Lecture 6

HW 2, Question 2

- There are a boxes, and slips of paper with the numbers $1, \ldots, a$. The slips of paper are randomly added to the boxes.
- Each player $i = 1, \ldots, a$ is going to try to find their slip of paper (the one with their number)
- Each player randomly selects $a/2$ boxes to open
- What is the probability that *all* players find their slip of paper when opening the boxes?

Tips on where to start

- There are a boxes, and slips of paper with the numbers $1, \ldots, a$. The slips of paper are randomly added to the boxes.

1) make a vector called boxes (a entries, one for each box)

2) `boxes <- sample(1:a, a, replace = FALSE)`

- Each player $i = 1, \ldots, a$ is going to try to find their slip of paper (the one with their number)

```
for(i in 1:a) {
    (try to find slip of paper)
}
```

- Each player randomly selects $a/2$ boxes to open

`sample(boxes, a/2, replace=F)`

- What is the probability that *all* players find their slip of paper when opening the boxes?

Another for loop! (around the whole process)

```
for (j in 1:nsim) {
        (here we play the game)


}
```

Function arguments

How could I change this function to allow the noise term to come from a different distribution?

```r
assess_coverage <- function(n, nsim, beta0, beta1){
  results <- rep(NA, nsim)

  for(i in 1:nsim){
    x <- runif(n, min=0, max=1)
    noise <- rchisq(n, 1)
    y <- beta0 + beta1*x + noise

    lm_mod <- lm(y ~ x)
    ci <- confint(lm_mod, "x", level = 0.95)

    results[i] <- ci[1] < beta1 & ci[2] > beta1
  }
  return(mean(results))
}
```

need to change

χ^2_1

Function arguments

We can also pass functions as arguments!

parameter for function to generate noise term

```r
assess_coverage <- function(n, nsim, beta0, beta1, noise_dist){
  results <- rep(NA, nsim)

  for(i in 1:nsim){
    x <- runif(n, min=0, max=1)
    noise <- noise_dist(n)
    y <- beta0 + beta1*x + noise

    lm_mod <- lm(y ~ x)
    ci <- confint(lm_mod, "x", level = 0.95)
    results[i] <- ci[1] < beta1 & ci[2] > beta1
  }
  return(mean(results))
}
```

use noise_dist argument to generate noise

```r
assess_coverage(n = 100, nsim = 1000, beta0 = 0.5, beta1 = 1,
                noise_dist = rexp)
```

just pass the name of the function to use

```
[1] 0.944
```

Simulate noise from exponential distribution

Function arguments

What must be true about the `noise_dist` function here?

```r
assess_coverage <- function(n, nsim, beta0, beta1, noise_dist){
  results <- rep(NA, nsim)

  for(i in 1:nsim){
    x <- runif(n, min=0, max=1)
    noise <- noise_dist(n)
    y <- beta0 + beta1*x + noise

    lm_mod <- lm(y ~ x)
    ci <- confint(lm_mod, "x", level = 0.95)
    results[i] <- ci[1] < beta1 & ci[2] > beta1
  }
  return(mean(results))
}
```

only passing one argument to the noise_dist function (sample size)

So: whatever I use as noise_dist needs to work with just the sample size!

Function arguments

```r
assess_coverage <- function(n, nsim, beta0, beta1, noise_dist){
  results <- rep(NA, nsim)

  for(i in 1:nsim){
    x <- runif(n, min=0, max=1)
    noise <- noise_dist(n)
    y <- beta0 + beta1*x + noise

    lm_mod <- lm(y ~ x)
    ci <- confint(lm_mod, "x", level = 0.95)
    results[i] <- ci[1] < beta1 & ci[2] > beta1
  }
  return(mean(results))
}
```

```r
assess_coverage(n = 100, nsim = 1000, beta0 = 0.5, beta1 = 1,
                noise_dist = rchisq)
```

```
Error in noise_dist(n): argument "df" is missing, with no default
```

when I call assess_coverage(...., noise_dist= rchisq) :

rchisq(n) ← this doesn't work, b/c we never specified df

Function defaults

```r
?rexp
```

Description

Density, distribution function, quantile function and random generation for the exponential distribution with rate `rate` (i.e., mean `1/rate`).

