEight's election forecasts.
  - Comparing models and estimators – showing that one method is better than the other in lieu of proofs. In *Political Analysis*, about 88 articles since 2012 that have "monte carlo" and simulation in the text (according to Google Scholar).

A Recipe for Monte Carlo

A Recipe for Monte Carlo

- To answer any question using a Monte Carlo simulation, just need to follow three basic steps.

A Recipe for Monte Carlo

- To answer any question using a Monte Carlo simulation, just need to follow three basic steps.
  - Write down a **probabilistic model** of the process you're interested in.

A Recipe for Monte Carlo

- To answer any question using a Monte Carlo simulation, just need to follow three basic steps.
  - Write down a **probabilistic model** of the process you're interested in.
  - Repeatedly sample from the random components of the model to obtain realizations of the outcome you care about.

A Recipe for Monte Carlo

- To answer any question using a Monte Carlo simulation, just need to follow three basic steps.
  - Write down a **probabilistic model** of the process you're interested in.
  - Repeatedly sample from the random components of the model to obtain realizations of the outcome you care about.
  - Compute the summary of interest using the realizations (e.g. expectation)!
- That simple!

A Recipe for Monte Carlo

- To answer any question using a Monte Carlo simulation, just need to follow three basic steps.
  - Write down a **probabilistic model** of the process you're interested in.
  - Repeatedly sample from the random components of the model to obtain realizations of the outcome you care about.
  - Compute the summary of interest using the realizations (e.g. expectation)!
- That simple!The challenge is figuring out the model.

Applied Example: Electoral College Forecasts

Applied Example: Electoral College Forecasts

- Poll-aggregating forecasting models are really popular for U.S. elections (see FiveThirtyEight, Votamatic (`http://votamatic.org/`), etc...).

Applied Example: Electoral College Forecasts

- Poll-aggregating forecasting models are really popular for U.S. elections (see FiveThirtyEight, Votamatic (`http://votamatic.org/`), etc...).
- How do they generate predictions for the Electoral College?

Applied Example: Electoral College Forecasts

- Poll-aggregating forecasting models are really popular for U.S. elections (see FiveThirtyEight, Votamatic (`http://votamatic.org/`), etc...).
- How do they generate predictions for the Electoral College? Same way we'll be getting results out of our models in this class!
  - Build a model that converts polling results to state-by-state forecasts – different models put different emphasis on polls vs. fundamentals

APPLIED EXAMPLE: ELECTORAL COLLEGE FORECASTS

- Poll-aggregating forecasting models are really popular for U.S. elections (see FiveThirtyEight, Votamatic (`http://votamatic.org/`), etc...).
- How do they generate predictions for the Electoral College? Same way we'll be getting results out of our models in this class!
  - Build a model that converts polling results to state-by-state forecasts – different models put different emphasis on polls vs. fundamentals
  - Generate predicted win probabilities for each state (taking into account "fundamental" and "estimation" uncertainty).

Applied Example: Electoral College Forecasts

- Poll-aggregating forecasting models are really popular for U.S. elections (see FiveThirtyEight, Votamatic (`http://votamatic.org/`), etc...).
- How do they generate predictions for the Electoral College? Same way we'll be getting results out of our models in this class!
  - Build a model that converts polling results to state-by-state forecasts – different models put different emphasis on polls vs. fundamentals
  - Generate predicted win probabilities for each state (taking into account "fundamental" and "estimation" uncertainty).
  - Use a weighted coin flip (bernoulli trial) to get a state-by-state winner and tally up Electoral College votes

Applied Example: Electoral College Forecasts

- Poll-aggregating forecasting models are really popular for U.S. elections (see FiveThirtyEight, Votamatic (`http://votamatic.org/`), etc...).
- How do they generate predictions for the Electoral College? Same way we'll be getting results out of our models in this class!
  - Build a model that converts polling results to state-by-state forecasts – different models put different emphasis on polls vs. fundamentals
  - Generate predicted win probabilities for each state (taking into account "fundamental" and "estimation" uncertainty).
  - Use a weighted coin flip (bernoulli trial) to get a state-by-state winner and tally up Electoral College votes
  - Repeat the process a lot of times to get a distribution of predicted electoral college results.

APPLIED EXAMPLE: ELECTORAL COLLEGE FORECASTS

- Poll-aggregating forecasting models are really popular for U.S. elections (see FiveThirtyEight, Votamatic (`http://votamatic.org/`), etc...).
- How do they generate predictions for the Electoral College? Same way we'll be getting results out of our models in this class!
  - Build a model that converts polling results to state-by-state forecasts – different models put different emphasis on polls vs. fundamentals
  - Generate predicted win probabilities for each state (taking into account "fundamental" and "estimation" uncertainty).
  - Use a weighted coin flip (bernoulli trial) to get a state-by-state winner and tally up Electoral College votes
  - Repeat the process a lot of times to get a distribution of predicted electoral college results.

WHY IT WORKS?

WHY IT WORKS?

