R and the tidyverse
Winter Institute in Data Science

Ryan T. Moore

2 January 2020

What is R?

R Functions

Data Structures

Core tidyverse Transformation Functions

Other Common Transformation Functions

Helper Functions

R + RTM

- $\approx$ 17th anniversaRy

R + RTM

- $\approx$ 17th anniversaRy
- Author of 3.5 R packages

R + RTM

- $\approx$ 17th anniversaRy
- Author of 3.5 R packages
  - Experimental design (`blockTools`)

R + RTM

- $\approx$ 17th anniversaRy
- Author of 3.5 R packages
  - Experimental design (`blockTools`)
  - Ecological inference (`eiPack`)

R + RTM

- $\approx$ 17th anniversaRy
- Author of 3.5 R packages
  - Experimental design (`blockTools`)
  - Ecological inference (`eiPack`)
  - Twitter conversation analysis (`botscan`)

R + RTM

- $\approx$ 17th anniversaRy
- Author of 3.5 R packages
  - Experimental design (`blockTools`)
  - Ecological inference (`eiPack`)
  - Twitter conversation analysis (`botscan`)
  - Web-scraping, mapping, mail-merging (`muRL`)

R + RTM

- $\approx$ 17th anniversaRy
- Author of 3.5 R packages
  - Experimental design (`blockTools`)
  - Ecological inference (`eiPack`)
  - Twitter conversation analysis (`botscan`)
  - Web-scraping, mapping, mail-merging (`muRL`)
- Research in R, teach with R, teach R, consult with R, run family gift exchange with R, War, . . .

What is R?

What is R?

“R is a language and environment
for statistical computing and graphics”

What is R?

“R is a language and environment
for statistical computing and graphics”

- Software for calculation, computation, data analysis

What is R?

"R is a language and environment
for statistical computing and graphics"

- Software for calculation, computation, data analysis
- Well-developed graphical facilities
- A programming language

Why use R (and Python)?

- R: standard for data analysis, modeling, graphics
- High-quality, powerful, flexible, extensible
- International community (including here!)
- Platform independent (Mac OSX, Windows, Linux/Unix)
- Free
- Reads `.xlsx`, `.dta`, `.csv`, `.txt`, `.json`, . . .
- Interfaces with C, C++, Ruby, Java, Python, Unix, . . .
- Command line (Mac OS Terminal prompt), Windows/Mac/Linux GUIs
- RStudio: excellent IDE (code, plots, etc. 1 window; GitHub)
- Let R teach you R: `swirl`

How does R Work?

How does R Work?

```r
5 + 2
```

How does R Work?

```
5 + 2
```

```
## [1] 7
```

How does R Work?

```r
5 + 2
```

```
## [1] 7
```

```r
sum(5, 2)
```

How does R Work?

```r
5 + 2
```

```
## [1] 7
```

```r
sum(5, 2)
```

```
## [1] 7
```

How does R Work?

```r
5 + 2
```

```
## [1] 7
```

```r
sum(5, 2)
```

```
## [1] 7
```

But not just printing:

How does R Work?

```
5 + 2
```

```
## [1] 7
```

```
sum(5, 2)
```

```
## [1] 7
```

But not just printing:

```
a <- sum(5, 2)
b <- median(1:10)
a + b # (Hi -- Notes after the `#' R ignores)
```

How does R Work?

```
5 + 2
```

```
## [1] 7
```

```
sum(5, 2)
```

```
## [1] 7
```

But not just printing:

```
a <- sum(5, 2)
b <- median(1:10)
a + b # (Hi -- Notes after the `#' R ignores)
```

```
## [1] 12.5
```

How does R Work?

```
5 + 2
```

```
## [1] 7
```

```
sum(5, 2)
```

```
## [1] 7
```

But not just printing:

```
a <- sum(5, 2)
b <- median(1:10)
a + b # (Hi -- Notes after the `#' R ignores)
```

```
## [1] 12.5
```

```
difftime("2020-01-02", "2020-07-04")
```

How does R Work?

```
5 + 2
```

```
## [1] 7
```

```
sum(5, 2)
```

```
## [1] 7
```

But not just printing:

```
a <- sum(5, 2)
b <- median(1:10)
a + b # (Hi -- Notes after the `#' R ignores)
```

```
## [1] 12.5
```

```
difftime("2020-01-02", "2020-07-04")
```

```
## Time difference of -548.9583 days
```

How to Work in R

- Open R/RStudio

How to Work in R

- Open R/RStudio
- Create a `.R` file

How to Work in R

- Open R/RStudio
- Create a `.R` file
- Add code and comments to the `.R` file

How to Work in R

- Open R/RStudio
- Create a `.R` file
- Add code and comments to the `.R` file
- Run them to get output, results, graphics, …

How to Work in R

- Open R/RStudio
- Create a `.R` file
- Add code and comments to the `.R` file
- Run them to get output, results, graphics, . . .

  ⇝ Mac: Cmd-Return to execute a line
  (better than copy-paste)

  ⇝ At `>`, [Up Arrow] recalls previous command

How to Work in R

- Open R/RStudio
- Create a `.R` file
- Add code and comments to the `.R` file
- Run them to get output, results, graphics, . . .

  ⇝ Mac: Cmd-Return to execute a line
  (better than copy-paste)

  ⇝ At `>`, [Up Arrow] recalls previous command
- Save `.R` file

How to Work in R

- Open R/RStudio
- Create a `.R` file
- Add code and comments to the `.R` file
- Run them to get output, results, graphics, ...

  ⇝ Mac: Cmd-Return to execute a line
  (better than copy-paste)

  ⇝ At `>`, [Up Arrow] recalls previous command
- Save `.R` file
- Quit

How to Work in R

- Open R/RStudio
- Create a `.R` file
- Add code and comments to the `.R` file
- Run them to get output, results, graphics, . . .

  ⇝ Mac: Cmd-Return to execute a line
  (better than copy-paste)

  ⇝ At `>`, [Up Arrow] recalls previous command
- Save `.R` file
- Quit (do not save workspace)

How to Work in R

- Open R/RStudio
- Create a `.R` file
- Add code and comments to the `.R` file
- Run them to get output, results, graphics, . . .

  ⇝ Mac: Cmd-Return to execute a line
  (better than copy-paste)

  ⇝ At `>`, [Up Arrow] recalls previous command
- Save `.R` file
- Quit (do not save workspace)

Later, . . .

- Open `.R` file
- Add more code and comments . . .

How do I get help?

Within R:

```r
help(mean)
help.search("median")
```

How do I get help?

Within R:

```
help(mean)
help.search("median")
```

```
example(mean)
```

```
##
## mean> x <- c(0:10, 50)
##
## mean> xm <- mean(x)
##
## mean> c(xm, mean(x, trim = 0.10))
## [1] 8.75 5.50
```

How do I get help?

Outside of R:

- Q & A
  - Stack Overflow (tags `r`, `rstats`)
  - `DATASCIENCE-L@listserv.american.edu` (info here)

How do I get help?

