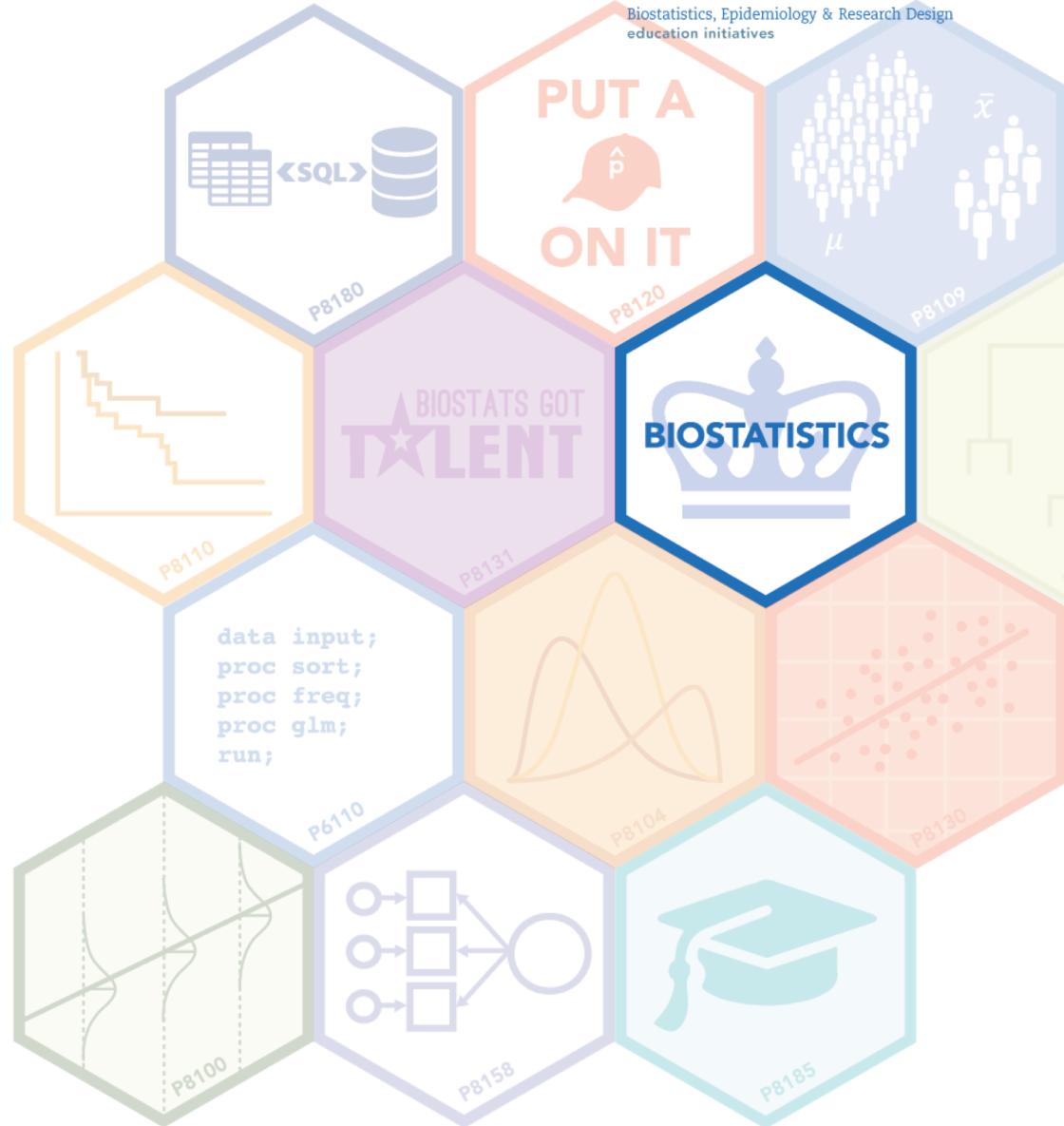


# GETTING STARTED WITH R (PART 1)

Christine Mauro, PhD  
November 19, 2018



# About Me

- Assistant Professor, Dpt. Of Biostatistics
- Teaching:
  - ReMA Quantitative Foundations (Fall, MPH Core)
  - Analysis of Categorical Data (Spring)
  - Grant Writing (Summer, CSRI)
- Research: Application of statistics to mental health, psychiatry, and health policy.
  - Diagnosis and Treatment of Complicated Grief
  - Impact of Medical Marijuana Laws on drug use
  - Impact of the Affordable Care Act among those with substance use disorders.
  - Opioid Center

# Outline

- Motivating Example
- Overview of R and R Studio
- Importing Data
  - Read CSV files using `readr` package
- Examining Data Attributes
  - Data structure, type and dimensionality
- Manipulating Data (Data Wrangling)
  - Select, Filter, Mutate, Arrange
  - Stacking and Merging