- Key principle: **The Law of Large Numbers**

Why it works?

- Key principle: **The Law of Large Numbers**
  - Let $X_1, \ldots, X_n$ be n *independent* and *identically* distributed random variables.

Why it works?

- Key principle: **The Law of Large Numbers**
  - Let $X_1, \ldots, X_n$ be n *independent* and *identically* distributed random variables. Let $E[X_i] = \mu$ and $Var(X_i) = \sigma^2$.

Why it works?

- Key principle: **The Law of Large Numbers**
  - Let $X_1, \ldots, X_n$ be n *independent* and *identically* distributed random variables. Let $E[X_i] = \mu$ and $Var(X_i) = \sigma^2$.
  - As n goes to ∞, the sample average $\frac{\sum_{i=1}^{N} X_i}{n}$ converges almost surely to the true value $E[X_i] = \mu$.

WHY IT WORKS?

- Key principle: **The Law of Large Numbers**
  - Let $X_1, \ldots, X_n$ be n *independent* and *identically* distributed random variables. Let $E[X_i] = \mu$ and $Var(X_i) = \sigma^2$.
  - As n goes to ∞, the sample average $\frac{\sum_{i=1}^{N} X_i}{n}$ converges almost surely to the true value $E[X_i] = \mu$.
  - The same thing holds for deterministic functions of X_i.

WHY IT WORKS?

- Key principle: **The Law of Large Numbers**
  - Let $X_1, \ldots, X_n$ be n *independent* and *identically* distributed random variables. Let $E[X_i] = \mu$ and $Var(X_i) = \sigma^2$.
  - As n goes to ∞, the sample average $\frac{\sum_{i=1}^{N} X_i}{n}$ converges almost surely to the true value $E[X_i] = \mu$.
  - The same thing holds for deterministic functions of X_i.
- What does this mean?

WHY IT WORKS?

- Key principle: **The Law of Large Numbers**
  - Let $X_1, \ldots, X_n$ be n *independent* and *identically* distributed random variables. Let $E[X_i] = \mu$ and $Var(X_i) = \sigma^2$.
  - As n goes to ∞, the sample average $\frac{\sum_{i=1}^{N} X_i}{n}$ converges almost surely to the true value $E[X_i] = \mu$.
  - The same thing holds for deterministic functions of X_i.
- What does this mean? It means that if I repeatedly take a lot of independent samples from a process and average the results, I get close to the true mean output of that process!

WORKING EXAMPLE: BLACKJACK

- What's the probability that a blackjack dealer will bust (score $>$ 21) when their first card is a 6?
- Hard analytically, since dealer has a draw-dependent stopping rule.

WORKING EXAMPLE: BLACKJACK

- What's the probability that a blackjack dealer will bust (score > 21) when their first card is a 6?
- Hard analytically, since dealer has a draw-dependent stopping rule. Keep drawing new cards as long as the value of the hand is lower than 17.

Working example: Blackjack

- How do we do this with simulation?

WORKING EXAMPLE: BLACKJACK

- How do we do this with simulation? First, connect probabilities to expectations.

WORKING EXAMPLE: BLACKJACK

- How do we do this with simulation? First, connect probabilities to expectations.
- Let X be the event that the dealer busts.

Working example: Blackjack

- How do we do this with simulation? First, connect probabilities to expectations.
- Let X be the event that the dealer busts. Let $\mathcal{I}(X = 1)$ be an "indicator" random variable that takes on 1 if the dealer busts and 0 if they do not.

WORKING EXAMPLE: BLACKJACK

- How do we do this with simulation? First, connect probabilities to expectations.
- Let X be the event that the dealer busts. Let $\mathcal{I}(X = 1)$ be an "indicator" random variable that takes on 1 if the dealer busts and 0 if they do not.
- What's $E[\mathcal{I}(X = 1)]$?

WORKING EXAMPLE: BLACKJACK

- How do we do this with simulation? First, connect probabilities to expectations.
- Let X be the event that the dealer busts. Let $\mathcal{I}(X = 1)$ be an "indicator" random variable that takes on 1 if the dealer busts and 0 if they do not.
- What's $E[\mathcal{I}(X = 1)]$? It's $Pr(\text{Bust})$!

Working example: Blackjack

- How do we do this with simulation? First, connect probabilities to expectations.
- Let X be the event that the dealer busts. Let $\mathcal{I}(X = 1)$ be an "indicator" random variable that takes on 1 if the dealer busts and 0 if they do not.
- What's $E[\mathcal{I}(X = 1)]$? It's $Pr(\text{Bust})$!
- So if we simulate the game a lot of times,

Working example: Blackjack

- How do we do this with simulation? First, connect probabilities to expectations.
- Let X be the event that the dealer busts. Let $\mathcal{I}(X = 1)$ be an "indicator" random variable that takes on 1 if the dealer busts and 0 if they do not.
- What's $E[\mathcal{I}(X = 1)]$? It's $Pr(\text{Bust})$!
- So if we simulate the game a lot of times, record 1 if the dealer busts and 0 if they don't,

WORKING EXAMPLE: BLACKJACK

- How do we do this with simulation? First, connect probabilities to expectations.
- Let X be the event that the dealer busts. Let $\mathcal{I}(X = 1)$ be an "indicator" random variable that takes on 1 if the dealer busts and 0 if they do not.
- What's $E[\mathcal{I}(X = 1)]$? It's $Pr(\text{Bust})$!
- So if we simulate the game a lot of times, record 1 if the dealer busts and 0 if they don't, then take the mean of those simulations...