Outside of R:

- Q & A
  - Stack Overflow (tags `r`, `rstats`)
  - `DATASCIENCE-L@listserv.american.edu` (info here)
- Courses and references
  - `rseek.org` (custom Google search)
  - `CRANsearcher` (RStudio add-in for pkgs)
  - Lynda.com video courses through AU Portal
  - Many good books and documents

    ⇝ Cookbook, Intro Statistics (1 | 2), Student Companion, Graphics, Mapping, Programming, Short Ref Card, . . .

R Functions

Functions

```
function(arg1, arg2, ...){
  <the function code here...>
}
```

Functions

```
function(arg1, arg2, ...){
  <the function code here...>
}
```

```
sum(5, 2)
```

```
## [1] 7
```

Functions

```
function(arg1, arg2, ...){
  <the function code here...>
}
```

```
sum(5, 2)
```

```
## [1] 7
```

```
mean(1:4)
```

```
## [1] 2.5
```

Functions

```
## [1] "(Ready, Marc?)"
```

Functions

```
## [1] "(Ready, Marc?)"
```

```r
nchar("greetings")
```

Functions

```
## [1] "(Ready, Marc?)"
```

```r
nchar("greetings")
```

```
## [1] 9
```

Functions

```
## [1] "(Ready, Marc?)"
```

```r
nchar("greetings")
```

```
## [1] 9
```

```r
ls()
```

Functions

```
## [1] "(Ready, Marc?)"
```

```r
nchar("greetings")
```

```
## [1] 9
```

```r
ls()
```

```
## [1] "a"  "b"  "us" "x"  "xm"
```

A Useful Function: `c()`

To concatenate objects into a vector, use `c()`:

A Useful Function: `c()`

To concatenate objects into a vector, use `c()`:

```r
c(1, 3, 8, 20)
```

```
## [1]  1  3  8 20
```

A Useful Function: `c()`

To concatenate objects into a vector, use `c()`:

```
c(1, 3, 8, 20)
```

```
## [1]  1  3  8 20
```

```
c("a", "merican", "u")
```

A Useful Function: `c()`

To concatenate objects into a vector, use `c()`:

```r
c(1, 3, 8, 20)
```

```
## [1]  1  3  8 20
```

```r
c("a", "merican", "u")
```

```
## [1] "a"       "merican" "u"
```

A Useful Function: `c()`

To concatenate objects into a vector, use `c()`:

```
c(1, 3, 8, 20)
```

```
## [1]  1  3  8 20
```

```
c("a", "merican", "u")
```

```
## [1] "a"       "merican" "u"
```

```
c(1, 2, "hello")
```

A Useful Function: c()

To concatenate objects into a vector, use c():

```
c(1, 3, 8, 20)
```

```
## [1]  1  3  8 20
```

```
c("a", "merican", "u")
```

```
## [1] "a"       "merican" "u"
```

```
c(1, 2, "hello")
```

```
## [1] "1"     "2"     "hello"
```

Functions' Arguments

What arguments does a function have?

Functions' Arguments

What arguments does a function have?

```r
help(median)
args(median)
```

Functions’ Arguments

What arguments does a function have?

```r
help(median)
args(median)
```

```
## function (x, na.rm = FALSE, ...)
## NULL
```

Functions' Arguments

```
median(1:3)
```

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

Functions' Arguments

```r
median(1:3)
```

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

```r
x <- c(1, 2, 3, NA)
median(x)
```

Functions' Arguments

```r
median(1:3)
```

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

```r
x <- c(1, 2, 3, NA)
median(x)
```

```
## [1] NA
```

Functions' Arguments

```r
median(1:3)
```

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

```r
x <- c(1, 2, 3, NA)
median(x)
```

```
## [1] NA
```

```r
median(x, na.rm = TRUE)
```

Functions' Arguments

```r
median(1:3)
```

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

```r
x <- c(1, 2, 3, NA)
median(x)
```

```
## [1] NA
```

```r
median(x, na.rm = TRUE)
```

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

Functions' Arguments

You can specify arguments in order or by name:

Functions' Arguments

You can specify arguments in order or by name:

```r
median(x, TRUE)
```

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

Functions' Arguments

You can specify arguments in order or by name:

```r
median(x, TRUE)
```

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

```r
median(na.rm = TRUE, x)
```

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

Functions' Arguments

You can specify arguments in order or by name:

```r
median(x, TRUE)
```

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

```r
median(na.rm = TRUE, x)
```

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

```r
median(TRUE, x)
```

```
## [1] TRUE
```

Some Useful Functions

Managing the workspace:

```
# Get the working directory ("Where am I?"):
getwd()
```

```
## [1] "/Users/rtm/Documents/github/winter-inst/01-intr
```

Some Useful Functions

Managing the workspace:

```r
# Get the working directory ("Where am I?"):
getwd()
```

```
## [1] "/Users/rtm/Documents/github/winter-inst/01-intr
```

```r
# Set the working directory:
setwd("~/Desktop/")
```

Some Useful Functions

Managing the workspace:

```
# Get the working directory ("Where am I?"):
getwd()
```

```
## [1] "/Users/rtm/Documents/github/winter-inst/01-intr
```

```
# Set the working directory:
setwd("~/Desktop/")
```

```
# List objects in working dir:
ls()
```

```
## [1] "a"  "b"  "us" "x"  "xm"
```

```
# Remove `x' from working dir:
rm(x)
# Remove everything from working dir:
rm(list = ls())
```

Some Useful Mathematical Functions

```
5 + 2
```

```
## [1] 7
```

```
5 - 2
```

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

```
5 * 2
```

```
## [1] 10
```

```
5 / 2
```

```
## [1] 2.5
```

Some Useful Mathematical Functions

```r
5 ^ 2
```

```
## [1] 25
```

```r
sqrt(25)
```

```
## [1] 5
```

```r
20 %% 3
```

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

Some Useful Mathematical Functions and Values

```r
pi
```

```
## [1] 3.141593
```

```r
abs(-3)
```

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

```r
exp(1)
```

```
## [1] 2.718282
```

```r
log(exp(2))
```

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

```r
sin(pi / 2)
```

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

Some Useful Mathematical Functions and Values

```r
pi
```

```
## [1] 3.141593
```

```r
abs(-3)
```

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

```r
exp(1)
```

```
## [1] 2.718282
```

```r
log(exp(2))
```

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

```r
sin(pi / 2)
```

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

(See R Short Ref Card . . .)

