# Introduction to R

Yiqing Xu (Stanford)

# Getting Started with R

- Download R from https://cran.r-project.org
- Download RStudio from http://www.rstudio.com
- We will be using `session1.R` (code file) and `nations.RData` (data file)

# Big Picture

1. Data are tables, but
   - Rows and columns
   - There are other types of objects: vectors, matrices
2. We write R scripts to manipulate data
   - Data need to be loaded and saved
   - R scripts need to be loaded and saved for reproducibility
3. Packages are collections of generalizable R scripts (functions)
   - Download and install them from CRAN or Github
   - You can write your own functions or packages
4. You learn to code by coding
   - Breaking things is an essential part of learning
   - Learn with your own pace

# Today's Agenda

- Arithmetic Operations
- Objects
- Vectors
- Functions
- Data Files
- Saving Objects
- Packages
- Programming Tips

# Arithmetic Operations

R can be used as a calculator:

```
5 + 3
```

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

```
5 / 3
```

```
## [1] 1.666667
```

```
5 ^ 3
```

```
## [1] 125
```

# Objects

R is an "object-oriented" programming language. An *object* is any piece of information stored by R. These can be anything, for example:

- A dataset (e.g. a country year dataset)
- A subset of a dataset (e.g. just the democracies in a country year dataset)
- A number (e.g. $2\pi + 1$)
- A phrase (e.g. "Stanford is awesome")
- A function (e.g. a function that takes in $x$ and gives you $x^2 + 8$)

# Objects (cont.)

R can store *objects* with a name of our choice. Use <- as an assignment operator for objects.

```
result <- 5 + 3
result
```

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

If we assign a new value to the same object name, then we will overwite this object (so be careful when doing so!)

```
result <- 5 - 3
result
```

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

# Objects (cont.)

R can also represent other types of values as objects, such as strings of characters:

```
Stanford <- "Stanford is awesome"
Stanford
```

```
## [1] "Stanford is awesome"
```

There are many other classes of data besides numeric and character, which we will talk about in class

# Vectors

A *vector* simply represents a collection of information stored in a specific order. We use the function `c()`, which stands for "concatenate," to enter a data vector (with commas separating elements of the vector):

```
world.pop <- c(2525779, 3026003, 3691173, 4449049,
               5320817, 6127700, 6916183)
world.pop
```

```
## [1] 2525779 3026003 3691173 4449049 5320817 6127700 6916
```

# Vectors (cont.)

To access specific elements of a vector, we use square brackets `[ ]`. This is called *indexing*:

```
world.pop[2]
```

```
## [1] 3026003
```

```
world.pop[c(2, 4)]
```

```
## [1] 3026003 4449049
```

# Vectors (cont.)

Since each element of this vector is a numeric value, we can apply arithmetic operations to it:

```
world.pop * 1000
```

```
## [1] 2525779000 3026003000 3691173000 4449049000 5320817(
```

# Functions

A *function* takes input object(s) and returns an output object. In `R`, a function generally runs as `funcname(input)` where `funcname` is the function name and `input` is the input object. We often call these inputs *arguments*. Some basic functions useful for summarizing data include:

- `length()`: length of a vector (number of elements)
- `min()`: minimum value
- `max()`: maximum value
- `range()`: range of data
- `mean()`: mean
- `sum()`: sum

# Functions (cont.)

```
length(world.pop)
```

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

```
min(world.pop)
```

```
## [1] 2525779
```

```
max(world.pop)
```

```
## [1] 6916183
```

# Functions (cont.)

```
range(world.pop)
```

```
## [1] 2525779 6916183
```

```
mean(world.pop)
```

```
## [1] 4579529
```

```
sum(world.pop)
```

```
## [1] 32056704
```

# Data Files

Most of the time, we will load data from an external file. For this class, we will mostly use:

- *CSV*: comma-separated files. These are conceptually similar to Microsoft Excel or Google Spreadsheet.
- *RData*: collection of `R` objects including data sets.