# Application

- A study was conducted to identify risk factors for low infant birth weight using data from 189 live births at Bay State Medical Center in Massachusetts. Low birthweight was defined as a <2500grams.
- We have one data set for low birthweight-babies (**lowbwt\_LOW.csv**) and another for normal birthweight babies (**lowbwt\_Normal.csv**).
  - id = ID number of infant
  - smoke = smoking during pregnancy = 1 if yes; 0 if no
  - age = mother's age in years
- We have a separate dataset with data on # of visits (**lowbwt\_ADMIN.csv**).
  - id = ID number of infant
  - visits = number of physician visits during 1st trimester = 0 if none; 1 if one; 2 if two or more

# Why R?

- R is a **FREE, open-source software used for statistical computing and graphics**
- Can be frustrating: a steep learning curve
- Installing R: <http://cran.r-project.org/> (for Windows, Mac, Linux)
- Some online resources:

<https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf>

<http://www.mayin.org/ajayshah/KB/R/index.html>

*R.D., Peng Exploratory Data Analysis With R: <https://leanpub.com/exdata>*

# What is RStudio?

- User-friendly development environment for R
- Installing RStudio: <http://www.rstudio.com/>
- Some online resources:

<http://dss.princeton.edu/training/>

<http://libguides.princeton.edu/dss>

# RStudio Windows

The screenshot shows the RStudio interface on a Windows system. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Help, and Addins. The Project dropdown shows '(None)'. The main area has four tabs: Console, Script Editor, Plots, and Environment.

**Console:** Shows R code and its output. A red box highlights the text "Console is where you execute the commands and see the output".

```

> A<-matrix(c(1,2,3,4,5,6),nrow=2, ncol=3, byrow=T)
> x<-runif(10,0,3)
> y<-runif(10, 10,100)
> plot(x,y,col=2,pch=19)
> getwd()
[1] "O:/Teaching/Regression/P8111_Chiuzan/Lecture2"
> setwd("O:/Teaching/Regression/P8111_Chiuzan/Lecture2")
>
> A<-matrix(c(1,2,3,4,5,6),nrow=2, ncol=3, byrow=T)
> x<-runif(10,0,3)
> y<-runif(10, 10,100)
> plot(x,y,col=2,pch=19)
> A
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> view(A)
> 

```

**Script Editor:** Shows an R script named "Untitled1.R" with code identical to the Console. A red box highlights the text "R script contains commands /functions to submit to the command line".

```

1
2
3 getwd()
4 setwd("O:/Teaching/Regression/P8111_Chiuzan/Lecture2")
5
6
7 A<-matrix(c(1,2,3,4,5,6),nrow=2, ncol=3, byrow=T)
8 x<-runif(10,0,3)
9 y<-runif(10, 10,100)
10 plot(x,y,col=2,pch=19)
11
12 data <- read.csv("SCBC2004.csv")
13
14 summary(data)
15 summary(data$stagen)
16
17 mean(data$stagen, na.rm=T)
18
19 q1 <- quantile(data$age, prob=c(0.025, 0.975))
20
21 age <- data$age
22 stagen <- data$stagen
23 site <- data$site
24
3:1 (Top Level) ▾

```

**Plots:** Displays a scatter plot with red dots. The x-axis is labeled "x" and ranges from 0.0 to 2.5. The y-axis is labeled "y" and ranges from 0 to 100. A red box highlights the text "Plots tab displays the graphs".

**Environment:** Shows the global environment with various objects listed. A red box highlights the text "Environment contains created objects".

Object	Type	Description
A	num [1:2, 1:3]	1 4 2 5 3 6
measure	list	211 obs. of 9 variables
ves0	list	93 obs. of 9 variables
ves1	list	74 obs. of 9 variables
ves2	list	21 obs. of 9 variables
ves3	list	23 obs. of 9 variables
Values	list	
apcs	double	0.6
bestdelta	double	0.07
defect	list	Named num [1:5] 33 19 15 10 0
foo	list	List of 29
K	double	5
level	list	num [1:3] 3 4 5
N	double	24
nlevel	double	4

**History:** Shows a list of previous commands. A red box highlights the text "History tab keeps a record of all previous commands".

```

> A<-matrix(c(1,2,3,4,5,6),nrow=2, ncol=3, byrow=T)
> x<-runif(10,0,3)
> y<-runif(10, 10,100)
> plot(x,y,col=2,pch=19)
> getwd()
[1] "O:/Teaching/Regression/P8111_Chiuzan/Lecture2"
> setwd("O:/Teaching/Regression/P8111_Chiuzan/Lecture2")
>
> A<-matrix(c(1,2,3,4,5,6),nrow=2, ncol=3, byrow=T)
> x<-runif(10,0,3)
> y<-runif(10, 10,100)
> plot(x,y,col=2,pch=19)
> A
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> view(A)
> 

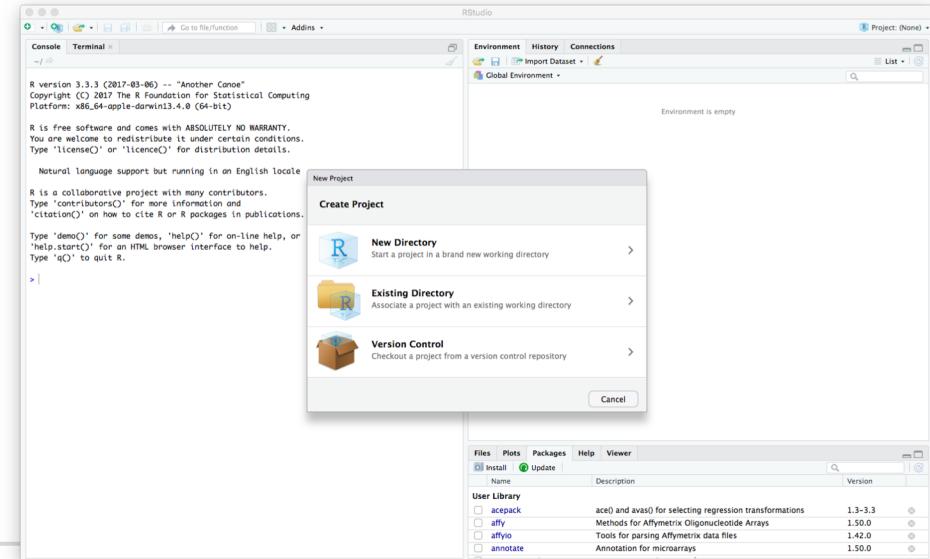
```