Working example: Blackjack

- How do we do this with simulation? First, connect probabilities to expectations.
- Let X be the event that the dealer busts. Let $\mathcal{I}(X = 1)$ be an "indicator" random variable that takes on 1 if the dealer busts and 0 if they do not.
- What's $E[\mathcal{I}(X = 1)]$? It's $Pr(\text{Bust})$!
- So if we simulate the game a lot of times, record 1 if the dealer busts and 0 if they don't, then take the mean of those simulations...
- We get an approximation of the probability the dealer busts!

Working example: Blackjack

- How do we write the simulation? Imagine playing the game in R. Start by initializing the simulation.

WORKING EXAMPLE: BLACKJACK

- How do we write the simulation? Imagine playing the game in R. Start by initializing the simulation.

```
### Set seed for Random Number Generator
set.seed(02138)

### Set number of iterations
nIter = 10000

### Initialize a place-holder vector to store results
dealer_busts <- rep(NA, nIter)

### Save the card number that the dealer shows
dealer_shows <- "6"
### We're saving as a string because that's how our deck
### will be laid out
```

- We also want to make a function to handle repetitive tasks (like calculating the value of a hand).

Working example: Blackjack

```
### Create a function to calculate the hand value
### 'hand' is a vector of character strings with cards
calculate_value <- function(hand){
  ## Convert face cards to their value
  hand[hand %in% c("J","Q","K")] <- "10"
  ## Label aces 11 for now
  hand[hand %in% c("A")] <- "11"
  ## Convert to integer and sum
  value <- sum(as.integer(hand))
  ## If value > 21, adjust any aces
  if (value > 21){
    # How many aces are left?
    ace_count <- sum(hand == "11")
    # As long as there are some 11-valued aces
    while(ace_count > 0){
      # Find an 11-valued ace and make it a 1
      hand[match("11",hand)] <- "1"
      # Recalculate value
      value <- sum(as.integer(hand))
      # If you brought the value down below or equal to 21, break early
      if (value <= 21){
        break
      }
      # Recalculate number of aces
      ace_count <- sum(hand == "11")
    }
  }
  return(value)
}
```

Working example: Blackjack

```
### Start the simulation
for (iter in 1:nIter){
  ### Initialize a deck of 52 cards (Note, all of these are strings)
  deck <- rep(c(2,3,4,5,6,7,8,9,10,"J","Q","K","A"), 4)
  deck <- deck[-match(dealer_shows, deck)] # match() returns the position of the
       first matches in a vector
  ### Initialize a place-holder containing their hand
  hand <- c(dealer_shows)
  ### Choose the second card
  card2 <- sample(deck, 1)
  ### Take that card out of the deck
  deck <- deck[-match(card2, deck)]
  ### Add it to the hand
  hand <- c(dealer_shows, card2)
  ### Is the dealer still playing?
  play_on <- T
  ### While the dealer is still playing
  while(play_on){
    ### Save hand value
    hand_value <- calculate_value(hand)
    ### Did the dealer bust?
    if (hand_value > 21){
      ### Dealer busted, store a 1
      dealer_busts[iter] <- 1
      break # Break the loop
    }
```

Code continues on the next page...

Working example: Blackjack

```
    ### Should the dealer stand (dealer stands on soft 17)
    else if (hand_value >= 17 & hand_value <= 21){
      ### Dealer didn't bust, store a 1
      dealer_busts[iter] <- 0
      break # Break the loop
    ### Otherwise, the dealer hits
    }else{
      ## Choose the new card
      new_card <- sample(deck, 1)
      ### Take that card out of the deck
      deck <- deck[-match(new_card, deck)]
      ### Add it to the hand
      hand <- c(hand, new_card)
    }
  }
}
#### Calculate the probability
mean(dealer_busts) ### About 42%
```

WORKING EXAMPLE: BLACKJACK

```
    ### Should the dealer stand (dealer stands on soft 17)
    else if (hand_value >= 17 & hand_value <= 21){
      ### Dealer didn't bust, store a 1
      dealer_busts[iter] <- 0
      break # Break the loop
    ### Otherwise, the dealer hits
    }else{
      ## Choose the new card
      new_card <- sample(deck, 1)
      ### Take that card out of the deck
      deck <- deck[-match(new_card, deck)]
      ### Add it to the hand
      hand <- c(hand, new_card)
    }
  }
}
#### Calculate the probability
mean(dealer_busts) ### About 42%
```

So the probability of busting when the dealer shows a 6 is about .42. See how this changes when a dealer shows a Jack!

