Writing Functions

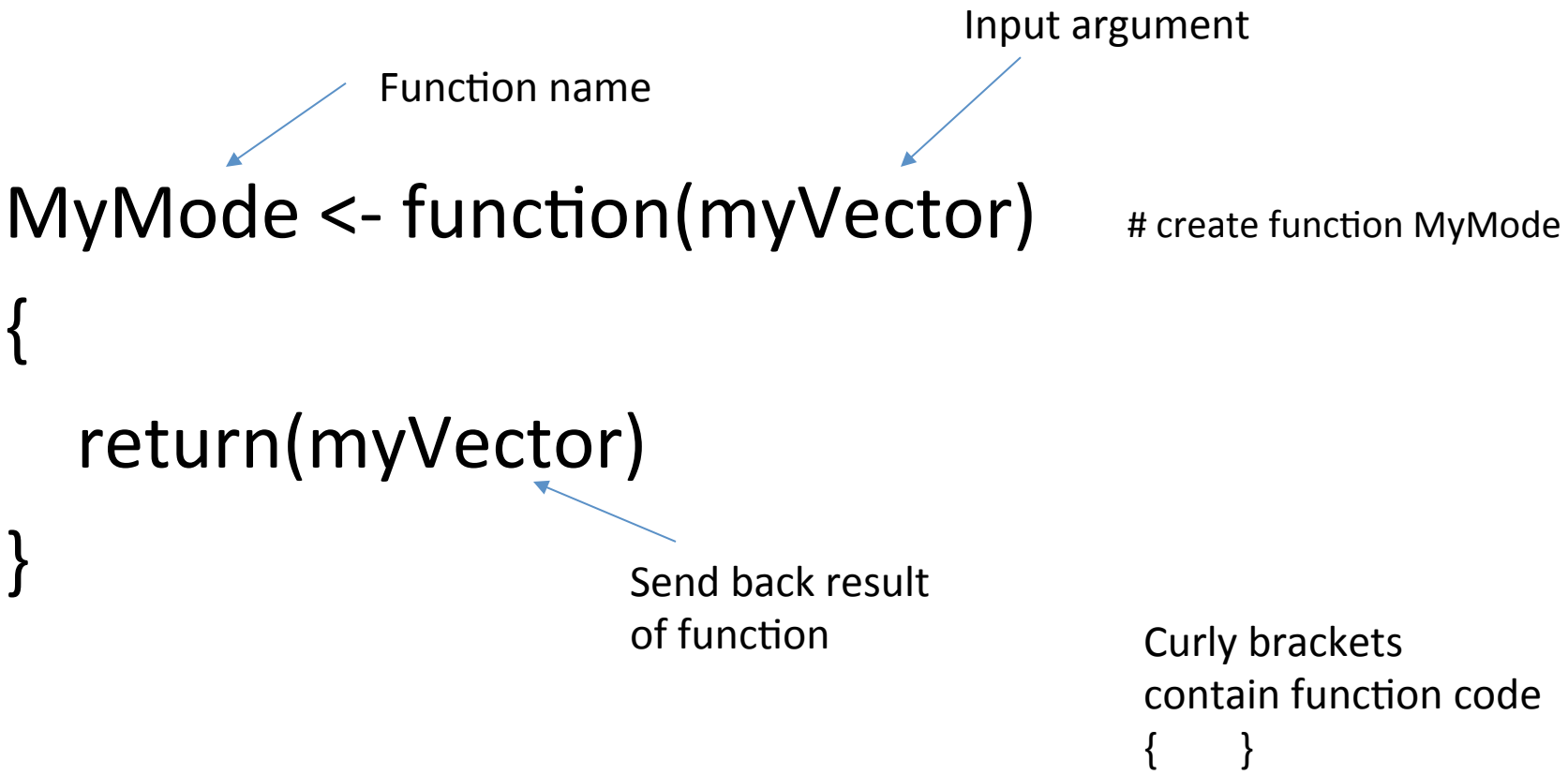
Using R Functions

- R functions
  - Function from previous chapters

    → mean is an example (used on a vector)
  - Format
    - Function name: **mean**
    - Argument/input to function: any vector
    - Function **mean** returns the **mean** value of that vector
  - Bundle of code that can be used over again without retyping

Writing R Functions

Writing a 'MyMode' function:



Writing a MyMode Function

```
> tinyData <- c(1,2,1,2,3,3,3,4,5,4,5)   # create vector tinyData
> tinyData                               # display content of tinyData
[1] 1 2 1 2 3 3 3 4 5 4 5

> MyMode(tinyData)                       #pass argument tinyData to function MyMode
[1] 1 2 1 2 3 3 3 4 5 4 5                # MyMode function designed to display
                                           content of argument passed to it
```

Writing a MyMode Function

```
MyMode <- function(myVector)
{
uniqueValues <- unique(myVector)
return(uniqueValues)
}
```

add unique function to MyMode
Note: return argument uniqueValues

```
> MyMode(tinyData)
[1] 1 2 3 4 5
```

execute MyMode with new unique function. Unique took out redundancies from tinyData vector passed to it.

Writing a MyMode Function

```
MyMode <- function(myVector)
{
uniqueValues <- unique(myVector)
uniqueCounts <- tabulate(myVector)
return(uniqueCounts)
}
```

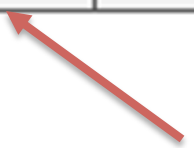
add tabulate function to MyMode
Note: uniqueCounts is being returned

```
> MyMode(tinyData)
[1] 2 2 3 2 2
```

MyMode returns count of how many times each unique value of tinyData occurs

Writing a MyMode Function

INDEX	1	2	3	4	5
uniqueValues	1	2	3	4	5
uniqueCounts	2	2	3	2	2



Most frequently occurring item is largest number in this row

Writing a MyMode Function

```
MyMode <- function(myVector)
{
uniqueValues <- unique(myVector)
uniqueCounts <- tabulate(myVector)
return(uniqueValues[which.max(uniqueCounts)])
}
```

add which.max code

return the uniqueValue that has the highest uniqueCount associated with it

```
> tinyData
[1] 1 2 1 2 3 3 3 4 5 4 5
> MyMode(tinyData)
[1] 3
```

display content of tinyData

execute MyMode

uniqueValue that has the highest uniqueCount associated with it

Testing MyMode

```
> tinyData<-c(tinyData,5,5,5)
> tinyData
[1] 1 2 1 2 3 3 3 4 5 4 5 5 5 5
> MyMode(tinyData)
[1] 5
```

add additional values to tinyData

display content of tinyData

uniqueValue that has the highest uniqueCount associated with it

Testing MyMode

```
tinyData<-c(tinyData,1,1,1)          # add additional values to tinyData
> tinyData                           # display content of tinyData
[1] 1 2 1 2 3 3 3 4 5 4 5 5 5 5 1 1 1
> MyMode(tinyData)
[1] 1                                uniqueValue that has the highest uniqueCount associated with it
```

Issue:

- tinyData contains 5 1's and 5 5's; however our MyMode function is only returning 1 as the uniqueValue with the highest uniqueCount associated with it.
- which.max returns the first maximum it finds.
- We'll see a solution to this toward the end of our discussion; stay tuned.

Testing MyMode – Finding an Error

```
> tinyData<-c(tinyData,9,9,9,9,9,9,9)  # add additional values to tinyData
> MyMode(tinyData)                     # execute MyMode
[1] NA                                 # ???????????
```

What's causing the issue?

Writing Functions

Writing Functions – Fixing MyMode

```
> tinyData<-c(tinyData,9,9,9,9,9,9,9)  # add additional values to tinyData
> MyMode(tinyData)                     # execute MyMode
[1] NA                                 # ???????????
```

What's causing the issue?

Writing Functions – Fixing MyMode

```
> tinyData<-c(tinyData,9,9,9,9,9,9,9)
> MyMode(tinyData)
[1] NA
```

```
> tabulate(tinyData)
[1] 5 2 3 2 5 0 0 0 7
```

tabulate returns count of how many times each unique value of tinyData occurs. Note 0's for 6, 7, 8 that were not present.

Writing Functions – Fixing MyMode

```
> tinyData<-c(tinyData,9,9,9,9,9,9,9)
> MyMode(tinyData)
[1] NA
```

```
> tabulate(tinyData)
[1] 5 2 3 2 5 0 0 0 7
```

tabulate returns count of how many times each unique value of tinyData occurs. Note 0's for 6, 7, 8 that were not present.

```
> unique(tinyData)
[1] 1 2 3 4 5 9
```

unique returns the six unique values in tinyData. It does not match the number of elements being returned by tabulate.

