1. The R command line; using variables
The Plan:

• Basic R syntax and approach
  – Variables, datatypes, data structures
  – Functional, vector-based language
  – Descriptive statistics and plotting
  – Pointers to useful references

• Introduce development and prototyping environments
  – RStudio
  – Jupyter Notebooks for R
Introductions: R

- Functional language
  - Functions are first-class objects
- Vector-based
  - There are no scalars
- Open-source language
- (Many!) User-contributed packages
- Visualization
VIDIA Dashboard: RStudio Tool
RStudio

Four Rstudio panes: Editor, Environment/History, Plots/Help, Console
Command Line Prompt

• RStudio:
  >

• Jupyter:
  In [integer]
R Practical Matters

RStudio characteristics

- R is case sensitive (R != r)
- Command line prompt is >
- To run R code: use command line, or save script and `source("script_name")`
- To separate commands, use ; or a newline
- The # character marks a non-executed comment
- To display help files: `?<command-name>` or `??<command-name>`
R as a Calculator

> 2 + 3 * 5  # Order of operations

> (2 + 3)*5  # Spaces are optional

On the RStudio command line...
R Output

> 2 + 3 * 5

[1] 17

Q: What's that [1] about?
A: R numbers outputs with [n]

# Try this in the command line:
> 1:500
About Comments

> 2 + 3 * 5    # Order of operations

# A comment is:
# Text useful to humans, ignored by computer
# Helps you understand what code does, or why
# Denoted by a pound sign in R

Use them!!
R as a Calculator

Try these in your RStudio console:

> 4^2    # 4 raised to the second power
> 3/2    # Division
> sqrt(16) # Square root
> 3 - 7   # Subtraction
> log(10) # Natural logarithm
#       with base e=2.718282
Variables: Save It

How do we keep a value for later use?
Variable assignment!

> y = 2 + 3 * 5  # Do some arithmetic
> y              # R stores this value as y
[1] 17

y can be found under Values in the Workspace window
Variable Assignment

> y = 2 + 3 * 5  # R stores this value as y

17

2 + 3 * 5

y can be found under Values in the Workspace window
Naming Variables in R

Variable names may consist of letters, numbers and the dot or underline characters. It should start with a letter. Keep it unique!

**Good:**

> y = 2  
> try.this = 33.3  
> oneMoreTime = “woohoo”

**Bad:**

> 2y = 2  
> _z = 33.3  
> function = “woohoo”

*function* is a reserved word in R
Assign Variables

Try these in your RStudio console:

```r
# make variable assignments
> abc = 3
> Abc = log(2.8) * pi
> ABC = "fiddle"
```

Now, check Workspace: Values
Variables: Save It

Alternate R syntax for assignment

```r
> y = 2 + 3 * 5
> z <- 2 + 3 * 5  # Same thing as y
```

Variable assignment: Use = or <-
R's Atomic Data Types

Let's take a look at some available data types:

- Numeric (includes integer)
  3.14, 1, 2600

- Character (string)
  “hey, I'm a string”

- Logical
  TRUE or FALSE

- NA
  No value known
Numeric Data

Find the type of a variable using class()

```r
> class(8)
[1] "numeric"  # numeric type

> class(6.02e+24)
[1] "numeric"  # numeric type

> class(pi)
[1] "numeric"  # numeric type (predefined in R)
```
Character and Logical Data

Find the type of a variable using class()

> class("phooey")  # character type:
[1] "character"  # notice the quotes

> class(TRUE)  # logical type: no quotes
[1] "logical"

> class(NA)  # NA (no quotes) means “no value known”
[1] "logical"
RStudio Test Flight

To whet your appetite for RStudio, let's try:

- Using the editor
- Entering data
- Making a plot in R
- Sourcing a file
The M&M Exercise

On your workstation:

- Sign in to vidia.ccr.buffalo.edu
- Start the RStudio tool
- Create/Access Project from GitHub
  git://github.com/ubccr/hsws.git
- Files pane: click examples, then mm, then:
  mm-single-example.R
The M&M Exercise

Inside mm-single-example.R:

- Change the M&M color counts in the \textit{mv} variable
- Edit \textit{ptitle}, if you want

```r
# EDIT HERE: ...

mvl = c("red", "blue", "green", "yellow", "orange", "brown")
mv = c(4, 5, 3, 2, 1, 3)
ptitle = "M&Ms in example package"
```
The M&M Exercise

Inside mm-single-example.R:
- Save the file to your home directory (File:Save)
- Source the file (Source button)
The M&M Exercise

Questions:
- What have you plotted?
- What outputs does R provide in the console?
- What variables were created?
- What else happens inside this source file?

OK, now you can eat...
The M&M Exercise

- Distribution of colors across many samples
- Increase the number of samples—reveal the underlying distributions
- Barplot
  - Counts of colors in one sample
- Histogram
  - Instances of color counts across all samples
Using Logical Operators

1==2 # equivalence test: *double equals*
9 != 19 # “not equal” test
3 < 204 # less-than test
18 > 44 # greater-than test
“tree”==89 # comparing mixed data types

What should the results of these tests be?
A Logical Test

Compare R syntax for assignment

> y = 2 + 3 * 5
> z <- 2 + 3 * 5  # Same thing as y

> y==z  # Here's the test...
[1] TRUE
A logical value is often created from a comparison between variables.

\[ u \& v \]  # Are \( u \) AND \( v \) both true?

\[ u \mid v \]  # Is at least one of \( u \) OR \( v \) true?

\[ !u \]  # “NOT \( u \)” flips the logical value of

# variable \( u \)
Learning about Object x

R stores everything, variables included, in Objects.

Objects have attributes, such as: name, dim, class.
Object x

> x <- 2.71
> print(x)  # print the value of the object
[1] 2.71

> class(x)  # what data type or object type?
[1] "numeric"

> is.na(x)  # is.na() tests whether a value has a known value
[1] FALSE
Interlude

Complete variable/atomic datatype exercises.

Open in the RStudio source editor:

<workshop>/exercises/1-exercises-variables-atomic-datatypes.R
Interlude++

Once you have completed the exercises, browse further information about R:

An R tutorial (Check out slides 25-32, then 45-49 for relevant material):
- http://jaredknowles.com/s/Tutorial1_Intro.html

The Vocabulary of R:
- http://adv-r.had.co.nz/Vocabulary.html