Commonly asked questions

COMMONLY ASKED QUESTIONS

- **Why independent and identical trials?**

Commonly asked questions

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.

COMMONLY ASKED QUESTIONS

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.
  - If trials aren't independent, we don't explore the full distribution.

Commonly asked questions

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.
  - If trials aren't independent, we don't explore the full distribution.
  - Intuitively, if $X_2, \ldots, X_n$ are perfectly correlated with X_1, then $\bar{X} = X_1$, rather than the true mean.

Commonly asked questions

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.
  - If trials aren't independent, we don't explore the full distribution.
  - Intuitively, if $X_2, \ldots, X_n$ are perfectly correlated with X_1, then $\bar{X} = X_1$, rather than the true mean.
  - More complex Monte Carlo methods exist for certain types of dependence...

Commonly asked questions

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.
  - If trials aren't independent, we don't explore the full distribution.
  - Intuitively, if $X_2, \ldots, X_n$ are perfectly correlated with X_1, then $\bar{X} = X_1$, rather than the true mean.
  - More complex Monte Carlo methods exist for certain types of dependence...
- **Why do I need to set a random seed?**

Commonly asked questions

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.
  - If trials aren't independent, we don't explore the full distribution.
  - Intuitively, if $X_2, \ldots, X_n$ are perfectly correlated with X_1, then $\bar{X} = X_1$, rather than the true mean.
  - More complex Monte Carlo methods exist for certain types of dependence...
- **Why do I need to set a random seed?**
  - Computers can't generate truly random numbers.

Commonly asked questions

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.
  - If trials aren't independent, we don't explore the full distribution.
  - Intuitively, if $X_2, \ldots, X_n$ are perfectly correlated with X_1, then $\bar{X} = X_1$, rather than the true mean.
  - More complex Monte Carlo methods exist for certain types of dependence...
- **Why do I need to set a random seed?**
  - Computers can't generate truly random numbers. There are algorithms that deterministically generate sequences of psuedo-random numbers.

Commonly asked questions

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.
  - If trials aren't independent, we don't explore the full distribution.
  - Intuitively, if $X_2, \ldots, X_n$ are perfectly correlated with X_1, then $\bar{X} = X_1$, rather than the true mean.
  - More complex Monte Carlo methods exist for certain types of dependence...
- **Why do I need to set a random seed?**
  - Computers can't generate truly random numbers. There are algorithms that deterministically generate sequences of psuedo-random numbers. That process takes as input some "seed" value.

Commonly asked questions

- **Why independent and identical trials?**
  - If trials aren't identical, there isn't a single "mean" parameter we get convergence to.
  - If trials aren't independent, we don't explore the full distribution.
  - Intuitively, if $X_2, \ldots, X_n$ are perfectly correlated with X_1, then $\bar{X} = X_1$, rather than the true mean.
  - More complex Monte Carlo methods exist for certain types of dependence...
- **Why do I need to set a random seed?**
  - Computers can't generate truly random numbers. There are algorithms that deterministically generate sequences of psuedo-random numbers. That process takes as input some "seed" value. Setting a seed allows you to replicate your simulated results.

Outline

Matrix Algebra

- You know that the `%*%` command can be used to do matrix multiplication (row-by-column).

Matrix Algebra

- You know that the `%*%` command can be used to do matrix multiplication (row-by-column).
- Suppose you want to transpose a matrix to make it conformable for multiplication.

MATRIX ALGEBRA

- You know that the `%*%` command can be used to do matrix multiplication (row-by-column).
- Suppose you want to transpose a matrix to make it conformable for multiplication. Use the `t()` function!

```
### Make a matrix with 3 rows and 2 columns
X <- matrix(data=c(1,3), nrow=3, ncol=2)
### Multiply the matrix by itself
XprimeX <- t(X)%*%X
XprimeX
```

Matrix Algebra

- You know that the `%*%` command can be used to do matrix multiplication (row-by-column).
- Suppose you want to transpose a matrix to make it conformable for multiplication. Use the `t()` function!

```
### Make a matrix with 3 rows and 2 columns
X <- matrix(data=c(1,3), nrow=3, ncol=2)
### Multiply the matrix by itself
XprimeX <- t(X)%*%X
XprimeX
```

- Or suppose you want to invert a square matrix.

Matrix Algebra

- You know that the `%*%` command can be used to do matrix multiplication (row-by-column).
- Suppose you want to transpose a matrix to make it conformable for multiplication. Use the `t()` function!

```
### Make a matrix with 3 rows and 2 columns
X <- matrix(data=c(1,3), nrow=3, ncol=2)
### Multiply the matrix by itself
XprimeX <- t(X)%*%X
XprimeX
```

- Or suppose you want to invert a square matrix. Use the `solve()` command.

