1 - Intro to R

Packages and Basic Programming

Karsten Maurer, Eric Hare, and Susan VanderPlas

Iowa State University

R Packages

- Commonly used R functions are installed with base R
- R packages containing more specialized R functions can be installed freely from CRAN servers using function `install.packages()`
- After packages are installed, their functions can be loaded into the current R session using the function `library()`

Install Package Demo

- Demo how to install and load the `plyr` package

Finding Packages

- How do I locate a package with the desired function?
- Google ("R project" + search term works well)
- R website task views to search relevent subjects `http://cran.r-project.org/web/views/`
- ??searchterm will seach R help for pages related to the search term
- `sos` package adds helpful features for searching for packages related to a particular topic

Handy Packages

- `ggplot2` : Statistical graphics
- `plyr` / `reshape2` : Manipulating data structures
- `lme4` : Mixed models
- `knitr` : integrate LaTeX, HTML, or Markdown with R for easy reproducible research

Programming

Functions

Creating your own functions isn't too bad!

```
# Basic format
foo <- function(arg1, arg2, ...){
    # Code goes here
    return(output)
}

mymean <- function(dat){
    ans <- sum(dat)/length(dat)
    return(ans)
}
```

Programming

Conditional statements

if/else statements are quite useful.

```
#Basic format
if(condition){
    # Some code that runs if condition is TRUE
}else{
    # Some code that runs if condition isn't TRUE
}
```

```
mymean <- function(dat){
    if(!is.numeric(dat)){
        warning("Numeric input is required")
        return(NA)
    }
    ans <- sum(dat)/length(dat)
    return(ans)
}
```

Programming

Looping

- Reducing the amount of typing we do can be nice
- If we have a lot of code that is essentially the same we can take advantage of looping.
- R offers several loops: for, while, repeat.

```
for(i in 1:3){
    print(i)
}
```

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

Programming

More for

```
id <- c("total_bill", "tip", "size")
for(colname in id){
    print(colname)
}
```

```
## [1] "total_bill"
## [1] "tip"
## [1] "size"
```

```
for(colname in id){
    print(paste(colname, mymean( tips[, colname] ) ) )
}
```

```
## [1] "total_bill 19.7859426229508"
## [1] "tip 2.99827868852459"
## [1] "size 2.56967213114754"
```

Programming

While

```
i <- 1
while(i <= 5){
    print(i)
    i <- i + 1
}
```

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

Your Turn

- Create a function that takes numeric input and provides the mean and a 95% confidence interval for the mean for the data (the t.test function could be useful)
- Add checks to your function to make sure the data is either numeric or logical. If it is logical convert it to numeric.
- Loop over the columns of the diamonds data set and apply your function to all of the numeric columns.

What you've learned!

- Use R for scientific/statistical calculations
- Be able to create or read in data and have the ability to manipulate the data accordingly
- Have the ability to explore data set characteristics and calculate summary statistics for real data sets
- Use the help functionality to find the functions you need to do what you want to do
- Install, use, and search for helpful external packages
- How to use basic programming constructs to make working with data easier

Questions?

- Any Questions???

Feedback Survey

- Please let us know how we did with the feedback survey!
- `http://heike.wufoo.com/forms/r-workshop-your-opinion-matters/`