Data Scientist’s Toolbox /R links and info

### R

http://www.r-project.org/

## RStudio

Main web link - <http://www.rstudio.com/>

Documentation - <http://www.rstudio.com/ide/docs/>

## File types

R script – .R

Reproducible markdown documents - .Rmd

## Markdown

<http://www.rstudio.com/ide/docs/authoring/using_markdown>

Getting help on R

#### **Access help file - tell you the help file for a function**

**?rnorm**

#### **Search help files – doesn’t require full function name**

**help.search( “rnorm” )**

#### **Get arguments for a function**

**args( “rnorm” )**

function (n, mean = 0, sd = 1)

#### Reproduce the code of a function on the console

Type the function name

Rnorm

R reference card

<http://cran.r-project.org/doc/contrib/Short-refcard.pdf>

How to get help when using R – video

<https://www.youtube.com/watch?v=ZFaWxxzouCY&list=PLjTlxb-wKvXNSDfcKPFH2gzHGyjpeCZmJ&index=3>

Where to look for different types of questions

* R programming (see also: <http://bit.ly/Ufaadn>)
  + Search the archive of the class forums
  + Read the manual/help files
  + Search on the web
  + Ask a skilled friend
  + Post to the class forums
  + Post to the [R mailing list](http://www.r-project.org/mail.html) - http://www.r-project.org/mail.html
  + or Post to [Stackoverflow](http://stackoverflow.com/) - http://stackoverflow.com/
* Data Analysis/Statistics
  + Search the archive of the class forums
  + Search on the web
  + Ask a skilled friend
  + Post to the class forums
  + Post to [CrossValidated](http://stats.stackexchange.com/) - http://stats.stackexchange.com/
* Other software - software specific websites

A note on Googling data science questions

* The best place to start for general questions is the forums
* [Stackoverflow](http://stackoverflow.com/) (use the tag "[r]"),[R mailing list](http://www.r-project.org/mail.html) for software questions, [CrossValidated](http://stats.stackexchange.com/) for more general questions
* Otherwise Google "[data type] data analysis" or "[data type] R package"
* Try to identify what data analysis is called for your data type
  + [Biostatistics](http://en.wikipedia.org/wiki/Biostatistics) for medical data
  + [Data Science](http://en.wikipedia.org/wiki/Data_science) for data from web analytics
  + [Machine learning](http://en.wikipedia.org/wiki/Machine_learning) for data in computer science/computer vision
  + [Natural language processing](http://en.wikipedia.org/wiki/Natural_language_processing) for data from texts
  + [Signal processing](http://en.wikipedia.org/wiki/Signal_processing) for data from electrical signals
  + [Business analytics](http://en.wikipedia.org/wiki/Business_analytics) for data on customers
  + [Econometrics](http://en.wikipedia.org/wiki/Econometrics) for economic data
  + [Statistical process control](http://en.wikipedia.org/wiki/Statistical_process_control) for data about industrial processes
  + etc.

Allowing easier software installs on Mac

Go to Apple Menu/System Preferences/Security & Privacy

Click on lock icon in lower left of window and type in your administrator credentials

Change the “Allow apps downloaded from:” setting to “Anywhere”

It is up to you to be sure stuff you install is OK

Installing R

On the web go to cran(Comprehensive R Archive Network) – cran.r-project.org

Click on “Download R for …” link for your platform

On windows, click on the “base” link

Download it and then run the installer

Under windows during setup you can do custom to choose between SMI and MDI (single or multiple windows)

Installing RStudio

You must have R installed before installing RStudio

Go to the RStudio website: <http://www.rstudio.com/>

Click the green button that says “Download”

Download the desktop version

It will autodetect your OS and give you that link as the default

For Mac, open the download and drag app into the applications folder

Installing Git

Go to this web page - <http://git-scm.com/downloads>

Download the installer

Run the installer going with the default options

Open Git Bash from the directory git was installed in or the start menu for Windows users

Configure Username and Email as follows in git bash

git config --global user.name “Your Name Here”

git config --global user.email [your\_email@example.com](mailto:your_email@example.com)

confirm your changes by typing

git config --list

exit – type exit to exit git bash

Setting up a GitHub Account

Go to the GitHub homepage at <https://github.com/>

Enter a username, email, and password and click "Sign up for GitHub"

NOTE: You should use the same email address that you used when setting up Git in the previous lecture

On next screen click “free plan” and then click “Finish sign up”

You will be on a page with some useful tutorials

If you click on your username in the upper right corner you will see your profile which shows all repositories and your activity

You can click “edit profile” if desired to update information

Command line interface(CLI)

On Windows use Git Bash

To find it open the windows menu, click search and type “git bash”

On Mac/Linux use Terminal

To find it open spotlight and search for terminal

Directories and folders are the same thing

Directories are in trees where a directory is inside a parent directory

The directory up from a directory is the parent and is referred to as ..