Matrix Algebra

- You know that the `%*%` command can be used to do matrix multiplication (row-by-column).
- Suppose you want to transpose a matrix to make it conformable for multiplication. Use the `t()` function!

```
### Make a matrix with 3 rows and 2 columns
X <- matrix(data=c(1,3), nrow=3, ncol=2)
### Multiply the matrix by itself
XprimeX <- t(X)%*%X
XprimeX
```

- Or suppose you want to invert a square matrix. Use the `solve()` command.
- See the help files to see how R deals with multiplying matrices by vectors `help("%*%")`. Often easier to convert a vector to a row or column matrix to make sure dimensions are right.

Matrix Algebra

- You know that the `%*%` command can be used to do matrix multiplication (row-by-column).
- Suppose you want to transpose a matrix to make it conformable for multiplication. Use the `t()` function!

```
### Make a matrix with 3 rows and 2 columns
X <- matrix(data=c(1,3), nrow=3, ncol=2)
### Multiply the matrix by itself
XprimeX <- t(X)%*%X
XprimeX
```

- Or suppose you want to invert a square matrix. Use the `solve()` command.
- See the help files to see how R deals with multiplying matrices by vectors `help("%*%")`. Often easier to convert a vector to a row or column matrix to make sure dimensions are right.
- Also, you can't do matrix operations on a data frame (which is the object type that `read.csv()` returns)! Use `as.matrix()` to convert.

Random Sampling

- If you want to sample from a finite set of values in a vector, use the `sample()` function.

RANDOM SAMPLING

- If you want to sample from a finite set of values in a vector, use the `sample()` function.
- `sample()` takes 4 arguments. `sample(x, size, replace = FALSE, prob = NULL)`.

Random Sampling

- If you want to sample from a finite set of values in a vector, use the `sample()` function.
- `sample()` takes 4 arguments. `sample(x, size, replace = FALSE, prob = NULL)`.
  - `x` is the vector that you want to sample from. Can be *any* vector (so not just sampling numerical values).

Random Sampling

- If you want to sample from a finite set of values in a vector, use the `sample()` function.
- `sample()` takes 4 arguments. `sample(x, size, replace = FALSE, prob = NULL)`.
  - `x` is the vector that you want to sample from. Can be *any* vector (so not just sampling numerical values).
  - `size` is the number of items you want chosen. The function will throw an error if `size` is greater than the length of `x` and `replace = FALSE`.
  - `replace` is a boolean (either `TRUE` or `FALSE`). If `TRUE`, sampling will be done with replacement, if `FALSE`, it will be done without replacement. By default it is `FALSE`.

Random Sampling

- If you want to sample from a finite set of values in a vector, use the `sample()` function.
- `sample()` takes 4 arguments. `sample(x, size, replace = FALSE, prob = NULL)`.
  - `x` is the vector that you want to sample from. Can be *any* vector (so not just sampling numerical values).
  - `size` is the number of items you want chosen. The function will throw an error if `size` is greater than the length of `x` and `replace = FALSE`.
  - `replace` is a boolean (either `TRUE` or `FALSE`). If `TRUE`, sampling will be done with replacement, if `FALSE`, it will be done without replacement. By default it is `FALSE`.
  - `prob` is a vector of weights if you want to do weighted random sampling – we won't be needing it. Defaults to `NULL` which uses uniform weights.

```
#### Roll two six-sided dice and add them together
twodice <- sum(sample(1:6, 2, replace=T))
```

Sampling from a probability density

- If you want to sample from a probability density, R has a whole bunch of pre-defined functions that do this. Most follow the naming convention `r[NAME OF DISTRIBUTION]`.

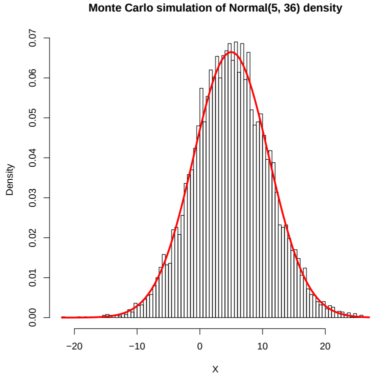
SAMPLING FROM A PROBABILITY DENSITY

- If you want to sample from a probability density, R has a whole bunch of pre-defined functions that do this. Most follow the naming convention `r[NAME OF DISTRIBUTION]`.For example `rnorm()` draws random normal variates. `runif()` draws from the continuous uniform distribution.
- Example: Repeatedly sample from a normal with mean $\mu = 5$, variance $\sigma^2 = 36$

```
normal_vars <- rnorm(n=10000, mean = 5, sd = 6)
```

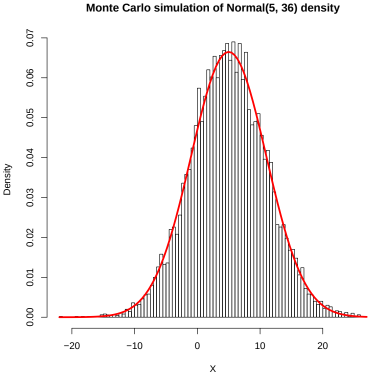
Sampling from a probability density

Plot a histogram of `normal_vars` with the normal density overlaid



SAMPLING FROM A PROBABILITY DENSITY

Plot a histogram of `normal_vars` with the normal density overlaid



By the Monte Carlo principle, the vector of independent draws approximates the true density.

