STOR 320.1
Programing II

Introduction to Functions

- Most Important Programming Skill in R

- Functions in R
  - Take Inputs
  - Do Calculations
  - Produce Outputs

- Control Structures Such as "If-else" Statements and Loops are Used in Functions

- Advantages
  - Memorable Names
  - Code Updates Occur in 1 Place
  - Makes Code Accessible by All

Build-in R Functions

- Before Writing a Function, Always Search for a Function That Does What You Want

- To See What a Function Does: `?dplyr::lag`

- To Understand How the Function Works, Algorithmically: `dplyr::lag`

Build-in R Functions

```
dplyr::lag
```

```
## function (x, n = 1L, default = NA, order_by = NULL, ...)
## {
##     if (!is.null(order_by)) {
##         return(with_order(order_by, lag, x, n = n, default = default))
##     }
##     if (inherits(x, "ts")) {
##         bad_args("x", "must be a vector, not a ts object, do you want `stats::lag()`?")
##     }
##     if (length(n) != 1 || !is.numeric(n) || n < 0) {
##         bad_args("n", "must be a nonnegative integer scalar, ",
##             "not {type_of(n)} of length {length(n)}")
##     }
##     if (n == 0)
##         return(x)
##     xlen <- length(x)
##     n <- pmin(n, xlen)
##     out <- c(rep(default, n), x[seq_len(xlen - n)])
##     attributes(out) <- attributes(x)
##     out
## }
## <bytecode: 0x00000000123d4f48>
## <environment: namespace:dplyr>
```

Creating R Functions

- General Form:

```
NAME = function(INPUTS){
	ACTIONS
	return(OUTPUT)
}
```

- Functions are Objects in R
- To Call Function: `NAME(INPUTS)`
- Create an Object to Save an Output from a Function

```
OUTPUT=NAME(INPUTS)
```

Example

- Example: Lag Operator
  - Used for Vectors According to Time (i.e Time Series Data)
  - Suppose a Vector Contains Information at Time = t
  - A Lagged Vector Contains Information at Time = t-k where k = Lag
  - Suppose y_t = Value of a Car at Time t. Then, y_{t-k} = Value of a Car at Time t-k

Example

- Example: Lag Operator
  - Vector of Values

```
V = c(35, 32, 30, 31, 27, 25)
```

  - Lagged Values for k=1

```
LV1 = c(NA, 35, 32, 30, 31, 27)
```

  - Lagged Values for k=2

```
LV2 = c(NA, NA, 35, 32, 30, 31)
```

  - Want to Create a Function that:
    - Inputs Vector (x) and Lag (k)
    - Returns Lagged Vector

Creating R Functions

- Example: Lag Operator
  - Attempt 1:

```
Uptown.Func1 = function(x, k=1){
                         t = length(x)
                         y = c(rep(NA,t))
                         for(i in (k+1):t){
                             y[i] = x[i-k]
                         }
                         return(y)
}
```

  - Attempt 2:

```
Uptown.Func2 = function(x,k){
       t=length(x)
       y1=x[1:(t-k)]
       y2=c(rep(NA,k),y1)
       return(y2)
}
```

Creating R Functions

- Example: Lag Operator

```
Value=c(35, 32, 30, 31, 27, 25)
Uptown.Func1(x=Value)
```

```
## [1] NA 35 32 30 31 27
```

```
Uptown.Func2(x=Value,k=1)
```

```
## [1] NA 35 32 30 31 27
```

```
Uptown.Func1(x=Value,k=3)
```

```
## [1] NA NA NA 35 32 30
```

```
Uptown.Func2(x=Value,k=3)
```

```
## [1] NA NA NA 35 32 30
```

Practicing Functions: 5 Summary

- Computing Five Number Summary
  - Input Vector of Observations
  - Output Vector of Statistics

```
Summary.func = function(data){
      min=min(data)
      max=max(data)
      q1=quantile(data,0.25)
      q2=quantile(data,0.5)
      q3=quantile(data,0.75)
      y=c(min,q1,q2,q3,max)
      names(y)=c("Min","Q1","Q2","Q3","Max")
      return(y)
}
```

```
Summary.func(data=Ecdat::Airq$airq)
```

```
##    Min     Q1     Q2     Q3    Max
##  59.00  81.00 114.00 126.25 165.00
```