Logicals

```r
TRUE
```

```
## [1] TRUE
```

```r
FALSE
```

```
## [1] FALSE
```

Logicals

```
TRUE
```

```
## [1] TRUE
```

```
FALSE
```

```
## [1] FALSE
```

```
TRUE == FALSE
```

Logicals

```r
TRUE
```

```
## [1] TRUE
```

```r
FALSE
```

```
## [1] FALSE
```

```r
TRUE == FALSE
```

```
## [1] FALSE
```

Logicals

```r
c(1, 2) == c(1, 3)
```

Logicals

```r
c(1, 2) == c(1, 3)
```

```
## [1]  TRUE FALSE
```

Logicals

```r
c(1, 2) == c(1, 3)
```

```
## [1]  TRUE FALSE
```

```r
c(1, 2) != c(1, 3)
```

Logicals

```r
c(1, 2) == c(1, 3)
```

```
## [1]  TRUE FALSE
```

```r
c(1, 2) != c(1, 3)
```

```
## [1] FALSE  TRUE
```

Logicals

```r
c(1, 2) == c(1, 3)
```

```
## [1]  TRUE FALSE
```

```r
c(1, 2) != c(1, 3)
```

```
## [1] FALSE  TRUE
```

```r
c(1, 2) < c(1, 3)
```

Logicals

```
c(1, 2) == c(1, 3)
```

```
## [1]  TRUE FALSE
```

```
c(1, 2) != c(1, 3)
```

```
## [1] FALSE  TRUE
```

```
c(1, 2) < c(1, 3)
```

```
## [1] FALSE  TRUE
```

Logicals

```
c(1, 2) > c(1, 3)
```

```
## [1] FALSE FALSE
```

```
c(1, 2) <= c(1, 3)
```

```
## [1] TRUE TRUE
```

```
c(1, 2) >= c(1, 3)
```

```
## [1]  TRUE FALSE
```

How to Write a New Function

```r
sumDiff <- function(num1 = 3, num2 = 5){

  sum <- num1 + num2

  diff <- num1 - num2

  return(c(sum, diff))
}
```

How to Write a New Function

```
sumDiff <- function(num1 = 3, num2 = 5){

  sum <- num1 + num2

  diff <- num1 - num2

  return(c(sum, diff))
}
```

Now, cut and paste function into R prompt.

How to Write a New Function

```r
sumDiff <- function(num1 = 3, num2 = 5){

  sum <- num1 + num2

  diff <- num1 - num2

  return(c(sum, diff))
}
```

Now, cut and paste function into R prompt.

(R will tell you if syntax error.)

My New Function

```r
sumDiff()
```

My New Function

```r
sumDiff()
```

```
## [1]  8 -2
```

My New Function

```r
sumDiff()
```

```
## [1]  8 -2
```

```r
sumDiff(3, 5)
```

My New Function

```r
sumDiff()
```

```
## [1]  8 -2
```

```r
sumDiff(3, 5)
```

```
## [1]  8 -2
```

My New Function

```r
sumDiff()
```

```
## [1]  8 -2
```

```r
sumDiff(3, 5)
```

```
## [1]  8 -2
```

```r
sumDiff(num2 = 5, num1 = 3)
```

My New Function

```r
sumDiff()
```

```
## [1]  8 -2
```

```r
sumDiff(3, 5)
```

```
## [1]  8 -2
```

```r
sumDiff(num2 = 5, num1 = 3)
```

```
## [1]  8 -2
```

My New Function

```
sumDiff()
```

```
## [1]  8 -2
```

```
sumDiff(3, 5)
```

```
## [1]  8 -2
```

```
sumDiff(num2 = 5, num1 = 3)
```

```
## [1]  8 -2
```

```
sumDiff(5, 3)
```

My New Function

```r
sumDiff()
```

```
## [1]  8 -2
```

```r
sumDiff(3, 5)
```

```
## [1]  8 -2
```

```r
sumDiff(num2 = 5, num1 = 3)
```

```
## [1]  8 -2
```

```r
sumDiff(5, 3)
```

```
## [1] 8 2
```

My New Function

```
sumDiff(2, 20)
```

My New Function

```r
sumDiff(2, 20)
```

```
## [1]  22 -18
```

My New Function

```r
sumDiff(2, 20)
```

```
## [1]  22 -18
```

```r
sumDiff(1, "yes")
```

My New Function

```
sumDiff(2, 20)
```

```
## [1]  22 -18
```

```
sumDiff(1, "yes")
```

```
## Error in num1 + num2: non-numeric argument
```

Data Structures

Data Types

- Numeric
- Integer
- Complex
- Logical
- Character
- Factor

Data Types

- Numeric
- Integer
- Complex
- Logical
- Character
- Factor

  $\rightsquigarrow$ categorical vars: stored as numeric, but w/ char label

  $\rightsquigarrow$ great for statistical modeling (auto indicators, e.g.)

Data Structures

- Scalar
- Vector
- Matrix

Data Structures

- Scalar
- Vector
- Matrix
- Data frame (like matrix, w/ attributes)

Data Structures

- Scalar
- Vector
- Matrix
- Data frame (like matrix, w/ attributes)
- Tibble (tidyverse dataframe)

Data Structures

- Scalar
- Vector
- Matrix
- Data frame (like matrix, w/ attributes)
- Tibble (tidyverse dataframe)
- List (flexible storage; regression output)

What is this thing?

```r
x <- 1:4
is.vector(x)
```

What is this thing?

```r
x <- 1:4
is.vector(x)
```

```
## [1] TRUE
```

What is this thing?

```
x <- 1:4
is.vector(x)
```

```
## [1] TRUE
```

```
is.numeric(x)
```

What is this thing?

```r
x <- 1:4
is.vector(x)
```

```
## [1] TRUE
```

```r
is.numeric(x)
```

```
## [1] TRUE
```

What is this thing?

```r
x <- 1:4
is.vector(x)
```

```
## [1] TRUE
```

```r
is.numeric(x)
```

```
## [1] TRUE
```

```r
is.character(x)
```

What is this thing?

```r
x <- 1:4
is.vector(x)
```

```
## [1] TRUE
```

```r
is.numeric(x)
```

```
## [1] TRUE
```

```r
is.character(x)
```

```
## [1] FALSE
```

What is this thing?

```r
y <- c("a", "hello")
is.vector(y)
```

What is this thing?

```r
y <- c("a", "hello")
is.vector(y)
```

```
## [1] TRUE
```

What is this thing?

```r
y <- c("a", "hello")
is.vector(y)
```

```
## [1] TRUE
```

```r
is.numeric(y)
```

What is this thing?

```r
y <- c("a", "hello")
is.vector(y)
```

```
## [1] TRUE
```

```r
is.numeric(y)
```

```
## [1] FALSE
```

What is this thing?

```r
y <- c("a", "hello")
is.vector(y)
```

```
## [1] TRUE
```

```r
is.numeric(y)
```

```
## [1] FALSE
```

```r
is.character(y)
```

What is this thing?

```
y <- c("a", "hello")
is.vector(y)
```

```
## [1] TRUE
```

```
is.numeric(y)
```

```
## [1] FALSE
```

```
is.character(y)
```

```
## [1] TRUE
```

What is this thing?

```r
z <- c(1, 2, 3, NA)
isNAz <- is.na(z)
```

What is this thing?

```r
z <- c(1, 2, 3, NA)
isNAz <- is.na(z)
```

```r
isNAz
```

```
## [1] FALSE FALSE FALSE  TRUE
```

What is this thing?

```r
z <- c(1, 2, 3, NA)
isNAz <- is.na(z)
```

```r
isNAz
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```r
sum(isNAz)
```

What is this thing?

```
z <- c(1, 2, 3, NA)
isNAz <- is.na(z)
```

```
isNAz
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
sum(isNAz)
```

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

What is this thing?

```
z <- c(1, 2, 3, NA)
isNAz <- is.na(z)
```

```
isNAz
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
sum(isNAz)
```

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

```
mean(isNAz)
```

What is this thing?

```
z <- c(1, 2, 3, NA)
isNAz <- is.na(z)
```

```
isNAz
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
sum(isNAz)
```

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

```
mean(isNAz)
```

```
## [1] 0.25
```

What is this thing?

```
z <- c(1, 2, 3, NA)
isNAz <- is.na(z)
```

```
isNAz
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
sum(isNAz)
```

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

```
mean(isNAz)
```

```
## [1] 0.25
```

(“coercion”)

Data from Where?