# R Workflow

- For every new project, do the following:
  - Create a directory with a reasonable name and path (e.g. `~/Documents/RCourse_PartI/`)
- Put an R Project in the directory
  - Create an R Project using File > New Project > Existing Directory and specifying the directory you just created.
- Keep everything related to the analysis – datasets, scripts, reports, output – in there!

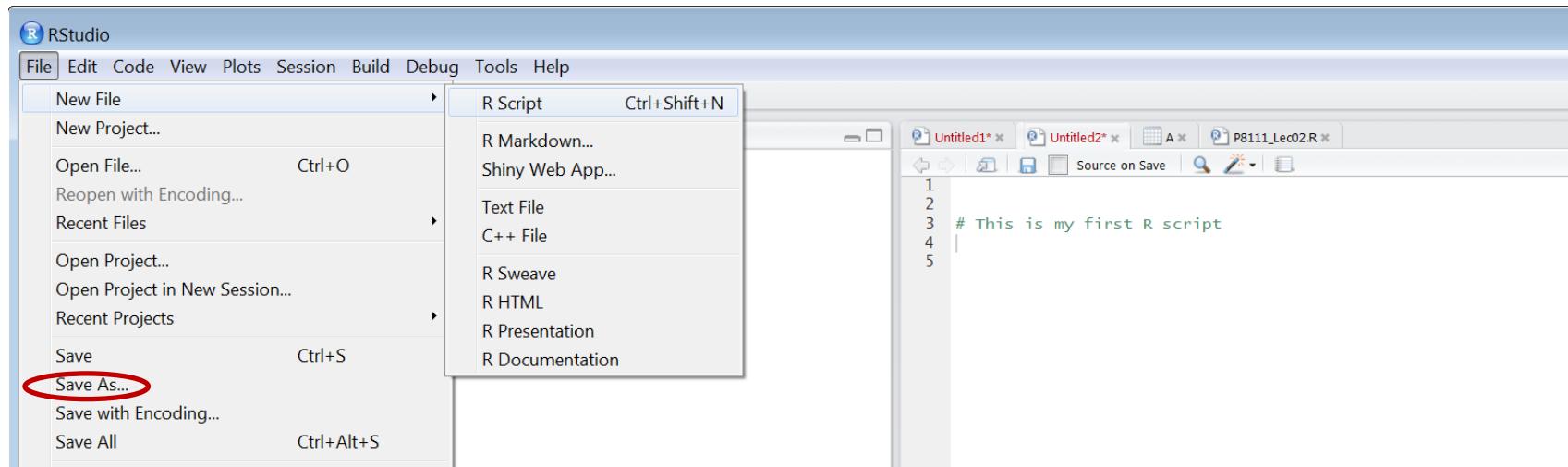
# Let's try it!

- Create a directory with a reasonable name and path (e.g. ~/Documents/RCourse\_Part1/)
- Create an R Project using File > New Project > Existing Directory and specifying the directory you just created.
- Move the three datasets for this assignment to that directory!



# RStudio - Essentials

- How to create and save an R script



- Type R commands and run them
  - Note that you can execute commands (e.g. the line with the cursor or highlighted code) in the console from a script using Command+Enter (Mac) or Ctrl+Enter (Windows).