Writing Functions – Fixing MyMode

```
> tinyData<-c(tinyData,9,9,9,9,9,9,9)
> MyMode(tinyData)
[1] NA
```

```
> tabulate(tinyData)
[1] 5 2 3 2 5 0 0 0 7
```

tabulate returns count of how many times each unique value of tinyData occurs. Note 0's for 6, 7, 8 that were not present.

```
MyMode <- function(myVector)
{
uniqueValues <- unique(myVector)
uniqueCounts <- tabulate(myVector)
return(uniqueValues[which.max(uniqueCounts)])
}
```

return the uniqueValue that has the highest uniqueCount associated with it

Writing Functions – Fixing MyMode

```
MyMode <- function(myVector)
{
uniqueValues <- unique(myVector)
uniqueCounts <- tabulate( match(myVector,uniqueValues))
return(uniqueValues[which.max(uniqueCounts)])
}
```

Add new code to 'tabulate' such that instead of tabulating every possible value in tinyData, including the ones for which there is no data, only tabulate those items where there is a match between the list of unique values and what is in myVector.

```
> MyMode(tinyData)
[1] 9
```

R Function – Most Frequent Value

```
> mfv(tinyData)
[1] 9
```

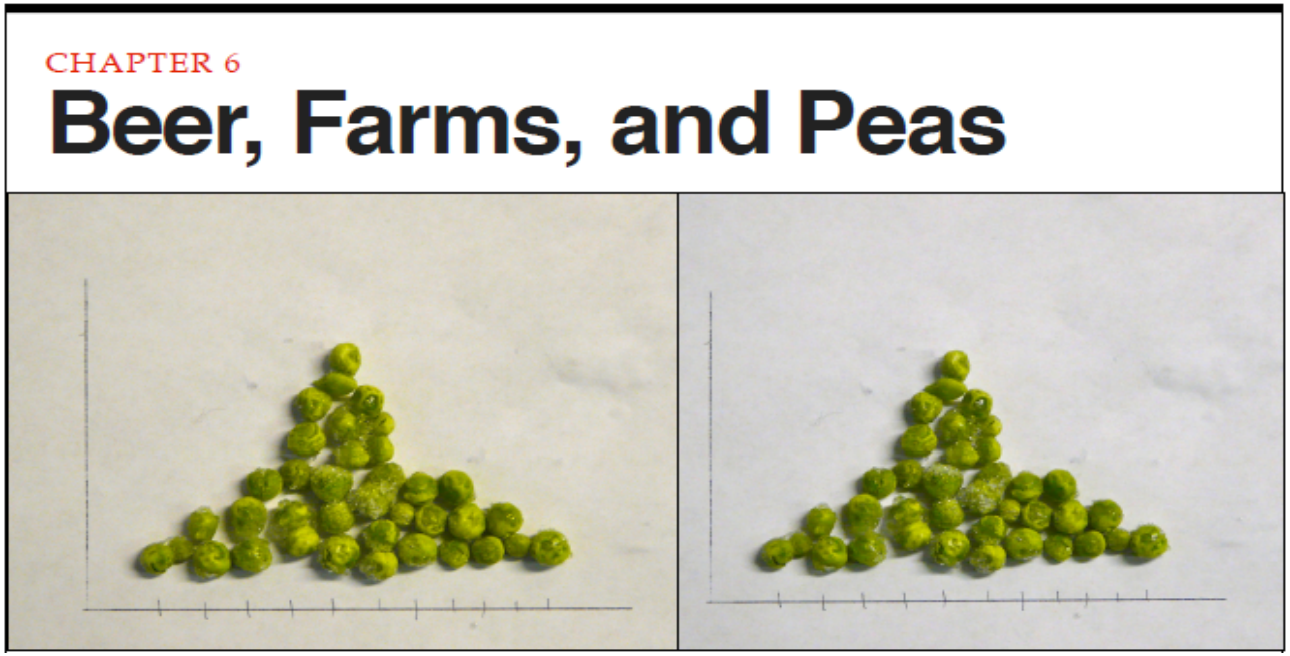
mfv – most frequent value

```
> multiData <- c(1,5,7,7,9,9,10)
> mfv(multiData)
[1] 7 9
> MyMode(multiData)
[1] 7
```

create vector and assign to multiData
note 2 7's and 2 9's
mfv accurately displays 7 and 9
our custom MyMode function only displays 7

Descriptive Stats Overview

Descriptive Stats Overview



Many of the simplest and most practical methods for summarizing collections of numbers come to us from four guys who were born in the 1800s at the start of the industrial revolution. A considerable amount of the work they did was focused on solving real-world problems in manufacturing and agriculture by using data to describe and draw inferences from what they observed.