- From the keyboard
- From within a package
- From `.RData` file
- From a local `.txt`, `.csv`, `.dta`, `.xlsx`, etc. file
- From a remote file on the web
- From remote HTML

Data: Extracting and Assigning Vector Elements

```
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
x[2]
```

Data: Extracting and Assigning Vector Elements

```r
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
x[2]
```

```
## [1] 20
```

Data: Extracting and Assigning Vector Elements

```
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
x[2]
```

```
## [1] 20
```

```
y[3:5]
```

Data: Extracting and Assigning Vector Elements

```r
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
x[2]
```

```
## [1] 20
```

```r
y[3:5]
```

```
## [1] 30 70 10
```

Data: Extracting and Assigning Vector Elements

```
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
x[2]
```

```
## [1] 20
```

```
y[3:5]
```

```
## [1] 30 70 10
```

```
x[c(1, 5)]
```

Data: Extracting and Assigning Vector Elements

```
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
x[2]
```

```
## [1] 20
```

```
y[3:5]
```

```
## [1] 30 70 10
```

```
x[c(1, 5)]
```

```
## [1] 10 25
```

Data: Extracting and Assigning Vector Elements

```
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
x[2]
```

```
## [1] 20
```

```
y[3:5]
```

```
## [1] 30 70 10
```

```
x[c(1, 5)]
```

```
## [1] 10 25
```

```
x[3] <- 100
x
```

Data: Extracting and Assigning Vector Elements

```r
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
x[2]
```

```
## [1] 20
```

```r
y[3:5]
```

```
## [1] 30 70 10
```

```r
x[c(1, 5)]
```

```
## [1] 10 25
```

```r
x[3] <- 100
x
```

```
## [1]  10  20 100  40  25
```

Data: Extracting and Assigning Matrix Elements

```r
m <- matrix(c(20, 20, 30, 10, 20, 30), 3, 2)
m
```

```
##      [,1] [,2]
## [1,]   20   10
## [2,]   20   20
## [3,]   30   30
```

Data: Extracting and Assigning Matrix Elements

```
m <- matrix(c(20, 20, 30, 10, 20, 30), 3, 2)
m
```

```
##      [,1] [,2]
## [1,]   20   10
## [2,]   20   20
## [3,]   30   30
```

```
m[1, 2]
```

Data: Extracting and Assigning Matrix Elements

```r
m <- matrix(c(20, 20, 30, 10, 20, 30), 3, 2)
m
```

```
##      [,1] [,2]
## [1,]   20   10
## [2,]   20   20
## [3,]   30   30
```

```r
m[1, 2]
```

```
## [1] 10
```

Data: Extracting and Assigning Matrix Elements

```
m <- matrix(c(20, 20, 30, 10, 20, 30), 3, 2)
m
```

```
##      [,1] [,2]
## [1,]   20   10
## [2,]   20   20
## [3,]   30   30
```

```
m[1, 2]
```

```
## [1] 10
```

```
m[2, 2] <- NA
m
```

Data: Extracting and Assigning Matrix Elements

```
m <- matrix(c(20, 20, 30, 10, 20, 30), 3, 2)
m
```

```
##      [,1] [,2]
## [1,]   20   10
## [2,]   20   20
## [3,]   30   30
```

```
m[1, 2]
```

```
## [1] 10
```

```
m[2, 2] <- NA
m
```

```
##      [,1] [,2]
## [1,]   20   10
## [2,]   20   NA
## [3,]   30   30
```

Data from Keyboard, into a Data Frame

```r
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
df <- data.frame(age = y, score = x)
```

Data from Keyboard, into a Data Frame

```
y <- c(20, 20, 30, 70, 10)
x <- c(10, 20, 30, 40, 25)
df <- data.frame(age = y, score = x)
```

```
df
```

```
##   age score
## 1  20    10
## 2  20    20
## 3  30    30
## 4  70    40
## 5  10    25
```

Data from Keyboard, into a Data Frame

```
df$age
```

```
## [1] 20 20 30 70 10
```

Data from Keyboard, into a Data Frame

```
df$age
```

```
## [1] 20 20 30 70 10
```

```
rownames(df)
```

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

Data from Keyboard, into a Data Frame

```r
df$age
```

```
## [1] 20 20 30 70 10
```

```r
rownames(df)
```

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

```r
colnames(df)
```

```
## [1] "age"   "score"
```

Data: From Keyboard, into a List

```r
my_list <- list(x = 1:3, y = letters[1:5],
                final = matrix(1:4, 2, 2))
```

Data: From Keyboard, into a List

```r
my_list <- list(x = 1:3, y = letters[1:5],
                final = matrix(1:4, 2, 2))
```

```r
my_list
```

```
## $x
## [1] 1 2 3
##
## $y
## [1] "a" "b" "c" "d" "e"
##
## $final
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

Data: From Keyboard, into a List

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

Data: From Keyboard, into a List

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

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

Data: From Keyboard, into a List

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

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

```
my_list[["final"]]
```

Data: From Keyboard, into a List

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

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

```
my_list[["final"]]
```

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

Data: Lists

```
my_list$x
```

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

```
my_list$y
```

```
## [1] "a" "b" "c" "d" "e"
```

Data: Lists

```r
my_list$x
```

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

```r
my_list$y
```

```
## [1] "a" "b" "c" "d" "e"
```

A data frame is a list.

Data from a Package

```r
library(car)
data(Chile)
```

Data from a Package

```
library(car)
data(Chile)
```

```
head(Chile)
```

```
##   region population sex age education income statusq
## 1      N     175000   M  65         P  35000   1.008
## 2      N     175000   M  29        PS   7500  -1.296
## 3      N     175000   F  38         P  15000   1.230
## 4      N     175000   F  49         P  35000  -1.031
## 5      N     175000   F  23         S  35000  -1.104
## 6      N     175000   F  28         P   7500  -1.046
```

Core tidyverse Transformation Functions

What is a Package?

An R package is an extension of R that includes

- a set of functions for users
- datasets
- demonstration code
- “background” code (in R or a compiled language)
- documentation
- metadata (authors, license, e.g.)

How do I get package `thispackage`?

How do I get package `thispackage`?

```
install.packages("thispackage")
```

How do I get package `thispackage`?

```r
install.packages("thispackage")
```

Then,

```r
library(thispackage)
```

What is the tidyverse?

Coherent, consistent *set* of R packages for data import, manipulation, visualization, etc.

What is the tidyverse?

Coherent, consistent *set* of R packages for data import, manipulation, visualization, etc.

How do I get the tidyverse?

```
## [1] "(Ready, Katherine?)"
```

What is the tidyverse?

Coherent, consistent *set* of R packages for data import, manipulation, visualization, etc.

How do I get the tidyverse?