Usage

```r
rexp(n, rate = 1)
```

- The *default* value of `rate` is 1!

By default, R assumes rate=1

rexp(n)

$\Updownarrow$

rexp(n, rate=1)

Function defaults

`rexp(n, rate = 1)`

The *default* value of `rate` is 1!

Same results:

```r
set.seed(93)
rexp(n=1)
```

```
[1] 1.188317
```

```r
set.seed(93)
rexp(n=1, rate=1)
```

```
[1] 1.188317
```

Different result:

```r
set.seed(93)
rexp(n=1, rate=2)
```

```
[1] 0.5941585
```

Function defaults

```r
?rchisq
```

Usage

```r
rchisq(n, df, ncp = 0)
```

- There is no default for `df` in the `rchisq` function!

```r
assess_coverage(n = 100, nsim = 1000, beta0 = 0.5, beta1 = 1,
                noise_dist = rchisq)
```

```
Error in noise_dist(n): argument "df" is missing, with no default
```

- How can we use a χ^2_1 for `noise_dist`?

if we do noise_dist = $\underbrace{\texttt{rchisq(n, df=1)}}_{\text{no longer a function! (vector of length n)}}$

Writing a new function

```r
set.seed(73)

chisq_1 <- function(m){
  return(rchisq(m, df=1))
}
assess_coverage(n = 100, nsim = 1000, beta0 = 0.5, beta1 = 1,
                noise_dist = chisq_1)
```

```
[1] 0.962
```

takes in a single argument (sample size)

always sampling with df=1

now use new function as the argument in noise_dist

noise_dist = chisq_1

n=100

⋮

noise <- noise_dist(n)

↳ noise <- chisq_1(n)

⇒ noise <- chisq_1(100)

chisq_1(5) ← etums 5 observations randomly chosen from χ^2_1

Function scoping

What value will the following code return?

```r
g01 <- function(x = 10) {
  return(x)
}

g01()
```

Function scoping

What value will the following code return?

```r
g01 <- function(x = 10) {
  return(x)
}

g01()
```

```
[1] 10
```

What if I try to look at `x`?

```r
x
```

Function scoping

What value will the following code return?

```r
g01 <- function(x = 10) {
  return(x)
}

g01()
```

```
[1] 10
```

What if I try to look at `x`?

```r
x
```

```
Error in eval(expr, envir, enclos): object 'x' not found
```

- Variables created within functions don't exist outside the function!

Function scoping

Variables created within functions don’t exist outside the function!

```r
g01 <- function() {
  x <- 10
  return(x)
}

g01()
```

```
[1] 10
```

```r
x
```

```
Error in eval(expr, envir, enclos): object 'x' not found
```

Function scoping

What will the following code return?

```r
x <- 10

g01 <- function(){
  return(x)
}

g01()
```

Function scoping

```r
x <- 10

g01 <- function(){
  return(x)
}

g01()
```

x is never defined

=> R looks outside and sees x = 10

(so return 10)

```
[1] 10
```

```r
x
```

```
[1] 10
```

- If a variable is not defined in a function, R looks outside the function (the *global environment*)

Name masking

What value will the following code return?

```r
x <- 10
g01 <- function() {
  x <- 20
  return(x)
}

g01()
x
```

Name masking

What value will the following code return?

```r
x <- 10
g01 <- function() {
  x <- 20
  return(x)
}

g01()
```

```
[1] 20
```

```r
x
```

```
[1] 10
```

- Names defined inside a function *mask* names defined outside a function
- Variables created within a function don’t exist outside

Summary

- Variables created within a function don’t exist outside
- If a variable is not defined in a function, R looks outside the function
- Names defined inside a function *mask* names defined outside a function

Class activity

https://sta279-s24.github.io/class_activities/ca_lecture_6.html