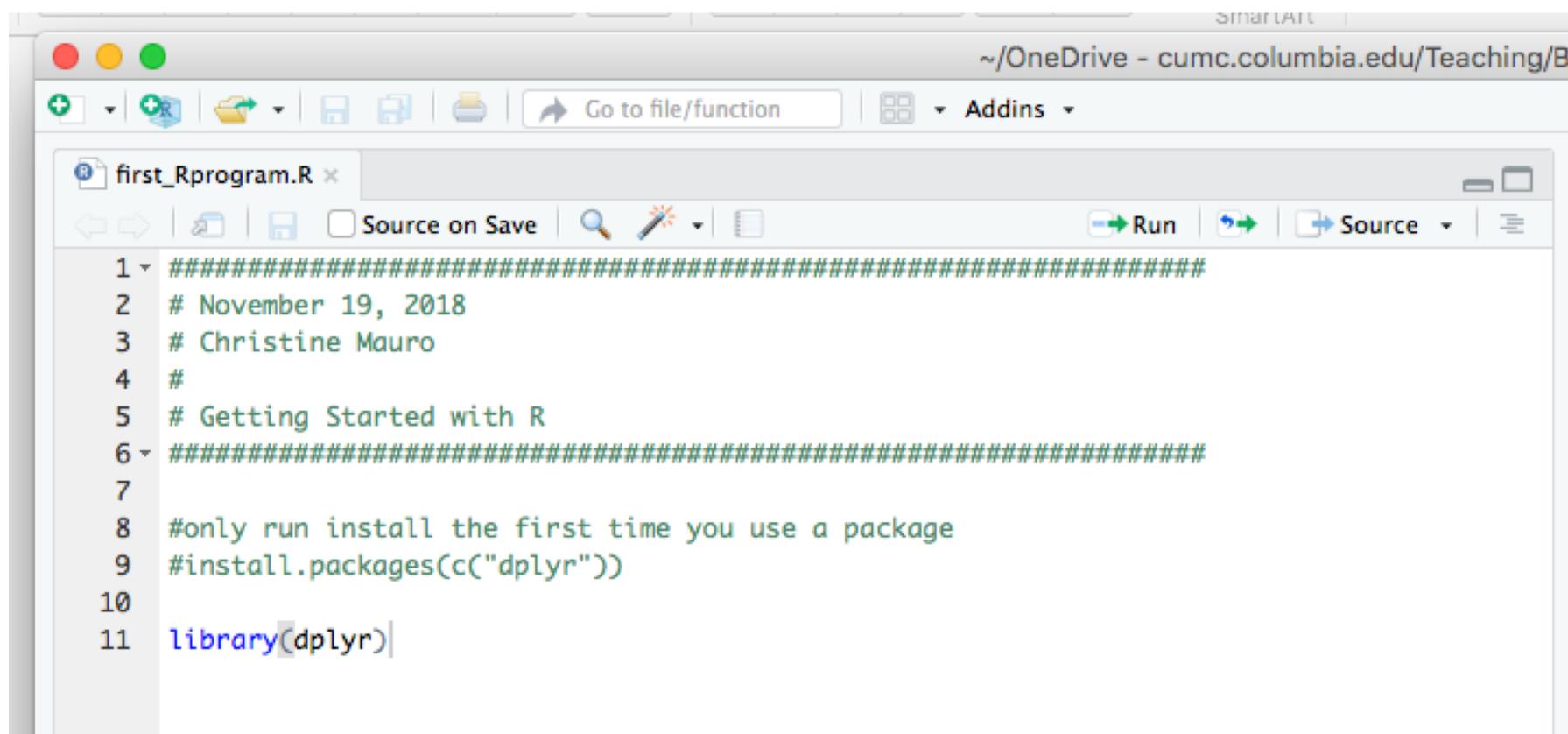
# R Packages

- Packages are collections of R functions, data, and compiled code in a well-defined format.
- The directory where packages are stored is called the library.
- R comes with a standard set of packages. Others are available for download and installation.
  - Only need to install package one time.
- Once installed, they have to be loaded into the session to be used.
  - Once installed, you need to load package every time you use R!

# Let's try it: Packages

- Start a new R script. Save the script so you have the code for later.
- Type a comment at the beginning of your program.
- Install and Load dplyr package:
  - `install.packages(c("dplyr"))`
    - This installs dplyr if you haven't used it before
    - Type y in console to any questions about installing dependencies!
  - `library(dplyr)`
    - This loads the dplyr library – repeat every time you open R!

# Let's try it: Packages



The screenshot shows the RStudio interface with a script file open. The file is titled "first\_Rprogram.R". The code in the script is as follows:

```
1 #####
2 # November 19, 2018
3 # Christine Mauro
4 #
5 # Getting Started with R
6 #####
7
8 #only run install the first time you use a package
9 #install.packages(c("dplyr"))
10
11 library(dplyr)
```

The code includes comments and two blank lines between the first and second sections. Lines 9 and 10 are commented out with a hash symbol. Line 11 is the command to load the dplyr package.

# Why dplyr?

- Most of what we are going to learn today can be done in base R using other code.
- We are using dplyr which is a part of the tidyverse.
- Why the tidyverse?
  - The tidyverse is a coherent system of packages for data manipulation, exploration and visualization that share a common design philosophy.
  - mostly developed by Hadley Wickham.
  - Tidyverse packages are intended to make statisticians and data scientists more productive by guiding them through workflows that facilitate communication, and result in reproducible work products.