The root directory is the slash: /

The home directory is under users and referred to as tilde: ~

Your working directory is whatever directory your CLI or application is in now

The path is the set of directories from the root to where you are

Commands can take flags, typically preceded by - to trigger certain behaviors

Commands can take arguments which provides input for the command

pwd will print the working directory

clear will clear the screen in your CLI

ls list files and folder in the working directory

ls -a hidden and unhidden files and folders in the working directory

ls -al lists details for hidden and unhidden files and folders in working directory

cd – changes the working directory to the directory you specify

cd with no argument takes you to your home directory

cd .. takes you up one directory

mkdir – creates a directory

touch – creates an empty file

cp source destination – copies a source file to the destination file

cp –r copies one directory recursively into another directory

rm – removes a file

rm –r will remove a directory and all files in it

mv source destination – moves a source file to a new directory or renames a file

echo prints whatever argument you provide

date prints the date

Introduction to git

Getting started - <http://git-scm.com/book/en/Getting-Started-About-Version-Control>

Git home page - <http://git-scm.com/>

Short history of git - <http://git-scm.com/book/en/Getting-Started-A-Short-History-of-Git>

Git download - <http://git-scm.com/downloads>

Introduction to GitHub

GitHub is a web-based hosting service for git - <http://en.wikipedia.org/wiki/GitHub>

Allows users to "push" and "pull" their local repositories to and from remote repositories on the web

Provides users with a homepage that displays their public repositories

Users' repositories are backed up on the GitHub server in case something happens to the local copies

Social aspect allows users to follow one another and share projects

Creating a Github Repository

The word repo is short for repository

Two methods – create your own from scratch or fork an existing one

# Start a Repository From Scratch

Either go to your profile page (https://github.com/yourUserNameHere/) and click on "Create a new repo" in the upper righthand corner of the page

...OR...

Go directly to https://github.com/new (you'll need to log into your GitHub account if you haven't already done so)

Create a name and brief description. Make the name google-able if you want to share it with others

Select public or private repo. – Private requires a paid account

Check the box “Initialize this repository with a README”

Click the “Create Repository” button at the bottom

# Fork another user’s existing GitHub Repository to your GitHub account

<https://help.github.com/articles/fork-a-repo>

<http://git-scm.com/book/en/Git-Basics-Getting-a-Git-Repository>

Navigate to the desired repository on the GitHub website and click the “Fork” button

You now have a copy of the desired repository on your GitHub account

You need to make a local copy of the repo on your computer

This process is called "cloning" and can be done using the following command:

$ git clone https://github.com/yourUserNameHere/repoNameHere.git

NOTE: This will clone the repository into your current directory.

Creating a local copy of a Github Repository

Open Git Bash

Create a directory where you will store the copy

mkdir ~/test-repo

Navigate to that directory

cd ~/test-repo

Initialize a local git repo there

git init

Link your local repository to the one on GitHub

git remote add origin <https://github.com/yourUserNameHere/test-repo.git>

Basic Git and GitHub commands

Pushing and pulling - http://gitready.com/beginner/2009/01/21/pushing-and-pulling.html



## Adding files

Suppose you add new files to a local repository under version control

You need to let Git know that they need to be tracked

git add . adds all new files

git add -u updates tracking for files that changed names or were deleted

git add -A does both of the previous

You should do this before committing

## Committing

You have changes you want to commit to be saved as an intermediate version

You type the command

git commit -m "message" where message is a useful description of what you did

This only updates your local repo, not the remote repo on Github

## Pushing

You have saved local commits you would like to update on the remote (Github)

You type the command

git push

## Branches

Sometimes you are working on a project with a version being used by many people

You may not want to edit that version

So you can create a branch with the command

git checkout -b branchname

To see what branch you are on type:

git branch

To switch back to the master branch type

git checkout master

## Pull requests

If you fork someone's repo or have multiple branches you will both be working separately

Sometimes you want to merge in your changes into the other branch/repo

To do so you need to send a pull request.

This is a feature of Github.

## For git/GitHub help

Git documentation http://git-scm.com/doc

Github help https://help.github.com/

Google/Stack Overflow are great for GithubBasic Markdown

Extension for markdown file is .md

To create Headings

## This is a secondary heading

### This is a tertiary heading

Unordered lists are created by starting each line with \*

\* this is a line in the list

\* this is another line in the list

\* and yet another line

And it gets turned into

* this is a line in the list
* this is another line in the list
* and yet another line

## Getting markdown help

An introduction to markdown http://daringfireball.net/projects/markdown/

Click the MD button in Rstudio for a quick guide

R markdown <http://www.rstudio.com/ide/docs/authoring/using_markdown>/

(you don't need this until Reproducible Research)

Installing R Packages

When you download R from the Comprehensive R Archive Network (CRAN), you get that ``base" R system

The base R system comes with basic functionality; implements the R language

One reason R is so useful is the large collection of packages that extend the basic functionality of R

R packages are developed and published by the larger R community

## Obtaining R Packages

The primary location for obtaining R packages is CRAN

http://cran.r-project.org/

For biological applications, many packages are available from the Bioconductor Project