Descriptive Stats - History

- Some history: contributions to the statistical party
  - Francis Galton: eugenics and peas
  - Karl Pearson: correlation and regression
  - William Sealy Gosset: small sample statistical techniques and beer
  - Ronald Fisher: analysis of variance and farms

Samples & Uncertainty

- Pervasive tenet: “sample of data”
- Whatever data you have there is more out there.
- Whatever data you have is just a snapshot or sample of what might be out there.
- There is always uncertainty in data, and we need to guard against putting too much stock in what a sample of data has to say.

Example Descriptive Stats

- Descriptive vs. inferential statistics
- Descriptive statistics covered previously
  - Mean
  - Median
  - Range
  - Mode
- New to this discussion
  - Variance
  - Standard deviation

Two Key Measures of Data

Measure of Central Tendency

- Mean
- Median
- Mode

Measure of Dispersion

- Range
- Variance
- Standard deviation

Question

Why is understanding central tendency and measure of dispersion useful?

Descriptive Stats Example

Example - State Populations

- Residential population data set
  - State
  - Population value
- Loaded into R
  - USstatePops$april10census

.Alaska	710,231
.Arizona	6,392,017
.Arkansas	2,915,918
.California	37,253,956
.Colorado	5,029,196
.Connecticut	3,574,097
.Delaware	897,934
.District of Columbia	601,723
.Florida	18,801,310
.Georgia	9,687,653
.Hawaii	1,360,301
.Idaho	1,567,582
.Illinois	12,830,632
.Indiana	6,483,802
.Iowa	3,046,355

Calculate Descriptive Stats Using R

R statistical functions

```
> mean(USstatePops$april10census)
[1] 6053834

> median(USstatePops$april10census)
[1] 4339367

> mode(USstatePops$april10census)
[1] "numeric”
```

Histograms

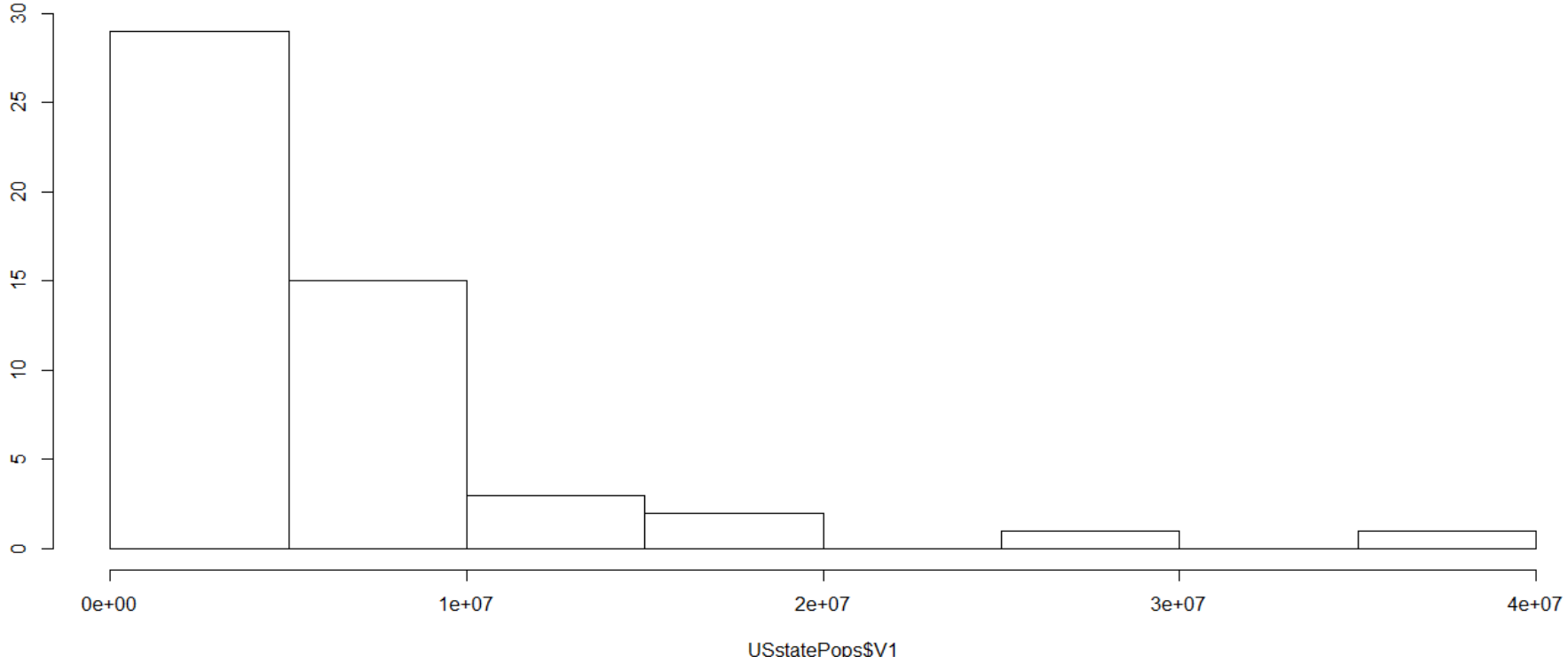
- Histogram
  - A picture that shows central tendency and dispersion