```
## [1] "(Ready, Katherine?)"
```

What is the tidyverse?

Coherent, consistent *set* of R packages for data import, manipulation, visualization, etc.

How do I get the tidyverse?

```
## [1] "(Ready, Katherine?)"
```

```r
install.packages("tidyverse")
```

What is the tidyverse?

Coherent, consistent *set* of R packages for data import, manipulation, visualization, etc.

How do I get the tidyverse?

```
## [1] "(Ready, Katherine?)"
```

```r
install.packages("tidyverse")
```

```r
library(tidyverse)
```

The Core Transformation Functions

- `filter()`
- `arrange()`
- `group_by()` (and `ungroup()`)
- `select()` (and `rename()`)
- `mutate()`
- `transmute()`
- `summarise()`

Core Transformation Functions: Social Pressure Experiment

```
# URL is "https://raw.githubusercontent.com/kosukeimai/
#         qss/master/CAUSALITY/social.csv"
url <- "http://j.mp/2Et71U0"
social <- read_csv(url)
dim(social)
```

```
## [1] 305866      6
```

Core Transformation Functions: Social Pressure Experiment

```r
# URL is "https://raw.githubusercontent.com/kosukeimai/
#           qss/master/CAUSALITY/social.csv"
url <- "http://j.mp/2Et71U0"
social <- read_csv(url)
dim(social)
```

```
## [1] 305866      6
```

```r
head(social, 3)
```

```
## # A tibble: 3 x 6
##   sex    yearofbirth primary2004 messages   primary2006 hhsize
##   <chr>        <dbl>       <dbl> <chr>            <dbl>  <dbl>
## 1 male          1941           0 Civic Duty           0      2
## 2 female        1947           0 Civic Duty           0      2
## 3 male          1951           0 Hawthorne            1      3
```

`filter()`

Keep only voters in households that might have interference:

filter()

Keep only voters in households that might have interference:

```
table(social$hhsize)
```

```
##
##      1      2      3      4      5      6      7
##  42524 190294  51057  18596   2955    390     42
```

filter()

Keep only voters in households that might have interference:

```
table(social$hhsize)
```

```
##
##       1      2      3      4      5      6      7
##   42524 190294  51057  18596   2955    390     42
```

```
df_interf <- filter(social, hhsize > 1)
dim(df_interf)
```

```
## [1] 263342      6
```

filter()

Keep only non-voters who might be subject to interference:

filter()

Keep only non-voters who might be subject to interference:

```r
filter(social, (hhsize > 1) & (primary 2004 == 0))
```

filter()

Keep only non-voters who might be subject to interference:

```
filter(social, (hhsize > 1) & (primary 2004 == 0))
```

```
## Error: <text>:1:40: unexpected numeric constant
## 1: filter(social, (hhsize > 1) & (primary 2004
##                                             ^
```

filter()

Keep only non-voters who might be subject to interference:

```
filter(social, (hhsize > 1) & (primary 2004 == 0))
```

```
## Error: <text>:1:40: unexpected numeric constant
## 1: filter(social, (hhsize > 1) & (primary 2004
##                                              ^
```

```
filter(social, (hhsize > 1) & (primary2004 == 0))
```

```
## # A tibble: 161,275 x 6
##    sex    yearofbirth primary2004 messages   primary2006 hhsi
##    <chr>        <dbl>       <dbl> <chr>            <dbl>  <db
##  1 male          1941           0 Civic Duty           0
##  2 female        1947           0 Civic Duty           0
##  3 male          1951           0 Hawthorne            1
##  4 female        1950           0 Hawthorne            1
##  5 female        1982           0 Hawthorne            1
##  6 male          1981           0 Control              0
##  7 female        1959           0 Control              1
##  8 male          1956           0 Control              1
```

arrange()

Sort by birth year, then household size

arrange()

Sort by birth year, then household size

```r
arrange(social, yearofbirth, hhsize)
```

```
## # A tibble: 305,866 x 6
##    sex    yearofbirth primary2004 messages  primary2006 hhsize
##    <chr>        <dbl>       <dbl> <chr>           <dbl>  <dbl>
##  1 female        1900           0 Control             0      1
##  2 female        1900           0 Control             0      2
##  3 male          1900           1 Control             0      2
##  4 male          1900           1 Control             1      2
##  5 female        1900           0 Hawthorne           0      2
##  6 female        1900           1 Control             1      3
##  7 female        1902           1 Control             0      1
##  8 female        1902           1 Control             0      3
##  9 male          1903           1 Control             0      1
## 10 female        1904           0 Control             0      1
## # ... with 305,856 more rows
```

```r
# social %>% arrange(yearofbirth, hhsize)
```

mutate()

Create new variable (under_30), TRUE/FALSE

mutate()

Create new variable (under_30), TRUE/FALSE

```r
social %>% mutate(under_30 = yearofbirth > 1976)
```

```
## # A tibble: 305,866 x 7
##    sex    yearofbirth primary2004 messages   primary2006 hhsize under_30
##    <chr>        <dbl>       <dbl> <chr>            <dbl>  <dbl> <lgl>
##  1 male          1941           0 Civic Duty           0      2 FALSE
##  2 female        1947           0 Civic Duty           0      2 FALSE
##  3 male          1951           0 Hawthorne            1      3 FALSE
##  4 female        1950           0 Hawthorne            1      3 FALSE
##  5 female        1982           0 Hawthorne            1      3 TRUE
##  6 male          1981           0 Control              0      3 TRUE
##  7 female        1959           0 Control              1      3 FALSE
##  8 male          1956           0 Control              1      3 FALSE
##  9 female        1968           0 Control              0      2 FALSE
## 10 male          1967           0 Control              0      2 FALSE
## # ... with 305,856 more rows
```

mutate()

Create new variable (under_30), TRUE/FALSE

```
social %>% mutate(under_30 = yearofbirth > 1976)
```

```
## # A tibble: 305,866 x 7
##    sex    yearofbirth primary2004 messages    primary2006 hhsize under_30
##    <chr>        <dbl>       <dbl> <chr>             <dbl>  <dbl> <lgl>
##  1 male          1941           0 Civic Duty            0      2 FALSE
##  2 female        1947           0 Civic Duty            0      2 FALSE
##  3 male          1951           0 Hawthorne             1      3 FALSE
##  4 female        1950           0 Hawthorne             1      3 FALSE
##  5 female        1982           0 Hawthorne             1      3 TRUE
##  6 male          1981           0 Control               0      3 TRUE
##  7 female        1959           0 Control               1      3 FALSE
##  8 male          1956           0 Control               1      3 FALSE
##  9 female        1968           0 Control               0      2 FALSE
## 10 male          1967           0 Control               0      2 FALSE
## # ... with 305,856 more rows
```

(There is also recode().)

mutate_all(), mutate_at(), mutate_if()

```
soc_numeric <- select(social, -sex, -messages)
```

mutate_all(), mutate_at(), mutate_if()

```
soc_numeric <- select(social, -sex, -messages)
```

```
# Halve every column's values:
divide_by_two <- function(x){x / 2}
mutate_all(soc_numeric, divide_by_two)
```