# R Help

- Use tab ‘Help’ to look for a topic

E.g., look for function ‘plot’

- Tab ‘Help’ has a history of the most recent topics you inquired about

- Or just type `help(plot)` in the console
  - Or `?plot`

The screenshot shows the RStudio interface. In the top-left corner, the R logo is visible. The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Help, and Addins. Below the menu is a toolbar with various icons. The main area is the Console window, which displays R code and its output. A red arrow points from the text 'E.g., look for function “plot”' to the word 'plot' in the console output. Another red arrow points from the text 'Or just type help(plot) in the console – Or ?plot' to the 'Help' tab in the navigation bar. The 'Help' tab is highlighted with a red circle. To the right of the console is the Help pane, which shows the documentation for the 'plot' function. The title is 'Generic X-Y Plotting'. It includes sections for Description, Usage, and Arguments. The 'Arguments' section contains a note about providing coordinates or a plotting structure.

```
> detach("package:dplyr", unload=TRUE)
> library("dplyr", lib.loc="~/R/win-library/3.3")
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':
  filter, lag

The following objects are masked from 'package:base':
  intersect, setdiff, setequal, union

Warning message:
package 'dplyr' was built under R version 3.3.2
```

plot {graphics}

Generic X-Y Plotting

Description

Generic function for plotting R objects. For more details about the graphical parameter arguments, see [par](#).

For simple scatter plots, [plot.default](#) will be used. However, there are plot methods for many R objects, including [functions](#), [data.frames](#), [density](#) objects, etc. Use [methods\(plot\)](#) and the documentation for these.

Usage

```
plot(x, y, ...)
```

Arguments

- x the coordinates of points in the plot. Alternatively, a single plotting structure, function or any R object with a `plot` method can be provided.

# R Syntax

- R is an object-oriented programming language
  - If you want to save results, need to store them in an object.
- R is *case sensitive*: ‘A’ and ‘a’ are different symbols
- Commands are separated either by (;) or by a new line
  - Commands can be grouped together (in functions) by { and }
- Comments can be inserted almost anywhere
  - Starting with a ‘#’, everything to the end of the line is a comment
  - Use comments to document your code: for YOU and OTHERS!!

# R Errors

- Syntax errors - generated by misspelling or forgetting to close a bracket
- Semantic errors – correct code, but the outcome is NOT what you expected
- Logic errors – worst case! The mistake is not in the code, but the logic of execution