- R Function

  Hist(USstatePops$april10census)

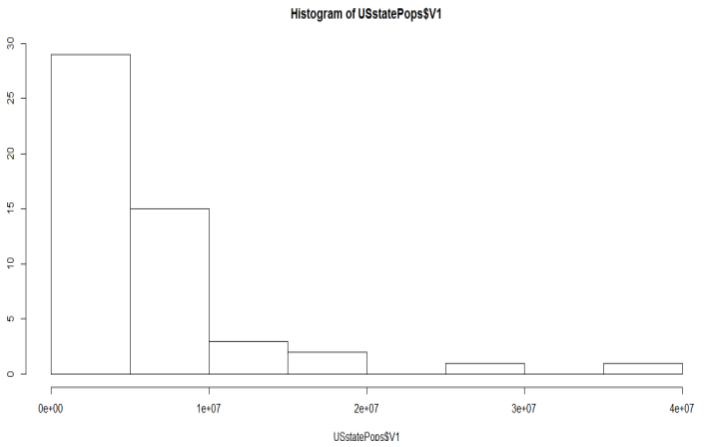
Example Histogram

Histogram: hist(USstatePops$april10census)



Example Histogram - Explained

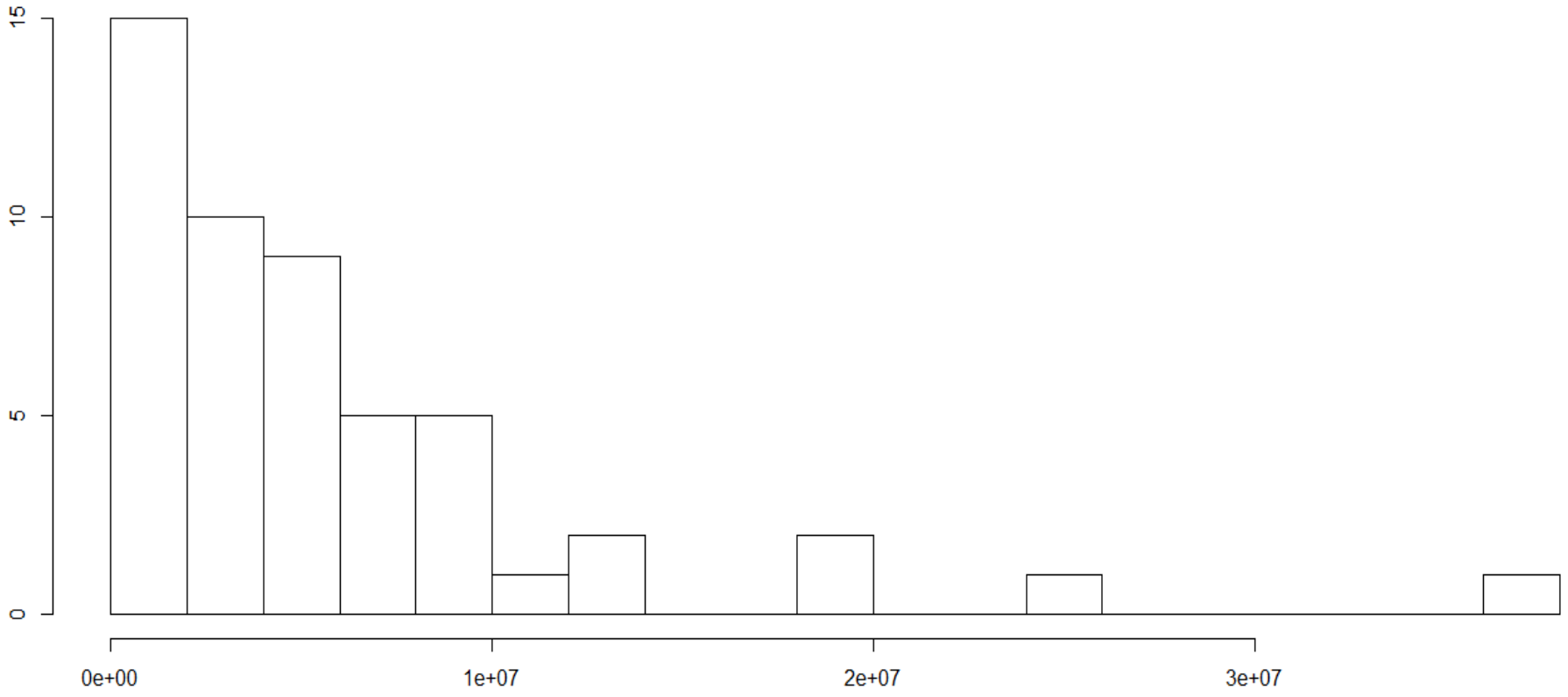
hist(USstatePops$april10census)



- Designed to show frequencies
- Interpretation
  - *x* and *y* axis
  - Two bars per tick mark
  - 30 states with populations under 5 million
  - Another 10 states with population under 10 million