<http://bioconductor.org/>

You can obtain information about the available packages on CRAN with the available.packages() function

a <- available.packages()

head(rownames(a), 3) ## Show the names of the first few packages

## [1] "A3" "abc" "abcdeFBA"

There are approximately 5200 packages on CRAN covering a wide range of topics

A list of some topics is available through the Task Views link, which groups together many R packages related to a given topic

<http://cran.r-project.org/web/views/>

## Installing an R Package from an R console

Packages can be installed with the install.packages() function in R

To install a single package, pass the name of the lecture to the install.packages() function as the first argument. The following the code installs the slidify package from CRAN

install.packages("slidify")

This command downloads the slidify package from CRAN and installs it on your computer

Any packages on which this package depends will also be downloaded and installed

To install multiple R packages at once with a single call to install.packages(), Place the names of the R packages in a character vector

install.packages(c("slidify", "ggplot2", "devtools"))

## Installing an R Package from RStudio

On the menu choose Tools:Install Packages. It will allow you to pick a repository and a package from there.

## Installing an R Package from Bioconductor

http://www.bioconductor.org/install/

To get the basic installer and basic set of R packages (warning, will install multiple packages)

source("http://bioconductor.org/biocLite.R")

biocLite()

Place the names of the R packages in a character vector and call biocLite

biocLite(c("GenomicFeatures", "AnnotationDbi"))

## Loading R Packages

Installing a package does not make it immediately available to you in R; you must load the package.

The library() function is used to load packages into R. The following code is used to load the ggplot2 package into R

library(ggplot2)

Any packages that need to be loaded as dependencies will be loaded first, before the named package is loaded. NOTE: Do not put the package name in quotes! Some packages produce messages when they are loaded (but some don't)

After loading a package, the functions exported by that package will be attached to the top of the search() list (after the workspace). Typing the commands below has the following output

library(ggplot2)

search()

## [1] ".GlobalEnv" "package:kernlab" "package:caret"

## [4] "package:lattice" "package:ggplot2" "package:makeslides"

## [7] "package:knitr" "package:slidify" "tools:rstudio"

## [10] "package:stats" "package:graphics" "package:grDevices"

## [13] "package:utils" "package:datasets" "package:methods"

## [16] "Autoloads" "package:base"

## Summary

R packages provide a powerful mechanism for extending the functionality of R

R packages can be obtained from CRAN or other repositories

The install.packages() can be used to install packages at the R console

The library() function loads packages that have been installed so that you may access the functionality in the package

Installing Rtools

This is primarily for Windows users.

## What is Rtools?

A collection of tools necessary for building R packages in Windows

## To Install

Available for download at http://cran.r-project.org/bin/windows/Rtools/

Select the .exe download link from the table that corresponds to your version of R

Note: If you're not sure what version of R you have, open or restart R and it's the first thing that comes up in the console

If you have the most recent version of R, you should select the most recent Rtools download (at the top of the chart)

Once the download completes, open the .exe file to begin the installation

Unless you really know what you are doing, you should just go with the default selections at each step of the installation

There are only two exceptions worth noting:

* If you already have Cygwin installed on your machine, you should follow the instructions given during installation (and linked to here: http://cran.r-project.org/bin/windows/Rtools/Rtools.txt)
* IMPORTANT: You should make sure that the box is checked to have the installer edit your PATH.

## Install devtools

Once the Rtools installation completes, open RStudio

Install the devtools R package if you have not previously done so -If you aren't sure, enter

find.package("devtools") in the console

To install devtools, use

install.packages("devtools")

## Verify Rtools installation

After devtools is done installing, load it using

library(devtools)

Then type

find\_rtools() as shown below

This should return TRUE in the console if your Rtools installation worked properly

# Types of Data Science Questions (in approximate order of difficulty)

# Descriptive

Just describes a set of data

Commonly applied to census data

Description and interpretation are separate steps

Cannot be generalized

Example – http://www.census.gov/2010census/

# Exploratory

Find relationships you didn’t know about

Good for discovering new connections

Useful for defining future studies(to confirm what you find)

Not usually the final say

Should not be used for generalizing/predicting

Correlation does not imply causation

# Inferential

Use a small sample to say something about a bigger population

Commonly the goal of statistical models

Involves estimating both the quantity you care about and your uncertainty about your estimate

# Predictive

Use the data on some objects to predict values for another object

Accurate prediction depends heavily on measuring right variables

In general, more data and simple model works really well

Prediction is very hard

Samples – predict election outcome, Target predicts girl is pregnant

# Causal

Find what happens to one variable when you change another

Normally applies “on average”

They are the “gold standard” for data analysis

# Mechanistic

Understand exact changes in one variable that leads to exact changes in another

Very hard and rarely done, needs no noise in the data

Example – changes in pavement design

Definition of Data

“set of items” you are measuring is the “population”

Data are the values of the qualitative or quantitative variables for the population

Variables are measurements of characteristics of an item

Government has lots of data - [www.data.gov](http://www.data.gov)

Data sharing plans

<https://github.com/>

<http://figshare.com/>

<https://github.com/jtleek/datasharing> - guide to data sharing