Practicing Functions: T-Test

- T-Test for Population Mean
  - Concept:
    - Null: Average # of Hours Spent Watching TV per Day is _____ in the USA
    - Alt: Average # of Hours Spent Watching TV per Day is not _____ in the USA
    - Does Data Provide Evidence that Alt is True

Practicing Functions: T-Test

- T-Test for Population Mean
  - Process:
    - Specify α (Type 1 Error)
    - Compute Test Statistic

$$t_s = \frac{\bar{x} - \mu_{Guess}}{s/\sqrt{n}}$$

    - Find P-value
    - If P-value < α, Reject Null

Building Functions

- T-Test for Population Mean

  - Inputs
    - Vector of Observations (ob)
    - Null Hypothesis (h0)
    - Alpha (a)
  - Output List
    - Test Statistic
    - P-value
    - Decision:
      - Reject
      - Fail to Reject
    - Plot Data and Null Guess

Building Functions

- T-Test for Population Mean
  - Function in R

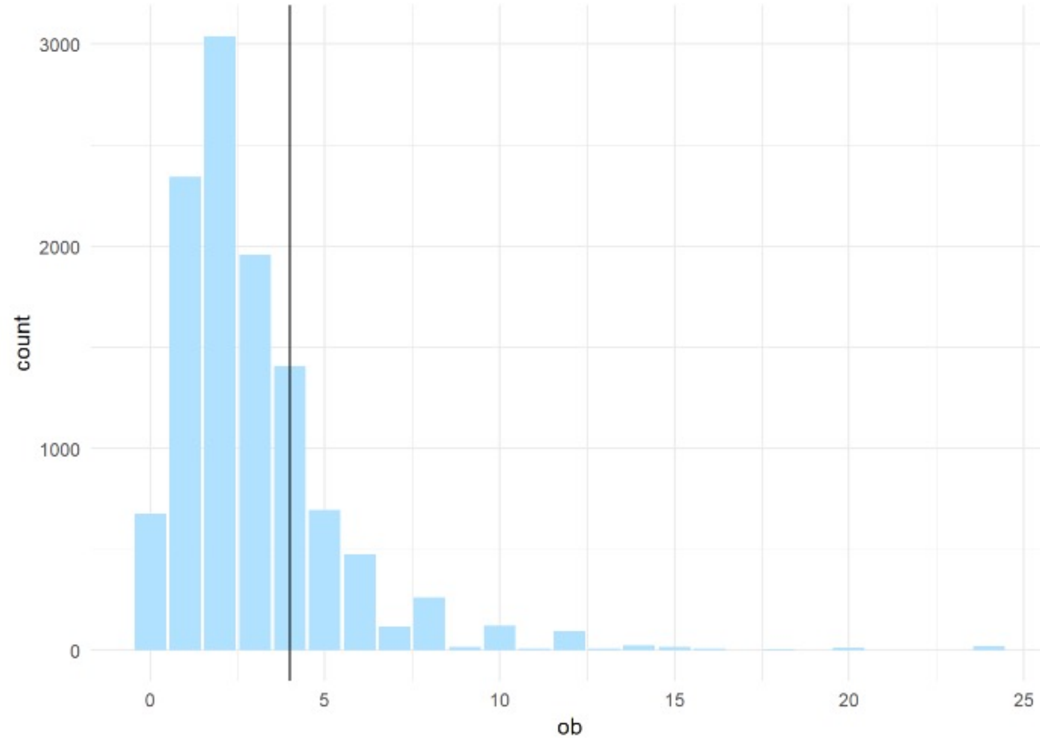
```
ttest = function(ob,h0,a){
  n=length(ob)
  ts=(mean(ob,na.rm=T)-h0)/(sd(ob,na.rm=T)/sqrt(n))
  pval=2*pt(-abs(ts),df=n-1)
  conclusion = if(pval<a){
                  "Reject Null Hypothesis"
                } else{
                  "Fail to Reject Null Hypothesis"
                }
  plot=ggplot() +
    geom_bar(aes(x=ob),fill="lightskyblue1") +
    theme_minimal() + geom_vline(xintercept=h0)
  return(list(ts=ts,pval=pval,
         conclusion=conclusion,plot=plot))
}
```