```
## # A tibble: 305,866 x 4
##    yearofbirth primary2004 primary2006 hhsize
##          <dbl>       <dbl>       <dbl>  <dbl>
##  1        970.           0         0      1  
##  2        974.           0         0      1  
##  3        976.           0         0.5    1.5
##  4        975            0         0.5    1.5
##  5        991            0         0.5    1.5
##  6        990.           0         0      1.5
##  7        980.           0         0.5    1.5
##  8        978            0         0.5    1.5
##  9        984            0         0      1  
## 10        984.           0         0      1  
```

mutate_all(), mutate_at(), mutate_if()

```r
# Double values of columns:
mult_by_two <- function(x){x * 2}
mutate_at(soc_numeric, c(2, 3), mult_by_two)
```

```
## # A tibble: 305,866 x 4
##    yearofbirth primary2004 primary2006 hhsize
##          <dbl>       <dbl>       <dbl>  <dbl>
##  1        1941           0           0      2
##  2        1947           0           0      2
##  3        1951           0           2      3
##  4        1950           0           2      3
##  5        1982           0           2      3
##  6        1981           0           0      3
##  7        1959           0           2      3
##  8        1956           0           2      3
##  9        1968           0           0      2
## 10        1967           0           0      2
## # ... with 305,856 more rows
```

mutate_all(), mutate_at(), mutate_if()

What does this do?

```r
mutate_at(soc_numeric, vars(matches("primary")),
          mult_by_two)
```

mutate_all(), mutate_at(), mutate_if()

What does this do?

```r
mutate_at(soc_numeric, vars(matches("primary")),
          mult_by_two)
```

```
## # A tibble: 305,866 x 4
##    yearofbirth primary2004 primary2006 hhsize
##          <dbl>       <dbl>       <dbl>  <dbl>
##  1        1941           0           0      2
##  2        1947           0           0      2
##  3        1951           0           2      3
##  4        1950           0           2      3
##  5        1982           0           2      3
##  6        1981           0           0      3
##  7        1959           0           2      3
##  8        1956           0           2      3
##  9        1968           0           0      2
## 10        1967           0           0      2
## # ... with 305,856 more rows
```

mutate_all(), mutate_at(), mutate_if()

What does this do?

```
mutate_if(social, is.numeric, mean)
```

mutate_all(), mutate_at(), mutate_if()

What does this do?

```
mutate_if(social, is.numeric, mean)
```

```
## # A tibble: 305,866 x 6
##    sex    yearofbirth primary2004 messages   primary2006
##    <chr>        <dbl>       <dbl> <chr>            <dbl>
##  1 male         1956.       0.401 Civic Duty       0.312
##  2 female       1956.       0.401 Civic Duty       0.312
##  3 male         1956.       0.401 Hawthorne        0.312
##  4 female       1956.       0.401 Hawthorne        0.312
##  5 female       1956.       0.401 Hawthorne        0.312
##  6 male         1956.       0.401 Control          0.312
##  7 female       1956.       0.401 Control          0.312
##  8 male         1956.       0.401 Control          0.312
##  9 female       1956.       0.401 Control          0.312
## 10 male         1956.       0.401 Control          0.312
## # ... with 305,856 more rows
```

`mutate_all()`, `mutate_at()`, `mutate_if()`

Warning: `mutate_all()`, `_at()`, `_if()` **overwrite** columns that are processed.

`mutate_all()`, `mutate_at()`, `mutate_if()`

Warning: `mutate_all()`, `_at()`, `_if()` **overwrite** columns that are processed.

Do **not** append new columns to the end.

mutate_all(), mutate_at(), mutate_if()

Warning: mutate_all(), _at(), _if() **overwrite** columns that are processed.

Do **not** append new columns to the end.

Useful for recoding, if want values of a function:

```
is_CD <- function(x){ x == "Civic Duty"}
mutate_at(social, vars(matches("messages")), is_CD)
```

```
## # A tibble: 305,866 x 6
##    sex    yearofbirth primary2004 messages primary2006 h
##    <chr>        <dbl>       <dbl> <lgl>          <dbl>
##  1 male          1941           0 TRUE               0
##  2 female        1947           0 TRUE               0
##  3 male          1951           0 FALSE              1
##  4 female        1950           0 FALSE              1
##  5 female        1982           0 FALSE              1
##  6 male          1981           0 FALSE              0
##  7 female        1959           0 FALSE              1
```

transmute() for new variables, summaries

```
transmute(social, age = 2006 - yearofbirth)
```

transmute() for new variables, summaries

```r
transmute(social, age = 2006 - yearofbirth)
```

```
## # A tibble: 305,866 x 1
##        age
##      <dbl>
##  1    65
##  2    59
##  3    55
##  4    56
##  5    24
##  6    25
##  7    47
##  8    50
##  9    38
## 10    39
## # ... with 305,856 more rows
```

transmute() for new vars, summaries as new vars

```
social_msg_grps <- group_by(social, messages)
```

transmute() for new vars, summaries as new vars

```
social_msg_grps <- group_by(social, messages)
```

```
transmute(social_msg_grps,
          avg_age = mean(2006 - yearofbirth))
```

```
## # A tibble: 305,866 x 2
## # Groups:   messages [4]
##    messages   avg_age
##    <chr>        <dbl>
##  1 Civic Duty    49.7
##  2 Civic Duty    49.7
##  3 Hawthorne     49.7
##  4 Hawthorne     49.7
##  5 Hawthorne     49.7
##  6 Control       49.8
##  7 Control       49.8
##  8 Control       49.8
##  9 Control       49.8
## 10 Control       49.8
```

What if I wanted just mean age per message?

What if I wanted just mean age per message?

```r
summarise(social_msg_grps,
          avg_age = mean(2006 - yearofbirth))
```

```
## # A tibble: 4 x 2
##   messages   avg_age
##   <chr>        <dbl>
## 1 Civic Duty    49.7
## 2 Control       49.8
## 3 Hawthorne     49.7
## 4 Neighbors     49.9
```

What if I wanted just mean age per message?