# IMPORTING DATA

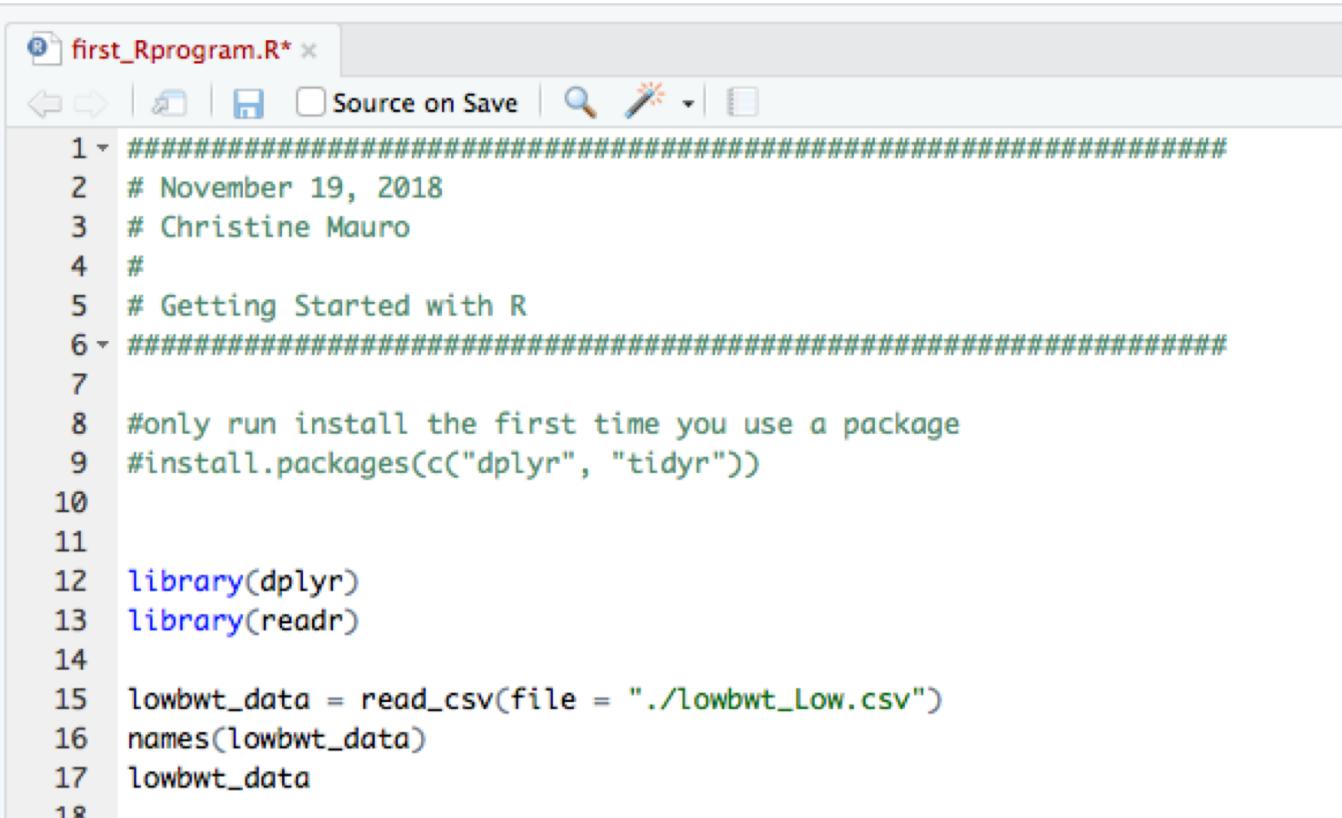
# Reading Data into R

- First, you need to save your data onto your computer
  - Excel, SPSS, or some other type of file
  - Datasets need to be in the “PROJECT” folder we created earlier
- Some useful tips:
  - Reserve the first row for headers (variable/column names)
  - First column is used to identify sampling units
  - Avoid named or fields with blank spaces; put a ‘\_’ instead.
  - Delete comments from Excel
  - Missing values should be noted with ‘.’ or ‘NA’
  - Avoid symbols such as: ‘#, ?, \*, <, /, -, }’

# Reading Data into R

- Read CSV files – if you have a ‘.csv’ file (comma separated file)

`read_csv("./Data.csv")` You need to load the `readr` package first!



The screenshot shows an RStudio interface with a code editor window titled "first\_Rprogram.R\*". The code in the editor is:

```
1 #####
2 # November 19, 2018
3 # Christine Mauro
4 #
5 # Getting Started with R
6 #####
7
8 #only run install the first time you use a package
9 #install.packages(c("dplyr", "tidyverse"))
10
11
12 library(dplyr)
13 library(readr)
14
15 lowbwt_data = read_csv(file = "./lowbwt_Low.csv")
16 names(lowbwt_data)
17 lowbwt_data
18
```

# DATA ATTRIBUTES

# Data Description

Before running any analysis, make sure you examine your data!!!

Number of variables and their types, number of observations, dates of creation, etc.

- Check variable names  
R: `names(mydata)`
- Check data dimensions: (#rows) by (# columns)
- Look at the ‘top’ and ‘bottom’ of the data  
R: `head(mydata), tail(mydata)`
- Check “structure”  
R: `str(mydata)`
- Check for missing data  
R: `anyNA(mydata)`

# Let's try it!

```
18
19 #Viewing data
20 View(lowbwt_data)
21 head(lowbwt_data)
22 tail(lowbwt_data)
23 str(lowbwt_data)
24 anyNA(lowbwt_data)
?<
```

```
> -----
> head(lowbwt_data)
# A tibble: 6 x 3
  id smoke age
  <int> <int> <int>
1 31     0    20
2 76     0    20
3 44     1    20
4 68     1    17
5 23     1    19
6 45     1    17
> tail(lowbwt_data)
# A tibble: 6 x 3
  id smoke age
  <int> <int> <int>
1 19     0    24
2 11     1    34
3 56     1    31
4 65     1    30
5 10     0    29
6 22     1    32
> str(lowbwt_data)
Classes 'tbl_df', 'tbl' and 'data.frame':      59 obs. of  3 variables:
 $ id  : int  31 76 44 68 23 45 51 49 71 83 ...
 $ smoke: int  0 0 1 1 1 1 1 0 0 0 ...
 $ age  : int  20 20 20 17 19 17 20 18 17 17 ...
 - attr(*, "spec")=List of 2
   ..$ cols  :List of 3
   ...$ id   : list()
```

# Data Description: Examples

- Tabulate your data  
R function: `table(mydata)`
- Symbol ‘\$’ is used to select a specific column from your dataset

Example: tabulate variable ‘smoke’ from data ‘low\_birth’

```
> table(lowbwt_data$smoke)

 0 1
29 30

> table(lowbwt_data$age)

14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 34
 2  2   1   5   2   3   8   5   2   5   5   6   4   2   2   1   1   1   1
```

- In data ‘low\_birth’ there are 30 subjects identified as ‘smokers’ and 29 subjects that are ‘non-smokers’.

# DATA MANIPULATION

# Operators in R

## Logical comparisons

- < for less than
- > for greater than
- <= for less than or equal to
- >= for greater than or equal to
- == for equal to each other
- != not equal to each other
- is.na()** is NA
- !is.na()** is not NA.

## Logical operators

value == 2|3; value equal 2 or () 3

&; means and. For example smoke == “0” & age > 25

# Data Transformations: R Math

>`log()` – natural logarithm

>`sqrt()` – square root

>`x^n` – exponent

## Matrix Operations:

>`A %*% B` - matrix multiplication

>`t(A)` – matrix transpose

>`det(A)` – determinant of A

>`diag(A)` – diagonal of A

>`solve(A)` – matrix inverse

# Data Manipulation (or Wrangling)

- From this point forward we will use **library(dplyr)** for data selection and manipulation
- Main Functions for data manipulation in **dplyr**:
  - I. Select
  2. Filter
  3. Mutate
  4. Arrange

# Select

- From this point forward we will use library(**dplyr**) for data selection and manipulation
- Select only certain columns (or variables in a dataset)  
R function: `select(mydata, col_name)`