Results

- T-Test for Population Mean
    - Guess 4 Hours

```r
ttest(ob=forcats::gss_cat$tvhours,h0=4,a=0.05)
```

```
## $ts
## [1] -57.74276
##
## $pval
## [1] 0
##
## $conclusion
## [1] "Reject Null Hypothesis"
##
## $plot
```

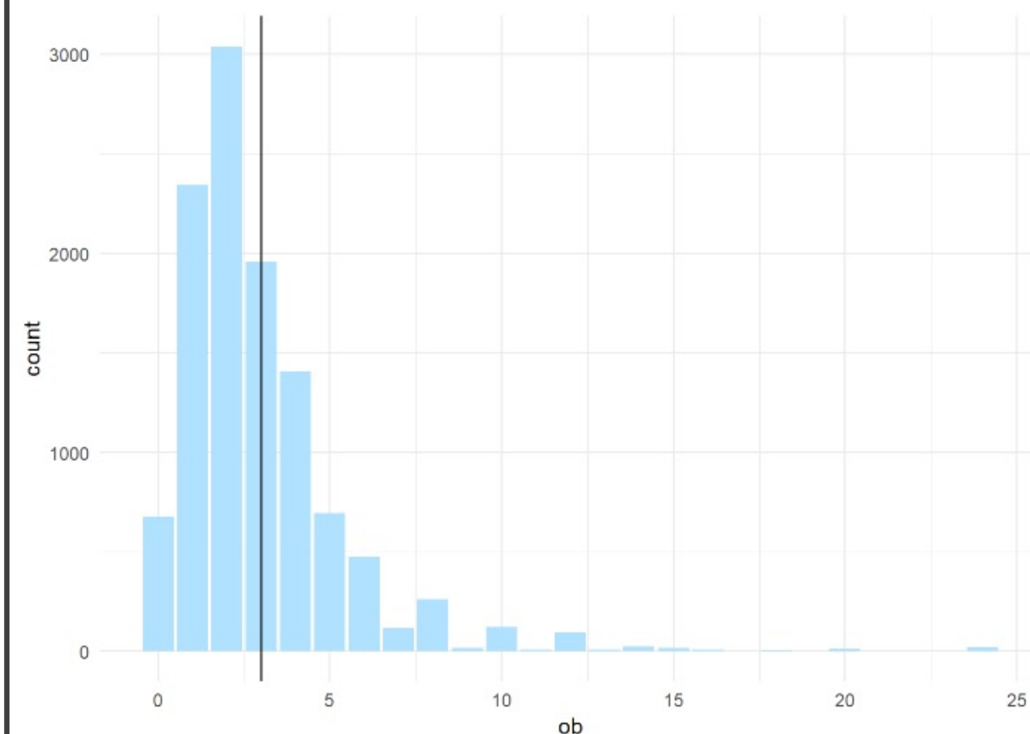


Results

- T-Test for Population Mean
  - Guess 3 Hours

```
ttest(ob=forcats::gss_cat$tvhours,h0=3,a=0.05)
```

```
## $ts
## [1] -1.089392
##
## $pval
## [1] 0.2759934
##
## $conclusion
## [1] "Fail to Reject Null Hypothesis"
##
## $plot
```



Practicing Functions: CLT

- Central Limit Theorem
  - Let *X* be a Random Variable
  - $\bar{X} \sim N\left(\mu_X, \frac{\sigma_X}{\sqrt{n}}\right)$ where
    n = sample size
  - One of the Biggest Results in Statistics
  - Foundational in Introductory Statistics Classes

Practicing Functions: CLT

- Central Limit Theorem
  - Inputs
    - n=sample size
    - S=number of simulations
    - D=distribution={1,2}
  - Output List
    - Theoretical Mean
    - Theoretical Standard Error
      $$\mathrm{SE}(\bar{X}) = \frac{\sigma_X}{\sqrt{n}}$$
    - Simulated Mean
    - Simulated Standard Error
    - Figure: Histogram of $\bar{X}$