  - Small number of states with populations greater than 10 million
- Observation: 40 states clustered into the first couple of bars

Example Histogram – More “breaks”

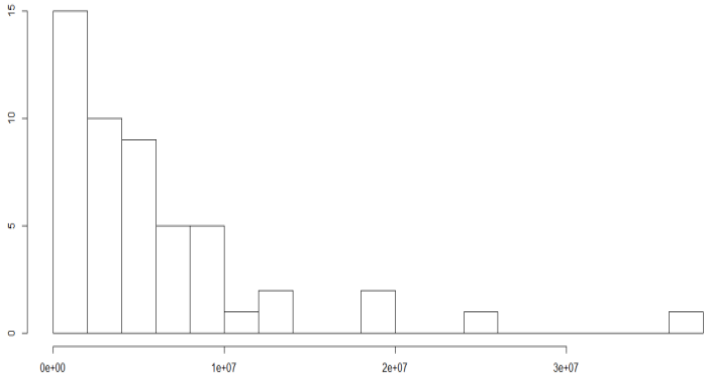
hist(USstatePops$april10census, breaks=20)



Example Histogram - Explained

hist(USstatePops$april10census, breaks=20)

- Designed to show more bars
- Interpretation
  - Five bars per tick mark
  - 15 states with populations under 2 million
  - Distribution pattern
  - Reverse-j distribution pattern