# Working Directory

- The *working directory* is where R will by default load data from and save data to.
- Use the function `getwd()` to display the current working directory.

```
getwd()
```

```
## [1] "/Users/xyq/Library/CloudStorage/Dropbox/Teaching/_U
```

- Use the function `setwd()` to change the working directory

```
path <- getwd()
setwd(path)
```

- In RStudio, you can also go to Session, Set Working Directory, To Source File Location to set it where your R file is located

# Reading in Files

- For *CSV* files:

```
d <- read.csv("nations.csv")
```

- For *RData* files:

```
# d <- load("d.RData")
```

# Data Frames

A *data frame* is a collection of vectors, but we can think of it like a spreadsheet. Useful functions for data frames include:

- `names()`: return a vector of variable names
- `nrow()`: return the number of rows
- `ncol()`: return the number of columns
- `dim()`: combine `ncol()` and `nrow()` into a vector
- `summary()`: produce a summary

# Data Frames (cont.)

```
names(d)
```

```
## [1] "country" "isocode" "year"    "demo"    "gdppc"
```

```
nrow(d)
```

```
## [1] 192
```

```
ncol(d)
```

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

# Data Frames (cont.)

```
dim(d)
```

```
## [1] 192   5
```

```
summary(d)
```

```
##     country             isocode              year          
##  Length:192         Length:192         Min.   :1960   Mi
##  Class :character   Class :character   1st Qu.:1960   1s
##  Mode  :character   Mode  :character   Median :1984   Me
##                                        Mean   :1984   Me
##                                        3rd Qu.:2009   3r
##                                        Max.   :2009   Ma
##       gdppc        
##  Min.   :   70.0  
##  1st Qu.:  588.4  
##  Median : 1920.2  
##  Mean   : 7321.1  
##  3rd Qu.: 7126.2
```

# Data Frames (cont.)

The $ operator is one way to access variables from a data frame:

```
d$country
```

```
##   [1] "Afghanistan"           "Afghanistan"           "Alb
##   [4] "Albania"               "Argentina"             "Arg
##   [7] "Australia"             "Australia"             "Aus
##  [10] "Austria"               "Belgium"               "Bel
##  [13] "Benin"                 "Benin"                 "Bol
##  [16] "Bolivia"               "Brazil"                "Bra
##  [19] "Burkina Faso"          "Burkina Faso"          "Cam
##  [22] "Cambodia"              "Cameroon"              "Can
##  [25] "Canada"                "Canada"                "Cen
##  [28] "Central African Rep." "Chad"                  "Cha
##  [31] "Chile"                 "Chile"                 "Chi
##  [34] "China"                 "Colombia"              "Col
##  [37] "Congo, Dem Rep"        "Congo, Dem Rep"        "Con
##  [40] "Congo, Rep."           "Costa Rica"            "Cos
##  [43] "Cote d'Ivoire"         "Cote d'Ivoire"         "Cub
```

# Saving Objects

- When you quit `RStudio`, you will be asked whether you would like to save the workspace. You should answer *no* to this in general: we only want to save what we want!
- To export *CSV*:

```
write.csv(d, file = "nations_new.csv")
```

- To export *RData*:

```
save(d, file = "nations.RData")
```

# Packages

One of R's strengths is the existence of a large community of R users who contribute various functionalities as R packages. For example, the foreign package is useful when dealing with files from other statistical software.

```
# install.packages("foreign") # install package
library(foreign) # load package
```

# Programming Tips

- First, use the text editor in `RStudio` to write your code rather than directly typing it into the R console. That way you can save a record of your program as a `R` file.
- Second, we can annotate our `R` code so that it is easily understandable to ourselves and others using #

```
# File: d.R
# Author: Yiqing Xu
# This code loads the UN population data

d <- read.csv("nations.csv")
```

# Programming Tips (cont.)

Third, we should follow a certain set of coding rules:

- Use informative names for files, variables, and functions
- Use systematic spacing and indentation

Finally, `R Markdown` is useful for quickly writing documents using `R`. I used it to make this presentation, and you will be using this to turn in your problem sets.