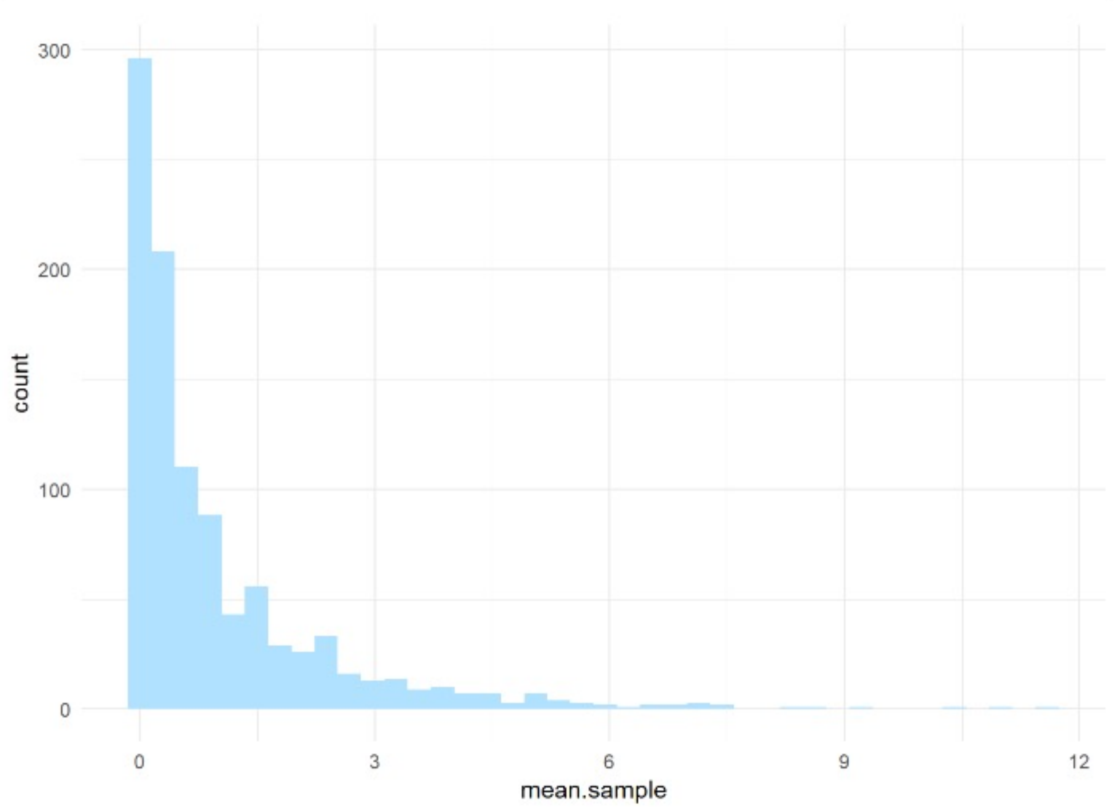
Writing Functions

```
CLT = function(n,S,D){
  if(D==1){
    initial=rnorm(1000000)
  } else if(D==2){
    initial=rgamma(1000000)
  }
  t.mean=mean(initial)
  t.se=sd(initial)/sqrt(n)

  mean.sample=rep(NA,S)
  for(k in 1:S){
    if(D==1){
      sample=rnorm(n)
    } else if(D==2){
      sample=rgamma(n)
    }
    mean.sample[k]=mean(sample)
  }
  s.mean=mean(mean.sample)
  s.se=sd(mean.sample)

  plot=ggplot()+
    geom_histogram(aes(x=mean.sample),
    fill=skyblue1)+theme_minimal()

  OUT=list(theory.mean=t.mean,
           theory.se=t.se,
           sim.mean=s.mean,
           sim.se=s.se,
           plot=plot)
  return(OUT)
}
```