## Examples:

- select only one column
- select several columns
- select all columns but one

# Let's try it: SELECT

```
--  
31 ##### Data Manipulation  
32 #Select  
33 select(lowbwt_data, id, smoke)  
34 lowbwt_data_subsetA = select(lowbwt_data, id, smoke)  
35  
36 select(lowbwt_data, -smoke)  
37 |  
--
```

```
> select(lowbwt_data, id, smoke)  
# A tibble: 59 x 2  
      id   smoke  
    <int> <int>  
 1    31     0  
 2    76     0  
 3    44     1  
 4    68     1  
 5    23     1  
 6    45     1  
 7    51     1  
 8    49     0  
 9    71     0  
10   83     0  
# ... with 49 more rows  
> select(lowbwt_data, -smoke)  
# A tibble: 59 x 2  
      id   age  
    <int> <int>  
 1    31    20  
 2    76    20  
 3    44    20  
 4    68    17  
 5    23    19  
 6    45    17  
 7    51    20  
 8    49    18  
 9    71    17  
10   83    17  
# ... with 49 more rows  
> |
```

# Renaming Variables

- Renaming a variable in a dataset can be done with:

R function: `rename(mydata, new_name_var = old_name_var)`

Example: Rename variable ‘smoke’ to ‘smoking\_status’

```
> rename(lowbwt_data, smoke_status=smoke)
# A tibble: 59 x 3
  id smoke_status age
  <int>      <int> <int>
1 31          0    20
2 76          0    20
3 44          1    20
4 68          1    17
5 23          1    19
6 45          1    17
7 51          1    20
8 49          0    18
9 71          0    17
10 83         0    17
# ... with 49 more rows
> |
```

# Filter

- Filter
  - Some data tables will include rows you don't need for your current analysis.
  - You should filter rows based on logical expressions using the filter function.
  - You will often filter using comparison operators ( $>$ ,  $\geq$ ,  $<$ ,  $\leq$ ,  $==$ , and  $!=$ ).
- Example: Suppose we only want to include mothers under 20 years of age.

# Let's try it: FILTER

```
> filter(lowbw_data, age < 20)
# A tibble: 15 x 3
  id smoke   age
  <int> <int> <int>
1 68     1     17
2 23     1     19
3 45     1     17
4 49     0     18
5 71     0     17
6 83     0     17
7 50     1     18
8 33     0     19
9 78     1     14
10 37    1     17
11 34    1     19
12 57    0     15
13 62    0     15
14 25    0     16
15 81    0     14
```

# More examples: FILTER

- What if we want moms < 20 and non-smokers??
  - `filter(lowbwt_data, age < 20 & smoke==0)`
  - `filter(lowbwt_data, age < 20, smoke==0)`
- What if we want moms < 20 OR non-smokers
  - `filter(lowbwt_data, age < 20 | smoke==0)`

# MUTATE

- Sometimes you need to change columns or create new ones.
  - You can do this using mutate.
  - NOTE: columns = variables in your data set; rows = observations in data set
- Example: apply a log transformation to skewed variables

R function: `mutate(mydata, new_name_var = transform_old_var)`

# Let's try it: Mutate

- Sometimes you want to create new variables derived from existing ones
  - E.g., apply a log transformation to skewed variables

R function: `mutate(mydata, new_name_var = transform_old_var)`

## Example:

Let's take the  
log of 'age'

```
> lowbwt_data2 = mutate(lowbwt_data, log_age=log(age))
> lowbwt_data2
# A tibble: 59 x 4
      id smoke   age log_age
      <int> <int> <int>    <dbl>
 1     31     0    20    3.00
 2     76     0    20    3.00
 3     44     1    20    3.00
 4     68     1    17    2.83
 5     23     1    19    2.94
 6     45     1    17    2.83
 7     51     1    20    3.00
 8     49     0    18    2.89
 9     71     0    17    2.83
10    83     0    17    2.83
# ... with 49 more rows
```

# Mutate Example 2

- What if we wanted to create a new binary variable to indicate age < 20?
- Need `if_else` function. General syntax:
  - `if_else(condition, value if true, value if false)`
- Let's try it!

```
> lowbwt_data2 = mutate(lowbwt_data, ageless20 = if_else(age<20, 1, 0))
> lowbwt_data2
# A tibble: 59 x 5
      id smoke   age bwt  ageless20
      <int> <int> <int> <chr>     <dbl>
    1    31     0    20  low        0
    2    76     0    20  low        0
    3    44     1    20  low        0
    4    68     1    17  low        1
    5    23     1    19  low        1
    6    45     1    17  low        1
    7    51     1    20  low        0
    8    49     0    18  low        1
    9    71     0    17  low        1
   10    83     0    17  low        1
# ... with 49 more rows
> |
```

Missing  
Data??

