

# Lecture 7: Lists

# Iterating over functions

So far:

```
1 set.seed(45)
2
3 # Simulate from a N(0,1)
4 assess_coverage(n = 100, nsim = 1000, beta0 = 0.5, beta1 = 1,
5               noise_dist = rnorm)
```

```
[1] 0.949
```

```
1 # Simulate from Exp(1)
2 assess_coverage(n = 100, nsim = 1000, beta0 = 0.5, beta1 = 1,
3               noise_dist = rexp)
```

```
[1] 0.96
```

```
1 # Simulate from chisquare(1)
2 assess_coverage(n = 100, nsim = 1000, beta0 = 0.5, beta1 = 1,
3               noise_dist = function(m) {return(rchisq(m, df=1))})
```

```
[1] 0.946
```

What if I want to simulate from *many* distributions?

# Idea

# Vectors revisited

Vectors can contain numbers, booleans, characters, etc:

```
1 x <- c(0, 1, 2)
2 x
```

```
[1] 0 1 2
```

```
1 typeof(x)
```

```
[1] "double"
```

```
1 x <- c("a", "b", "c")
2 x
```

```
[1] "a" "b" "c"
```

```
1 typeof(x)
```

```
[1] "character"
```

The `typeof` function tells what *type* of object we have

# Vectors of multiple types?

```
1 x <- c(0, 1, "a")  
2 x
```

```
[1] "0" "1" "a"
```

```
1 x[1] + 1
```

```
Error in x[1] + 1: non-numeric argument to binary operator
```

Basic vectors (called *atomic* vectors) only contain one type.

# Lists

```
1 x <- list(c(0, 1), "a")  
2 x
```

```
[[1]]
```

```
[1] 0 1
```

```
[[2]]
```

```
[1] "a"
```

# Lists

```
1 x <- list(c(0, 1), "a")  
2 x
```

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

```
[[2]]  
[1] "a"
```

```
1 x[[1]]
```

```
[1] 0 1
```

```
1 x[[1]][1]
```

```
[1] 0
```

# Lists

```
1 x <- list(c(0, 1), "a")  
2 x
```

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

```
[[2]]  
[1] "a"
```

```
1 x[[1]]
```

```
[1] 0 1
```

```
1 x[[1]][1]
```

```
[1] 0
```

```
1 typeof(x[[1]])
```

```
[1] "double"
```

```
1 x[[2]]
```

```
[1] "a"
```

```
1 typeof(x[[2]])
```

```
[1] "character"
```



# Visualizing list structure

```
1 x1 <- list(c(1, 2), c(3, 4))  
2 x1
```

```
[[1]]  
[1] 1 2
```

```
[[2]]  
[1] 3 4
```

```
1 x2 <- list(list(1, 2), list(3, 4))  
2 x2
```

```
[[1]]  
[[1]][[1]]  
[1] 1
```

```
[[1]][[2]]  
[1] 2
```

```
[[2]]  
[[2]][[1]]  
[1] 3
```

```
[[2]][[2]]  
[1] 4
```



# Indexing lists

```
1 x <- list(c(1, 2), c(3, 4))  
2  
3 x[1]
```

```
[[1]]  
[1] 1 2
```

```
1 typeof(x[1])
```

```
[1] "list"
```

```
1 x[[1]]
```

```
[1] 1 2
```

```
1 typeof(x[[1]])
```

```
[1] "double"
```

- `x[1]` returns a *list* which contains the first component of `x`
- `x[[1]]` returns the object stored in the first component

# Indexing lists

```
1 x <- list(list(1, 2), list(3, 4))  
2 x[1]
```

**Question:** What will `x[1]` return?

# Indexing lists

```
1 x <- list(list(1, 2), list(3, 4))  
2 x[1]
```

```
[[1]]
```

```
[[1]][[1]]
```

```
[1] 1
```

```
[[1]][[2]]
```

```
[1] 2
```

# Indexing lists

```
1 x <- list(list(1, 2), list(3, 4))  
2 x[[1]]
```

**Question:** What will `x[[1]]` return?

# Indexing lists

```
1 x <- list(list(1, 2), list(3, 4))  
2 x[[1]]
```

```
[[1]]
```

```
[1] 1
```

```
[[2]]
```

```
[1] 2
```

**Question:** How do I get just the 3?

# Indexing lists

```
1 x <- list(list(1, 2), list(3, 4))  
2 x[[2]][[1]]
```

```
[1] 3
```



# Vectors of functions?

Can we make a vector of *functions*?

```
1 x <- c(rexp, rnorm, function(m) {return(rchisq(m, df=1))})  
2 x
```

```
[[1]]
```

```
function (n, rate = 1)  
.Call(C_rexp, n, 1/rate)  
<bytecode: 0x7fd50901b778>  
<environment: namespace:stats>
```

```
[[2]]
```

```
function (n, mean = 0, sd = 1)  
.Call(C_rnorm, n, mean, sd)  
<bytecode: 0x7fd508c3e718>  
<environment: namespace:stats>
```

```
[[3]]
```

```
function(m) {return(rchisq(m, df=1))}
```

# Lists of functions

```
1 x <- list(rexp, rnorm, function(m) {return(rchisq(m, df=1))})
2 x[1]
```

```
[[1]]
function (n, rate = 1)
.Call(C_rexp, n, 1/rate)
<bytecode: 0x7fd50901b778>
<environment: namespace:stats>
```

```
1 x[1](10)
```

Error in eval(expr, envir, enclos): attempt to apply non-function

**Question:** Why does this cause an error?

# Lists of functions

```
1 x <- list(rexp, rnorm, function(m) {return(rchisq(m, df=1))})
2 x[[1]]
```

```
function (n, rate = 1)
.Call(C_rexp, n, 1/rate)
<bytecode: 0x7fd50901b778>
<environment: namespace:stats>
```

```
1 x[[1]](10)
```

```
[1] 1.24406908 0.07592609 0.57794348 1.02337796 0.43257139 0.73254842
[7] 1.28476853 1.47824260 1.50658414 1.71665563
```

# Iterating over functions

```
1  set.seed(45)
2
3  noise_dists <- list(rnorm, rexp,
4                      function(m) {return(rchisq(m, df=1))})
5  ci_coverage <- rep(NA, length(noise_dists))
6
7  for(i in 1:length(noise_dists)){
8    ci_coverage[i] <- assess_coverage(n = 100, nsim = 1000,
9                                     beta0 = 0.5, beta1 = 1,
10                                     noise_dist = noise_dists[[i]])
11  }
12
13  ci_coverage
```

```
[1] 0.949 0.960 0.946
```

# Class activity

[https://sta279-f23.github.io/class\\_activities/ca\\_lecture\\_7.html](https://sta279-f23.github.io/class_activities/ca_lecture_7.html)