```
summarise(social_msg_grps,
          avg_age = mean(2006 - yearofbirth))
```

```
## # A tibble: 4 x 2
##   messages   avg_age
##   <chr>        <dbl>
## 1 Civic Duty    49.7
## 2 Control       49.8
## 3 Hawthorne     49.7
## 4 Neighbors     49.9
```

What information does this provide about the experiment?

select()

```r
select(social, yearofbirth, messages, primary2006) # or
social %>% select(yearofbirth, messages, primary2006)
```

select()

```r
select(social, yearofbirth, messages, primary2006) # or
social %>% select(yearofbirth, messages, primary2006)
```

```
## # A tibble: 305,866 x 3
##    yearofbirth messages   primary2006
##          <dbl> <chr>            <dbl>
##  1        1941 Civic Duty           0
##  2        1947 Civic Duty           0
##  3        1951 Hawthorne            1
##  4        1950 Hawthorne            1
##  5        1982 Hawthorne            1
##  6        1981 Control              0
##  7        1959 Control              1
##  8        1956 Control              1
##  9        1968 Control              0
## 10        1967 Control              0
## # ... with 305,856 more rows
```

Other Common Transformation Functions

Other Common Transformation Functions: slice()

```r
slice(social, 1000:1004)
```

```
## # A tibble: 5 x 6
##   sex    yearofbirth primary2004 messages  primary2006 h
##   <chr>        <dbl>       <dbl> <chr>           <dbl>
## 1 male          1955           1 Neighbors           1
## 2 female        1952           0 Control             1
## 3 male          1947           1 Control             1
## 4 female        1985           0 Hawthorne           0
## 5 male          1956           0 Hawthorne           0
```

Other Common Transformation Functions: `slice()`

```r
slice(social, n())
```

```
## # A tibble: 1 x 6
##   sex    yearofbirth primary2004 messages primary2006 hh
##   <chr>        <dbl>       <dbl> <chr>          <dbl> <
## 1 female        1949           1 Control            1
```

Other Common Transformation Functions: sample_n(), sample_frac()

```r
sample_n(social, 4)
```

```
## # A tibble: 4 x 6
##   sex    yearofbirth primary2004 messages  primary2006
##   <chr>        <dbl>       <dbl> <chr>           <dbl>
## 1 female        1962           0 Neighbors           1
## 2 male          1954           0 Control             0
## 3 female        1957           1 Control             0
## 4 female        1966           1 Control             1
```

Other Common Transformation Functions: `sample_n()`, `sample_frac()`

```r
sample_frac(social, 0.00001)
```

```
## # A tibble: 3 x 6
##   sex    yearofbirth primary2004 messages   primary2006
##   <chr>        <dbl>       <dbl> <chr>            <dbl>
## 1 female        1964           0 Civic Duty           0
## 2 female        1932           0 Control              1
## 3 male          1967           1 Neighbors            1
```

Other Common Transformation Functions: `distinct()`

```r
social_distinct <- distinct(social)
dim(social_distinct)
```

```
## [1] 9235    6
```

Other Common Transformation Functions: distinct()

```r
social_distinct <- distinct(social)
dim(social_distinct)
```

```
## [1] 9235    6
```

$(100 \text{ yrs}) \cdot (4 \text{ msgs}) \cdot (4 \text{ votes}) \cdot (2 \text{ sex}) \cdot (3 \text{ HHsize}) = 9600$

Common Structure

```
verb(df, <conditions or calculations>)
```

Common Structure

```
verb(df, <conditions or calculations>)
```

Value: a dataframe

Common Structure

This structure:

$$\text{dataframe in} \rightsquigarrow \text{dataframe out}$$

enables the pipe: `%>%`

The pipe inserts the previous result as the first argument of the subsequent function.

The pipe inserts the previous result as the first argument of the subsequent function.

```
x %>% f(y)
```

is the same as

```
f(x, y)
```

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`
- We want to apply `f()` to `x`, then apply `g()` to the output, then `h()` to the output of `g()`, . . .

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`
- We want to apply `f()` to `x`, then apply `g()` to the output, then `h()` to the output of `g()`, ...
- `f(x)`

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`
- We want to apply `f()` to `x`, then apply `g()` to the output, then `h()` to the output of `g()`, ...
- `f(x)`
- `g(f(x))`

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`
- We want to apply `f()` to `x`, then apply `g()` to the output, then `h()` to the output of `g()`, . . .
- `f(x)`
- `g(f(x))`
- `h(g(f(x)))`

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`
- We want to apply `f()` to `x`, then apply `g()` to the output, then `h()` to the output of `g()`, . . .
- `f(x)`
- `g(f(x))`
- `h(g(f(x)))`

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`
- We want to apply `f()` to `x`, then apply `g()` to the output, then `h()` to the output of `g()`, . . .
- `f(x)`
- `g(f(x))`
- `h(g(f(x)))`

Or, with more assignments,

- `y <- f(x)`

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`
- We want to apply `f()` to `x`, then apply `g()` to the output, then `h()` to the output of `g()`, …
- `f(x)`
- `g(f(x))`
- `h(g(f(x)))`

Or, with more assignments,

- `y <- f(x)`
- `z <- g(y)`

The Pipe

- Suppose we have functions `f()`, `g()`, and `h()`
- We want to apply `f()` to `x`, then apply `g()` to the output, then `h()` to the output of `g()`, . . .
- `f(x)`
- `g(f(x))`
- `h(g(f(x)))`

Or, with more assignments,

- `y <- f(x)`
- `z <- g(y)`
- `h(z)`

The Pipe

The pipe (`%>%`) allows us to write

```
x %>% f() %>% g() %>% h()
```

The Pipe

The pipe (`%>%`) allows us to write

```
x %>% f() %>% g() %>% h()
```

Likely better,

```
x %>%
  f() %>%
  g() %>%
  h()
```

The Pipe

The pipe (`%>%`) allows us to write

`x %>% f() %>% g() %>% h()`

Likely better,

```
x %>%
  f() %>%
  g() %>%
  h()
```

To be able to reorder *depends* on functions all

- taking same first input
- producing output of same type as input

The Pipe

The `%>%` is like $\circ$ for function composition, but still reads in order.

The Pipe

The `%>%` is like $\circ$ for function composition, but still reads in order.

(Unlike $h(g(f(x)))$ or $(h \circ g \circ f)(x)$)

The Pipe

The `%>%` is like $\circ$ for function composition, but still reads in order.

(Unlike $h(g(f(x)))$ or $(h \circ g \circ f)(x)$)

Read “then”.

The Pipe

Suppose each function takes more than 1 argument:

The Pipe

Suppose each function takes more than 1 argument:

```
h(g(f(x, arg1 = value_here), arg2 = another_val),
  arg3 = 5, arg4 = TRUE)
```

The Pipe

Suppose each function takes more than 1 argument:

```
h(g(f(x, arg1 = value_here), arg2 = another_val),
  arg3 = 5, arg4 = TRUE)
```

Messy. Which function is `arg2`? `arg3`?

The Pipe

Suppose each function takes more than 1 argument:

```r
h(g(f(x, arg1 = value_here), arg2 = another_val),
  arg3 = 5, arg4 = TRUE)
```

Messy. Which function is `arg2`? `arg3`?

```r
x %>%
  f(arg1 = value_here) %>%
  g(arg2 = another_val) %>%
  h(arg3 = 5, arg4 = TRUE)
```

The Pipe

Suppose each function takes more than 1 argument:

```
h(g(f(x, arg1 = value_here), arg2 = another_val),
  arg3 = 5, arg4 = TRUE)
```

Messy. Which function is `arg2`? `arg3`?