OUTLINE

Why bootstrap?

- In many cases, we want to learn something about the sampling distribution of an estimator.

WHY BOOTSTRAP?

- In many cases, we want to learn something about the sampling distribution of an estimator. What's a sampling distribution?

Why bootstrap?

- In many cases, we want to learn something about the sampling distribution of an estimator. What's a sampling distribution?
- The distribution of an estimator in repeated samples from a population!

Why bootstrap?

- In many cases, we want to learn something about the sampling distribution of an estimator. What's a sampling distribution?
- The distribution of an estimator in repeated samples from a population! In practice, we only observe a single sample, so this distribution is *theoretical*.
- How do we figure out the standard error?

WHY BOOTSTRAP?

- In many cases, we want to learn something about the sampling distribution of an estimator. What's a sampling distribution?
- The distribution of an estimator in repeated samples from a population! In practice, we only observe a single sample, so this distribution is *theoretical*.
- How do we figure out the standard error? Before, we used theory to derive estimators.

Why bootstrap?

- In many cases, we want to learn something about the sampling distribution of an estimator. What's a sampling distribution?
- The distribution of an estimator in repeated samples from a population! In practice, we only observe a single sample, so this distribution is *theoretical*.
- How do we figure out the standard error? Before, we used theory to derive estimators.
- Example: In linear regression, what assumptions do we need to derive an unbiased/consistent estimator of the regression SEs?

WHY BOOTSTRAP?

- In many cases, we want to learn something about the sampling distribution of an estimator. What's a sampling distribution?
- The distribution of an estimator in repeated samples from a population! In practice, we only observe a single sample, so this distribution is *theoretical*.
- How do we figure out the standard error? Before, we used theory to derive estimators.
- Example: In linear regression, what assumptions do we need to derive an unbiased/consistent estimator of the regression SEs? The Gauss-Markov assumptions!
- But what happens when we don't have an easy theory or can't make certain simplifying assumptions...?

WHY BOOTSTRAP?

- We can use a Monte Carlo technique!

WHY BOOTSTRAP?

- We can use a Monte Carlo technique!Suppose we could repeatedly sample from the population and compute our estimator.

Why bootstrap?

- We can use a Monte Carlo technique!Suppose we could repeatedly sample from the population and compute our estimator. Then the properties of Monte Carlo simulation mean that for a large number of re-samples, we would get the sampling distribution.

Why bootstrap?

- We can use a Monte Carlo technique!Suppose we could repeatedly sample from the population and compute our estimator. Then the properties of Monte Carlo simulation mean that for a large number of re-samples, we would get the sampling distribution.
- However, we can't repeatedly sample from the true population - we just have a sample.

Why bootstrap?

- We can use a Monte Carlo technique!Suppose we could repeatedly sample from the population and compute our estimator. Then the properties of Monte Carlo simulation mean that for a large number of re-samples, we would get the sampling distribution.
- However, we can't repeatedly sample from the true population - we just have a sample. Good enough!

Why bootstrap?

- We can use a Monte Carlo technique!Suppose we could repeatedly sample from the population and compute our estimator. Then the properties of Monte Carlo simulation mean that for a large number of re-samples, we would get the sampling distribution.
- However, we can't repeatedly sample from the true population - we just have a sample. Good enough!
- Replace sampling from the population with sampling from our particular sample.

WHY BOOTSTRAP?

- We can use a Monte Carlo technique!Suppose we could repeatedly sample from the population and compute our estimator. Then the properties of Monte Carlo simulation mean that for a large number of re-samples, we would get the sampling distribution.
- However, we can't repeatedly sample from the true population - we just have a sample. Good enough!
- Replace sampling from the population with sampling from our particular sample. This gives us a Monte Carlo approximation for a sampling distribution.

Why bootstrap?

- We can use a Monte Carlo technique!Suppose we could repeatedly sample from the population and compute our estimator. Then the properties of Monte Carlo simulation mean that for a large number of re-samples, we would get the sampling distribution.
- However, we can't repeatedly sample from the true population - we just have a sample. Good enough!
- Replace sampling from the population with sampling from our particular sample. This gives us a Monte Carlo approximation for a sampling distribution. And if our sample is a random sample from the population, we know our Monte Carlo approximation its consistent for the true sampling distribution.

How to bootstrap

- Step 1: Get a sample of size n
- Step 2: Re-sample from your sample n observations **with replacement** – some observations will be repeated, others will not be included.
- Step 3: Calculate and store your estimate.
- Step 4: Repeat steps 2 and 3 many times until you have a long vector of bootstrapped estimates. This is your simulated sampling distribution.
- Step 5: Calculate your quantity of interest using the simulated sampling distribution (e.g. for the standard error, take the standard deviation of your bootstrapped estimates)

Working example - Regression

We know that the OLS standard error estimator is biased and inconsistent under heteroskedasticity.