Results

- Central Limit Theorem
  - Plot of Gamma Population

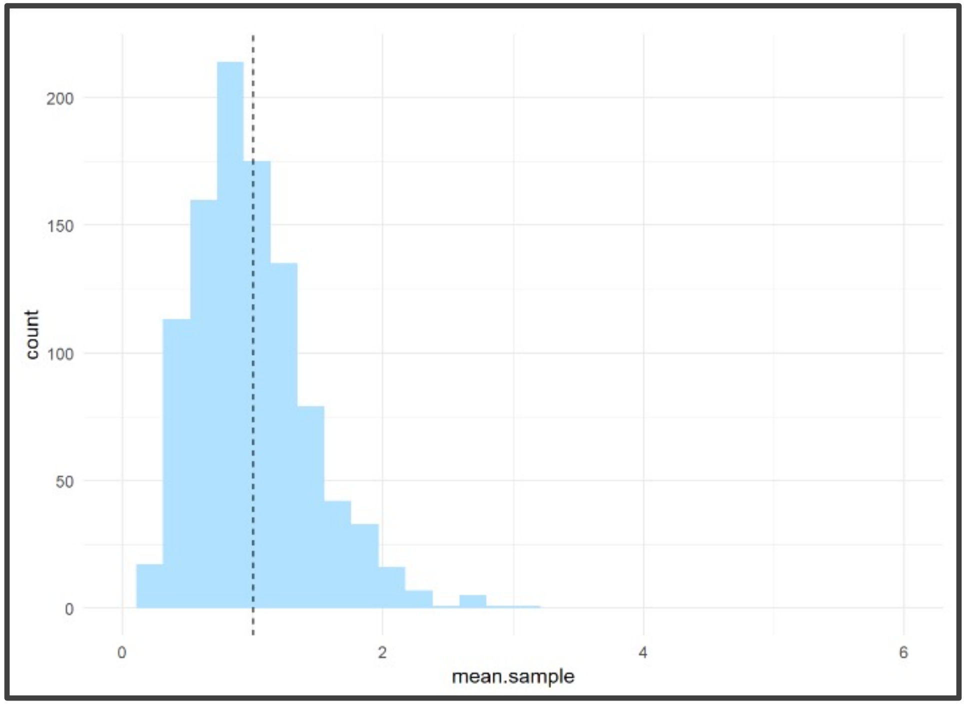
```
CLT(1,1000,D=2)$plot
```



Results: n=10

- Central Limit Theorem
  - Sampling Distribution of $\bar{X}$ when n=10

```
OUT=CLT(10,1000,D=2)
OUT[[5]]+scale_x_continuous(limits=c(0,6))+
  geom_vline(xintercept=OUT$theory.mean,linetype="dashed")
```

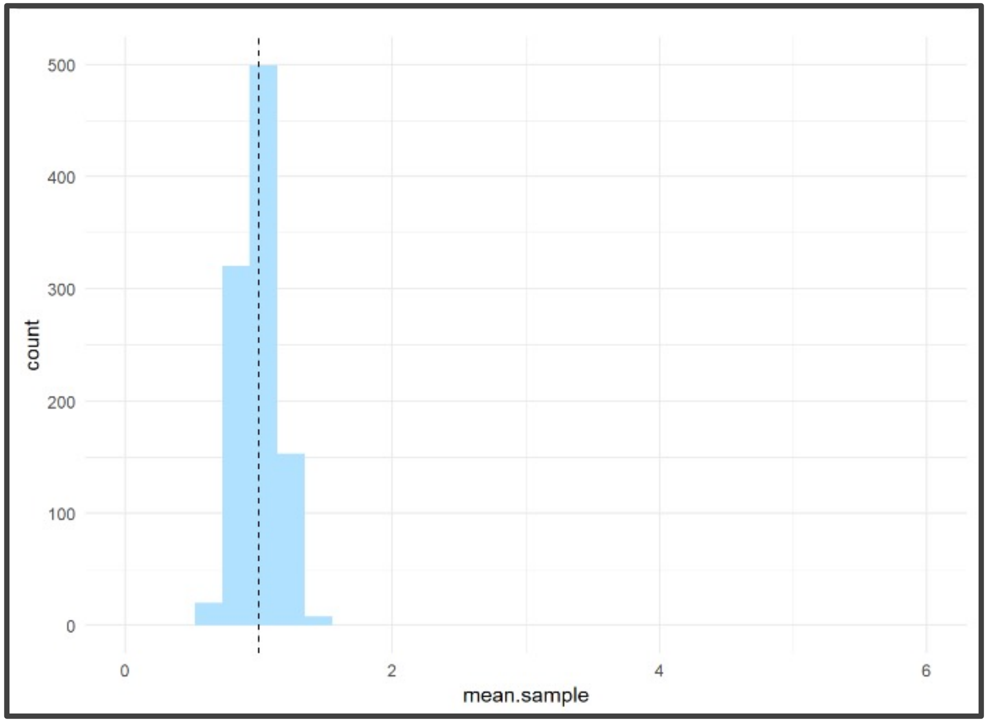



```
$theory.mean
[1] 1.001844

$theory.se
[1] 0.4472895

$sim.mean
[1] 1.031698

$sim.se
[1] 0.4547647
```

