Lecture 3

Andrea Boskovic

2022-06-29

Reminders

1. Short lab 1 due at 11:59 PM tonight
2. Lab 1 will be open at 3PM today and due on 07/11 at 11:59 PM
3. No class on Monday (Happy Fourth of July!)
4. My OH are tomorrow 8-9 AM on Zoom

Back to Lecture 1

How do you make a table/make it look nice in R?

```
# install.packages("knitr")
# install.packages("kableExtra")
library(knitr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
my_matrix <- diag(3)
my_matrix
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```
rownames(my_matrix) <- c("First Row", "Second Row", "Third Row")
colnames(my_matrix) <- c("Column 1", "Column 2", "Column 3")
```

	Column 1	Column 2	Column 3
First Row	1	0	0
Second Row	0	1	0
Third Row	0	0	1

```
kable_styling(kable(my_matrix))
```

```
my_matrix
```

```
##            Column 1 Column 2 Column 3
## First Row         1        0        0
## Second Row        0        1        0
## Third Row         0        0        1
```

The `kableExtra` package is a really useful tool for making tables and will be useful in your projects.

Look how much nicer the `kableExtra` output is than just printing `my_matrix`.

Here's a link to explore what else this package can do: https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html

Lists

```
my_list <- list("some_numbers" = 1:5,
                "some_characters" = c("a", "b", "c"),
                "a_matrix" = diag(2))
my_list
```

```
## $some_numbers
## [1] 1 2 3 4 5
##
## $some_characters
## [1] "a" "b" "c"
##
## $a_matrix
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

Important: In R, the dollar sign ($) is used to refer to list items AND to column names.

Accessing list elements

```
# You can access list elements in three different ways
my_list$some_numbers
```

```
## [1] 1 2 3 4 5
```

```
my_list[["some_numbers"]]
```

```
## [1] 1 2 3 4 5
```

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

```
## [1] 1 2 3 4 5
```

How do you subset a list?

Subsetting a list means accessing just part of it. Our whole list is the contents of my_list:

```
my_list
```

```
## $some_numbers
## [1] 1 2 3 4 5
##
## $some_characters
## [1] "a" "b" "c"
##
## $a_matrix
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

But what if I only want to see `some_numbers` and `a_matrix`? How do I do that? This is called *subsetting*.

```
my_list[c(1,3)]
```

```
## $some_numbers
## [1] 1 2 3 4 5
##
## $a_matrix
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```
my_list[c("some_numbers", "a_matrix")]
```

```
## $some_numbers
## [1] 1 2 3 4 5
##
## $a_matrix
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

Solution: Make a vector of the list element positions that you want or the list element names.

var1	var2	var3
1	a	TRUE
2	b	FALSE
3	c	TRUE

Adding to a list

What if I want to add another element to `my_matrix`?

```
my_list$STAT302 <- "my favorite class"
my_list$good_question <- c(1, "jack", TRUE, 5000)
```

You can access the names of all your list elements with

```
names(my_list)
```

```
## [1] "some_numbers"    "some_characters" "a_matrix"        "STAT302"
## [5] "good_question"
```

Data Frames

```
my_data <- data.frame("var1" = 1:3,
                      "var2" = c("a", "b", "c"),
                      "var3" = c(TRUE, FALSE, TRUE))
my_data
```

```
##   var1 var2  var3
## 1    1    a  TRUE
## 2    2    b FALSE
## 3    3    c  TRUE
```

```
kable_styling(kable(my_data))
```

```
my_matrix
```

```
##            Column 1 Column 2 Column 3
## First Row         1        0        0
## Second Row        0        1        0
## Third Row         0        0        1
```

```
my_matrix2 <- as.data.frame(my_matrix)
#typeof(my_matrix)
#typeof(my_matrix2)
```

Subsetting Data Frames

```
# Accessing a column
my_data$var1
```

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

```
my_data[c("var1", "var2")]
```

```
##   var1 var2
## 1    1    a
## 2    2    b
## 3    3    c
```

```
# Things can get fishy depending on how you try to extract a column
my_data["var1"]
```

```
##   var1
## 1    1
## 2    2
## 3    3
```

```
my_data$var1
```

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

```
my_data[["var1"]]
```

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

Adding a column

```
my_data$STAT302 <- c("coding", "is", "fun")
#my_data$test <- c(1, 2, 3, 4)
```

```
# Preview into the tidyverse
my_data <- my_data %>%
  mutate(var1x2 = var1*2)
```

Add row names

```
rownames(my_data) <- c("Obs 1", "Obs 2", "Obs 3")
kable_styling(kable(my_data))
```

Coding Style

	var1	var2	var3	STAT302	var1x2
Obs 1	1	a	TRUE	coding	2
Obs 2	2	b	FALSE	is	4
Obs 3	3	c	TRUE	fun	6

```
# my_data %>%
#   filter(var1 > 1) %>%
#   mutate(var1x3 = var1 * 3) %>%
#   group_by()
```

Addins -> Styler -> Style Selection is a really useful tool to make your code look pretty!

```
AddValues <- function(x, y) {
  z <- 4
  return(x + y)
  z
}
AddValues(x = 5, y = 3)
```

```
## [1] 8
```

Always use return!! Try taking it out in the previous chunk and see what happens!

```
#first_day_of_the_month<-sdkfjdaaajjjjjjjjjjjjjjjjjjjasdoifjavdoifbjoisjdfoiajsdiovajadfjhakjsdc
```

Be careful of the margins!

```
a <- 2; b <- 3 # bad practice


# good practice
a <- 2
b <- 3
```