Working example - Regression

We know that the OLS standard error estimator is biased and inconsistent under heteroskedasticity. Let's generate some simulated data with heteroskedasticity.

```
# Set random seed
set.seed(02138)
# Number of observations in our simulated dataset
obs <- 200
# Initialize the data frame that will store our simulated data
simulated_data <- data.frame(Y=rep(NA,100), X=rep(NA, 100))
# For each observation
for (i in 1:obs){
  # Generate some simulated X value from a normal (0,1)
  x <- rnorm(1, 0, 1)
  # Generate our "regression" error but with non-constant variance
  # The variance depends on x - higher for larger absolute values of X
  u <- rnorm(1, 0, 5 + 2*(x^2))
  # Generate Y as a linear combination of X and the error
  y <- 1 + 5*x + u
  # Store in our data frame
  simulated_data[i,] <- c(y, x)
}
### Run Regression
reg <- lm(Y ~ X, data=simulated_data)
### Estimated standard errors
sqrt(diag(vcov(reg)))
```

WORKING EXAMPLE - REGRESSION

We know that the OLS standard error estimator is biased and inconsistent under heteroskedasticity. Let's generate some simulated data with heteroskedasticity.

```
# Set random seed
set.seed(02138)
# Number of observations in our simulated dataset
obs <- 200
# Initialize the data frame that will store our simulated data
simulated_data <- data.frame(Y=rep(NA,100), X=rep(NA, 100))
# For each observation
for (i in 1:obs){
  # Generate some simulated X value from a normal (0,1)
  x <- rnorm(1, 0, 1)
  # Generate our "regression" error but with non-constant variance
  # The variance depends on x - higher for larger absolute values of X
  u <- rnorm(1, 0, 5 + 2*(x^2))
  # Generate Y as a linear combination of X and the error
  y <- 1 + 5*x + u
  # Store in our data frame
  simulated_data[i,] <- c(y, x)
}
### Run Regression
reg <- lm(Y ~ X, data=simulated_data)
### Estimated standard errors
sqrt(diag(vcov(reg)))
```

For this particular dataset, we estimate $\widehat{SE(\hat{\beta}_0)} = 0.56$ and $\widehat{SE(\hat{\beta}_1)} = 0.514$

Working example - Regression

Let's imagine we can sample from the true population to use a Monte Carlo approximation to find true standard error.

Working example - Regression

Let's imagine we can sample from the true population to use a Monte Carlo approximation to find true standard error.

```
# Set random seed
set.seed(02138)
## Number of iterations
nIter = 10000
## Placeholder for ``true" sampling distribution
true_sampling <- matrix(ncol=2, nrow=nIter)
for(iter in 1:nIter){
  # Number of observations in our simulated dataset
  obs <- 200
  # Initialize the data frame that will store our simulated data
  simulated_data <- data.frame(Y=rep(NA,200), X=rep(NA, 200))
  # For each observation
  for (i in 1:obs){
    # Generate some simulated X value from a normal (0,1)
    x <- rnorm(1, 0, 1)
    # Generate our "regression" error but with non-constant variance
    u <- rnorm(1, 0, 5 + 2*(x^2))
    # Generate Y as a linear combination of X and the error
    y <- 1 + 5*x + u
    # Store in our data frame
    simulated_data[i,] <- c(y, x)
  }
  ### Run Regression
  reg <- lm(Y ~ X, data=simulated_data)
  ### Store the point estimates
  true_sampling[iter,] <- reg$coefficients
}
### True SEs
apply(true_sampling, 2, sd)
```

Working example - Regression

- From the simulation, the true $SE(\hat{\beta}_0) \approx 0.535$ while the true $SE(\hat{\beta}_1) \approx 0.855$.

Working example - Regression

- From the simulation, the true $SE(\hat{\beta}_0) \approx 0.535$ while the true $SE(\hat{\beta}_1) \approx 0.855$. So while our estimate was pretty close for the standard error of the intercept, it is *way off* for the slope!.

Working example - Regression

Can the bootstrap do any better? First, let's make our sample again.

```
# Set random seed
set.seed(02138)
# Number of observations in our simulated dataset
obs <- 200
# Initialize the data frame that will store our simulated data
single_sample <- data.frame(Y=rep(NA,200), X=rep(NA, 200))
# For each observation
for (i in 1:obs){
  # Generate some simulated X value from a normal (0,1)
  x <- rnorm(1, 0, 1)
  # Generate our "regression" error but with non-constant variance
  # The variance depends on x - higher for larger absolute values of X
  u <- rnorm(1, 0, 5 + 2*(x^2))
  # Generate Y as a linear combination of X and the error
  y <- 1 + 5*x + u
  # Store in our data frame
  single_sample[i,] <- c(y, x)
}
```