# ARRANGE

- Order rows of a data according to one of the variables

R function: `arrange(mydata, ordering_variable)`

## Example: order data by ‘id’

- Use function `desc()` to arrange in descending order
- See R code for ordering by multiple variables/columns

```
> #Arrange to sort data
> arrange(lowbwt_data, id)
# A tibble: 59 x 3
  id smoke age
  <int> <int> <int>
1 4     1    28
2 10    0    29
3 11    1    34
4 13    0    25
5 15    0    25
6 16    0    27
7 17    0    23
8 18    0    24
9 19    0    24
10 20   1    21
# ... with 49 more rows
|
```

# COMBINING DATASETS

# Combining Datasets (Stacking)

- Combine datasets that have the same variables but different observations
- Combine by Rows:

data 1
ID
1
4
6



data 2
ID
2
5
7



combined
ID
1
4
6
2
5
7



# Combine by Rows

- General Syntax:
  - `combined_data = bind_rows(data1, data2, data3, ...)`
- Before stacking datasets, it's helpful to create a variable to identify which data source they are coming from!
  - `mutate(data1, variable=1)`
  - `mutate(data2, variable=2)`

# Let's try it: Stacking Datasets

- Right now we only have data on low birthweights; suppose we want to combine that with data on normal birthweights.

```
58  
59 ##### stacking datasets  
60  
61 # Read in normal weight data  
62 normalbwt_data = read_csv(file = "./lowbwt_Normal.csv")  
63  
64 #create a variable to identify data source in each file  
65 lowbwt_data = mutate(lowbwt_data, bwt="low")  
66 normalbwt_data = mutate (normalbwt_data, bwt="normal")  
67  
68 #combine the data sets by stacking  
69 combined_data = bind_rows(lowbwt_data, normalbwt_data)  
70
```

# Result

```
> combined_data
# A tibble: 189 x 4
  id smoke age bwt
  <int> <int> <int> <chr>
1 31     0    20 low
2 76     0    20 low
3 44     1    20 low
4 68     1    17 low
5 23     1    19 low
6 45     1    17 low
7 51     1    20 low
8 49     0    18 low
9 71     0    17 low
10 83    0    17 low
# ... with 179 more rows
> |
```

# Merging Datasets

- Useful when you need to combine data from different sources, or at different times



data 1		data 2			combined		
ID	Age	ID	Age	Weight	ID	Age	Weight
1	15	1	15	115	1	15	115
2	20	2	20	134	2	20	134
3	18	6	22	140	3	18	.
					6	22	140

- Merging two datasets require that both have at least one variable in common (either character or numeric).
  - If character, make sure the categories have the same spelling (i.e. country names, etc.).

# Merging Datasets in R

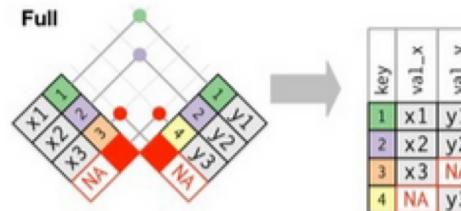
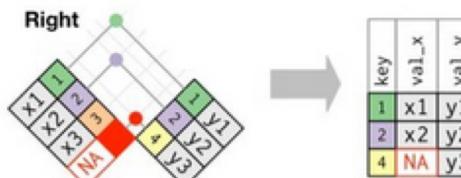
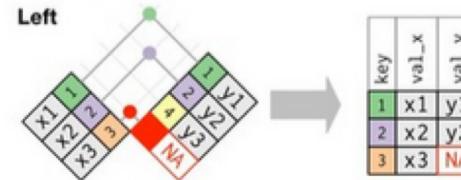
## Join types

- Joining datasets x and y

### Inner joins



### Outer joins



R for Data Science

Inner: keeps data that appear in both x and y

Left: keeps data that appear in x

Right: keeps data that appear in y

Full: keeps data that appear in either x or y

Missing  
Data??

# \* join in R

- General Syntax

- newdata = inner\_join(data1, data2, by = “ID”)
- newdata = left\_join(data1, data2, by = “ID”)
- newdata = right\_join(data1, data2, by = “ID”)
- newdata = full\_join(data1, data2, by = “ID”)

# Let's try it: Merging datasets

- Suppose we'd like to bring in information on # of visits during the first trimester from another administrative data source.

```
71
72 ##### merging datasets
73
74 #Read in administrative data set
75 admin_data = read_csv(file = "./lowbwt_ADMIN.csv")
76 admin_data
77
78 combined_data
79
80 merged_data = full_join(combined_data, admin_data, by = "id")
81 merged_data
82
```

# Results and write\_csv

```
> merged_data
# A tibble: 189 x 5
  id smoke age bwt visits
  <int> <int> <int> <chr> <int>
1 31     0    20 low      0
2 76     0    20 low      2
3 44     1    20 low      0
4 68     1    17 low      2
5 23     1    19 low      0
6 45     1    17 low      0
7 51     1    20 low      0
8 49     0    18 low      0
9 71     0    17 low      2
10 83    0    17 low      0
# ... with 179 more rows
-|
```

```
87
88 ##export merged dataset as csv
89 write_csv(merged_data, "./lowbwt_merged.csv")
90
```

# Next Steps

- We now have one data set that includes data on low and normal birthweight babies and includes data on # of visits!
- Next time, we will explore this data.
  - Descriptive Statistics
  - Visualizing Data
  - Basic Hypothesis Testing

# *Thank you!*

Visit our BERD EDU website for additional resources:

[http://irvinginstitute.columbia.edu/resources/biostat\\_educational\\_initiatives.html](http://irvinginstitute.columbia.edu/resources/biostat_educational_initiatives.html)

Data Wrangling Cheat Sheet on Dropbox !

Acknowledgements: Jeff Goldsmith (Data Science I notes:

<http://p8105.com/>