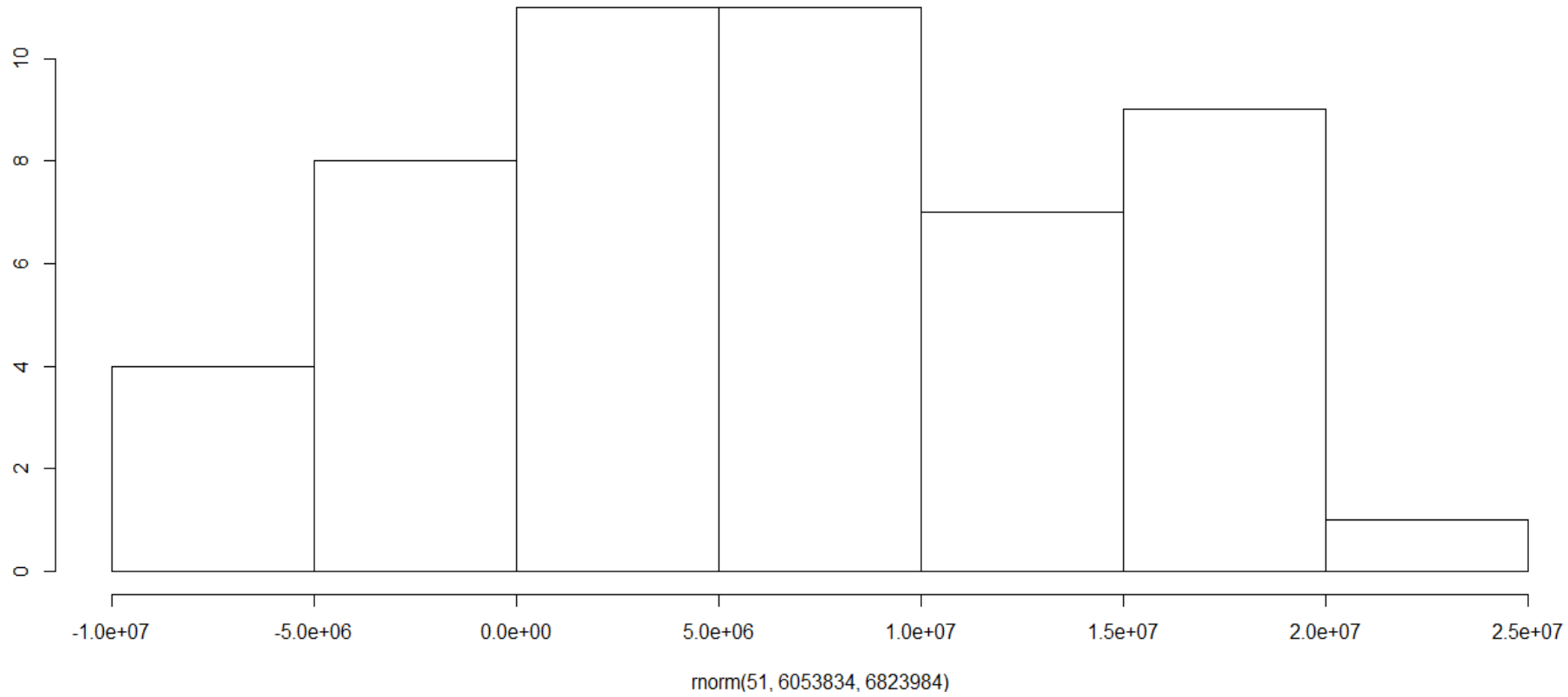
Distributions

Other distributions shapes

- The “bell” or normal distribution
- Meant to represent the most typical (normal) distribution
- Normal shape is that of a bell

Example Normal Distribution

Hist(rnorm(51, 6053834, 6823984))



Generating Normal Distributions

R components to render the normal distribution of the state population data set

- sd(USstatePops$april10census)

  6823984

- mean(USstatePops$april10census)

  605384

- 51 data points
- hist(rnorm(51, 6043834, 6823984))
- Function within a function