Working example - Regression

Now, let's re-sample from the sample (not the population) and store our regression estimates

```
## Number of iterations
nIter <- 20000
## Placeholder for bootstrapped sampling distribution
bootstrap_sampling <- matrix(ncol=2, nrow=nIter)
# For each observation
for (iter in 1:nIter){
  ### Sample n indices of ``single_sample "With replacement!
  index <- sample(1:nrow(single_sample), nrow(single_sample), replace=T)
  ### Create bootstrapped dataset
  boot_data <- single_sample[index,]
  ### Run Regression
  reg <- lm(Y ~ X, data=boot_data)
  ### Store the point estimates
  bootstrap_sampling[iter,] <- reg$coefficients
}
### Bootstrap SEs
apply(bootstrap_sampling, 2, sd)
```

WORKING EXAMPLE - REGRESSION

Now, let's re-sample from the sample (not the population) and store our regression estimates

```
## Number of iterations
nIter <- 20000
## Placeholder for bootstrapped sampling distribution
bootstrap_sampling <- matrix(ncol=2, nrow=nIter)
# For each observation
for (iter in 1:nIter){
  ### Sample n indices of ``single_sample "With replacement!
  index <- sample(1:nrow(single_sample), nrow(single_sample), replace=T)
  ### Create bootstrapped dataset
  boot_data <- single_sample[index,]
  ### Run Regression
  reg <- lm(Y ~ X, data=boot_data)
  ### Store the point estimates
  bootstrap_sampling[iter,] <- reg$coefficients
}
### Bootstrap SEs
apply(bootstrap_sampling, 2, sd)
```

Now we get estimates $\widehat{SE(\hat{\beta}_0)} = 0.551$ and $\widehat{SE(\hat{\beta}_1)} = 0.782$. Much better!

When the bootstrap fails

- Really small samples – In-sample distribution is a bad estimate of the truth

When the bootstrap fails

- Really small samples – In-sample distribution is a bad estimate of the truth
- High-dimensionality – Regressions with number of covariates close to the sample size are going to run into trouble as the sample distribution becomes a poor approximation for the truth (very few observations for any given combination of covariates).

When the bootstrap fails

- Really small samples – In-sample distribution is a bad estimate of the truth
- High-dimensionality – Regressions with number of covariates close to the sample size are going to run into trouble as the sample distribution becomes a poor approximation for the truth (very few observations for any given combination of covariates).
- Correlations across observations – Can be fixed by sampling in "blocks" or "clusters"

When the bootstrap fails

- Really small samples – In-sample distribution is a bad estimate of the truth
- High-dimensionality – Regressions with number of covariates close to the sample size are going to run into trouble as the sample distribution becomes a poor approximation for the truth (very few observations for any given combination of covariates).
- Correlations across observations – Can be fixed by sampling in "blocks" or "clusters"
- Estimates of extrema (e.g. maximum of a distribution).

When the bootstrap fails

- Really small samples – In-sample distribution is a bad estimate of the truth
- High-dimensionality – Regressions with number of covariates close to the sample size are going to run into trouble as the sample distribution becomes a poor approximation for the truth (very few observations for any given combination of covariates).
- Correlations across observations – Can be fixed by sampling in "blocks" or "clusters"
- Estimates of extrema (e.g. maximum of a distribution).

COMMON QUESTIONS

- **Why do I need the size of each resample to be the size of my sample? Can't I set it to anything?**

Common questions

- **Why do I need the size of each resample to be the size of my sample? Can't I set it to anything?** Yes, but then you're approximating a different sampling distribution, not the one your estimate came from. You calculated your estimate using a sample of size n, so your bootstrap samples should be the same size!
- **Why with replacement when we took our original sample without replacement?**

Common questions

- **Why do I need the size of each resample to be the size of my sample? Can't I set it to anything?** Yes, but then you're approximating a different sampling distribution, not the one your estimate came from. You calculated your estimate using a sample of size n, so your bootstrap samples should be the same size!
- **Why with replacement when we took our original sample without replacement?** So the draws are independent!

COMMON QUESTIONS

- **Why do I need the size of each resample to be the size of my sample? Can't I set it to anything?** Yes, but then you're approximating a different sampling distribution, not the one your estimate came from. You calculated your estimate using a sample of size n, so your bootstrap samples should be the same size!
- **Why with replacement when we took our original sample without replacement?** So the draws are independent!If we sampled n observations without replacement, we would only ever get 1 unique sample!

COMMON QUESTIONS

- **Why do I need the size of each resample to be the size of my sample? Can't I set it to anything?** Yes, but then you're approximating a different sampling distribution, not the one your estimate came from. You calculated your estimate using a sample of size n, so your bootstrap samples should be the same size!
- **Why with replacement when we took our original sample without replacement?** So the draws are independent!If we sampled n observations without replacement, we would only ever get 1 unique sample! Technically, sampling without replacement from the population also induces dependence, but its negligible when the population is really large.

QUESTIONS

Questions?