```
x %>%
  f(arg1 = value_here) %>%
  g(arg2 = another_val) %>%
  h(arg3 = 5, arg4 = TRUE)
```

Better.

The Pipe

Fun note: The pipe is defined in package `magrittr`

The Pipe

Fun note: The pipe is defined in package `magrittr`

The motif is played **all** the way out: http://j.mp/2Eu679T

The Pipe

Fun note: The pipe is defined in package `magrittr`

The motif is played **all** the way out: http://j.mp/2Eu679T

(For similar missing data example, see Amelia.)

Helper Functions

Helpers for `select()`-ing Variables

- `contains()`
- `starts_with()`, `ends_with()`
- `matches()`
- `num_range()`
- `one_of()`
- `everything()`

Helpers for select()-ing Variables

```
social %>% select(contains("s")) %>% slice(1:2) # literal
```

```
## # A tibble: 2 x 3
##   sex    messages    hhsize
##   <chr>  <chr>        <dbl>
## 1 male   Civic Duty       2
## 2 female Civic Duty       2
```

Helpers for select()-ing Variables

```r
social %>% select(contains("s")) %>% slice(1:2) # literal
```

```
## # A tibble: 2 x 3
##   sex    messages    hhsize
##   <chr>  <chr>        <dbl>
## 1 male   Civic Duty       2
## 2 female Civic Duty       2
```

```r
social %>% select(starts_with("primary")) %>% slice(1:2)
```

```
## # A tibble: 2 x 2
##   primary2004 primary2006
##         <dbl>       <dbl>
## 1           0           0
## 2           0           0
```

Helpers for `select()`-ing Variables

```r
social %>% select(ends_with("size")) %>% slice(1:2)
```

```
## # A tibble: 2 x 1
##   hhsize
##    <dbl>
## 1      2
## 2      2
```

Helpers for select()-ing Variables

```
social %>% select(ends_with("size")) %>% slice(1:2)
```

```
## # A tibble: 2 x 1
##   hhsize
##    <dbl>
## 1      2
## 2      2
```

```
social %>% select(matches(".00.")) %>% slice(1:2) # regex
```

```
## # A tibble: 2 x 2
##   primary2004 primary2006
##         <dbl>       <dbl>
## 1           0           0
## 2           0           0
```

Helpers for select()-ing Variables

```
social %>% select(num_range("primary", 2000:2008)) %>% slic
```

```
## # A tibble: 2 x 2
##   primary2004 primary2006
##         <dbl>       <dbl>
## 1           0           0
## 2           0           0
```

Helpers for select()-ing Variables

```
social %>% select(num_range("primary", 2000:2008)) %>% slic
```

```
## # A tibble: 2 x 2
##   primary2004 primary2006
##         <dbl>       <dbl>
## 1           0           0
## 2           0           0
```

But

```
social %>% select(num_range("primary", 2000:2005)) %>% slic
```

```
## # A tibble: 2 x 1
##   primary2004
##         <dbl>
## 1           0
## 2           0
```

Helpers for `select()`-ing Variables

```
social %>% select(one_of(c("sex", "hhsize"))) %>% slice(1:2
```

```
## # A tibble: 2 x 2
##   sex    hhsize
##   <chr>   <dbl>
## 1 male        2
## 2 female      2
```

Helpers for select()-ing Variables

```
social %>% select(primary2006, messages, everything()) %>%
  slice(1:9)
```

```
## # A tibble: 9 x 6
##   primary2006 messages   sex    yearofbirth primary2004
##         <dbl> <chr>      <chr>        <dbl>       <dbl>
## 1           0 Civic Duty male          1941           0
## 2           0 Civic Duty female        1947           0
## 3           1 Hawthorne  male          1951           0
## 4           1 Hawthorne  female        1950           0
## 5           1 Hawthorne  female        1982           0
## 6           0 Control    male          1981           0
## 7           1 Control    female        1959           0
## 8           1 Control    male          1956           0
## 9           0 Control    female        1968           0
```

Helpers for `select()`-ing Variables

```
social %>% select(primary2006, messages, everything()) %>%
  slice(1:9)
```

```
## # A tibble: 9 x 6
##   primary2006 messages   sex    yearofbirth primary2004
##         <dbl> <chr>      <chr>        <dbl>       <dbl>
## 1           0 Civic Duty male          1941           0
## 2           0 Civic Duty female        1947           0
## 3           1 Hawthorne  male          1951           0
## 4           1 Hawthorne  female        1950           0
## 5           1 Hawthorne  female        1982           0
## 6           0 Control    male          1981           0
## 7           1 Control    female        1959           0
## 8           1 Control    male          1956           0
## 9           0 Control    female        1968           0
```

(Use `select()` as the `arrange()` of columns.)

Helpers for `mutate()`

- Offsets
- Cumulative aggregates
- Ranking functions

Viewing the Data

- `df`
- `View(df)`
- `as.data.frame(tbl)`
- `tbl %>% as.data.frame()`

Recently, at The Lab... preprocessing

```r
df_outcomes <- df_outcomes %>%
  rename(ic_case_id = "IC# (Household)",
         pdc_number = "TANF PDC Case #",
         pdc_status = "PDC Current Status",
         renewal_date = "Renewal Date",
         )
```

Recently, at The Lab...

```r
df_arrest <- df_arrest %>%
  rename(age = Age,
         race = `Defendant Race`)

df_stop <- df_stop %>%
  rename(age = `Subject Age`,
         race = `Subject_Race`)
```

Recently, at The Lab…

```
df_stop$sex <- recode(df_stop$sex,
                      Female = "F", Male = "M")

df_stop$sex <- na_if(df_stop$sex, "Unknown")
```

Recently, at The Lab… deduplication

```
final_baseline_data <- final_baseline_data %>%
  filter(!((ic_case_id == 1234) & (pdc_number == 2))) %>%
  filter(!((ic_case_id == 5678) &
             (address == "1600 Pennsylvania Ave NW"))) %>%
  filter(!((ic_case_id == 6961) & (pdc_number == 9))) %>%
  filter(!((ic_case_id == 2087) & (pdc_number == 7)))
```

Recently, at The Lab. . .

```r
df_only_dup_months <- df_only_duplicated %>%
  group_by(ic_case_id) %>%
  summarise(month_count = n_distinct(recert_month)) %>%
  filter(month_count > 1) %>%
  select(ic_case_id)
```

Comparing Base R vs. the Tidyverse

Which do you prefer?

```r
df[1, 3]
```

vs.

```r
df %>%
  slice(1) %>%
  select(3)
```

Comparing Base R vs. the Tidyverse

Which do you prefer?

```r
select(df, x1, x2)
```

vs.

```r
df %>% select(x1, x2)
```

vs.

```r
df[, c("x1", "x2")]
```

Find a Friend

One	Two
JessicaG	Zeinabou
Edward	Jocelyn
Xiaofeng	Olan
Katherine	Milika
Kathleen	JessicaK
Carine	Erin
Tanesia	Hubbert
Kelly	Lucas
Mark	Hannah
Cameron	Bryce
AndrewE	AndrewZ
Robin	Lauren
Marc	Ethan

The Core Transformation Functions[1] Quiz

Suppose we have dataframe `df` with 100 rows, continuous variable `x` and categorical `y`.

Hand-write code to

1. sort `df` by the values of `x`? (largest first)
2. create a new variable `x_sq` – the square of each row's `x` value – and attach it as a column of `df`?
3. create `df2`, which has only the rows of `df` where `x > 5`?
4. calculate the median value of `x` within categories of `y`?

[1] `filter()`, `arrange()`, `group_by()`, `ungroup()`, `select()`, `rename()`, `mutate()`, `transmute()`, `summarise()`