Results: n=100

- Central Limit Theorem
  - Sampling Distribution of $\bar{X}$ when n=100

```
OUT=CLT(100,1000,D=2)
OUT[[5]]+scale_x_continuous(limits=c(0,6))+
  geom_vline(xintercept=OUT$theory.mean,linetype="dashed")
```

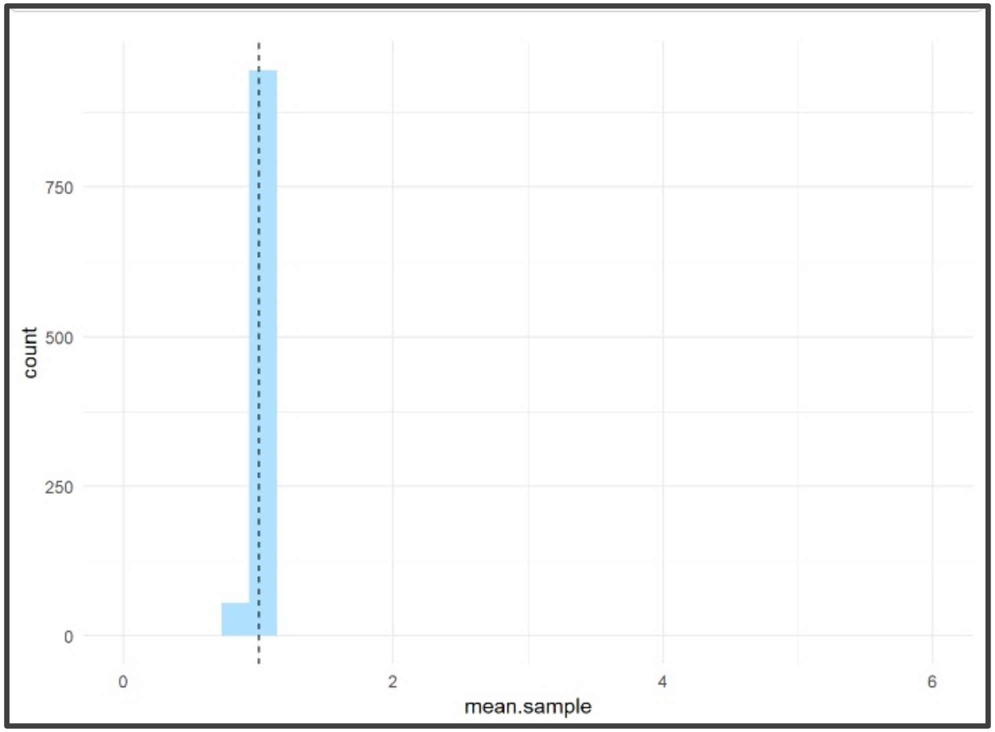



```
$theory.mean
[1] 0.999974

$theory.se
[1] 0.141634

$sim.mean
[1] 1.001891

$sim.se
[1] 0.1438765
```

Results: n=1000

- Central Limit Theorem
  - Sampling Distribution of $\bar{X}$ when n=1000

```
OUT=CLT(1000,1000,D=2)
OUT[[5]]+scale_x_continuous(limits=c(0,6))+
  geom_vline(xintercept=OUT$theory.mean,linetype="dashed")
```




```
$theory.mean
[1] 0.9992336

$theory.se
[1] 0.04454787

$sim.mean
[1] 0.9979233

$sim.se
[1] 0.04499497
```