Generating Normal Distributions

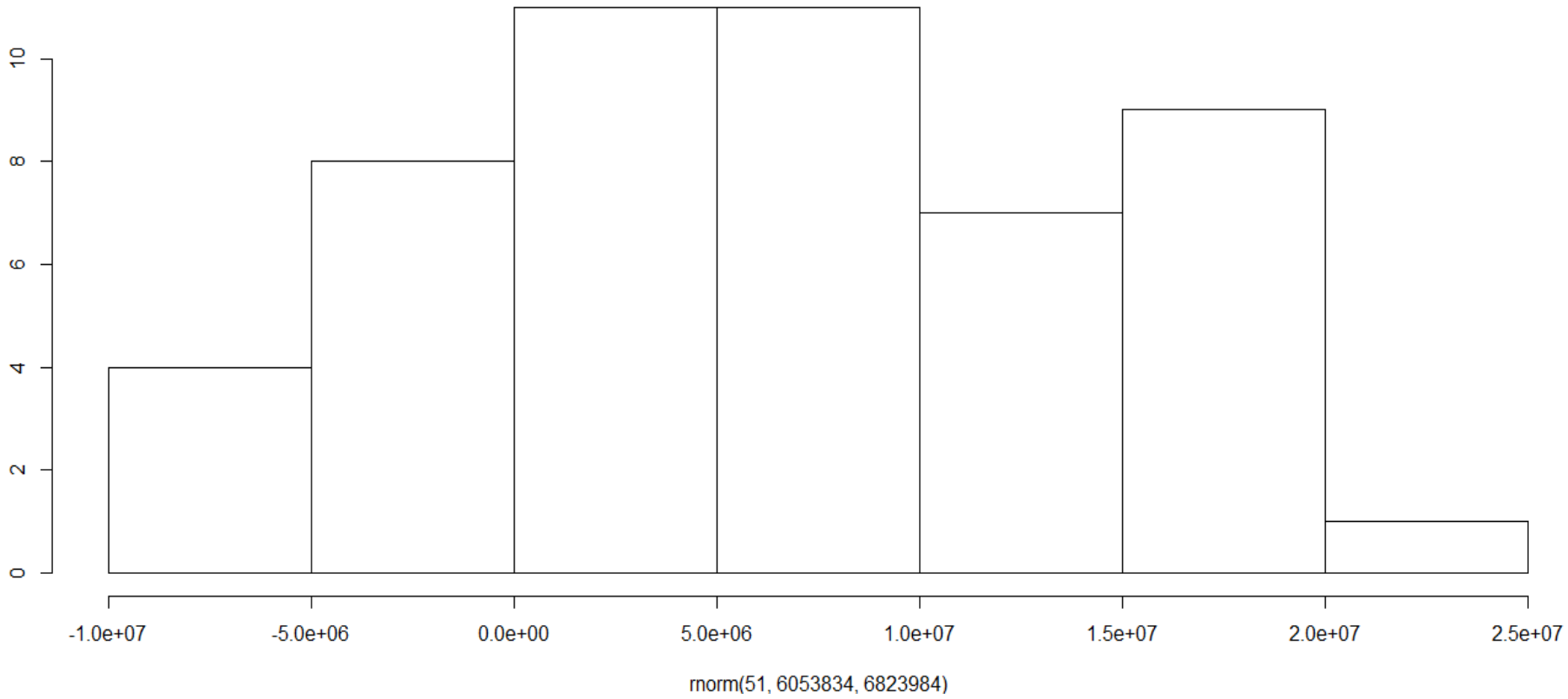
rnorm(51, 6053834, 6823984)

```
> rnorm(51,6053834, 6823984)
 [1]   2140927.6   8321903.6   7778823.9   1975788.9   9596056.1   9687365.8 -1796469.9   1731564.3   -600726.4    210103.8   7394635.1  16401844.3   -440114.7
[14]   8094555.1   8925681.5  10091285.1  10068318.1   1436260.6  14582483.8  11291498.0  14076210.0  -2962166.8  -9855689.9   1168526.9   1826509.8  13127648.3
[27]  -3398999.6   3643236.9   3841644.3  23247836.8  -3381465.9   8236009.4  10619751.7  12571853.5   2011040.6   9655436.7  10854206.4   5199112.7  10984438.5
[40]  -4681428.8   1457661.1   3361793.7  12720280.5    265577.1  -3793762.3   7048391.6 -10050883.7    931311.3  18782698.5   6231469.0   9752514.4
>
```

hist(rnorm(51, 6053834, 6823984)

Generating Normal Distributions

Hist(rnorm(51, 6053834, 6823984))



Reviewing Distributions

R takeaways

- Normal distribution is used extensively through applied statistics as a tool for making comparisons

- Key pieces of information to enable comparisons
  - The distribution had a characteristic shape
  - The distribution had a center point, mean
  - A “spread” (variability) which was the standard deviation

Real World Examples

Data Science Examples

Data Science in the real world

- Data Science is used in many Industries

- Data Science is used in many situations

Provide some examples of
"Data Science in the real world"


SYRACUSE UNIVERSITY
SUOS
CULTORES
SCIENTIA
CORONAT