Scripting for Data Science in Python and R

SMU Interdisciplinary Master's Degree in Data Science

Unit 3 - I. an introduction to the week

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University

Scripting for Data Science in Python and R

SMU Interdisciplinary Master's Degree in Data Science

Unit 3 - II. working with the scripting environment

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University

python environments

- use the anaconda distribution to create and switch environments
- each environment is like a fresh install
- but retains links to master packages when possible

why install environments?

- don’t want to mess up your system architecture
- you might want to install older versions of packages but don’t want to delete currently installed versions
- you might want different versions of python!

the conda environment

1. name environment
2. tell conda what packages to install
3. switch to environment

   tip: if you don't switch, you are **root**

4. install additional packages as needed

```
conda create --name MyFirstEnv numpy
conda create --name MyPython2Env python=2 numpy scipy

source activate MyPython2Env
conda install jupyter

source deactivate
```

conda packages

- not installed yet on conda for your operating system?
- use the conda cloud to find the right package for the right system

```
eclarson$ anaconda search -t conda rpy2
Packages:
     Name                          |  Version | Package Types   | Platforms
     ------------------------------ |  ------- | --------------- | ---------------
     Richarizardd/rpy2             |    2.5.6 | conda           | win-64
                                   :          : Python interface to the R language (embedded R)
     andywocky/rpy2                |    2.5.6 | conda           | osx-64
                                   :          : Python interface to the R language (embedded R)
     asmeurer/rpy2                 |    2.7.0 | conda           | linux-64, linux-32, osx-64
                                   :          : Python interface to the R language (embedded R)
     bce/rpy2                      |          | conda           | linux-64
                                   :          : Python interface to the R language (embedded R)
     bioconda/rpy2                 |    2.7.8 | conda           | linux-64, osx-64
                                   :          : Python interface to the R language (embedded R)
       r-old/rpy2                  |    2.4.2 | conda           | linux-64
       r/rpy2                      |    2.8.2 | conda           | linux-64, win-32, win-64, linux-3
       ralexx/rpy2                 |    2.7.6 | conda           | osx-64

eclarson$ conda install -c r rpy2
```

jupyter notebooks and conda environments

Scripting for Data Science in Python and R

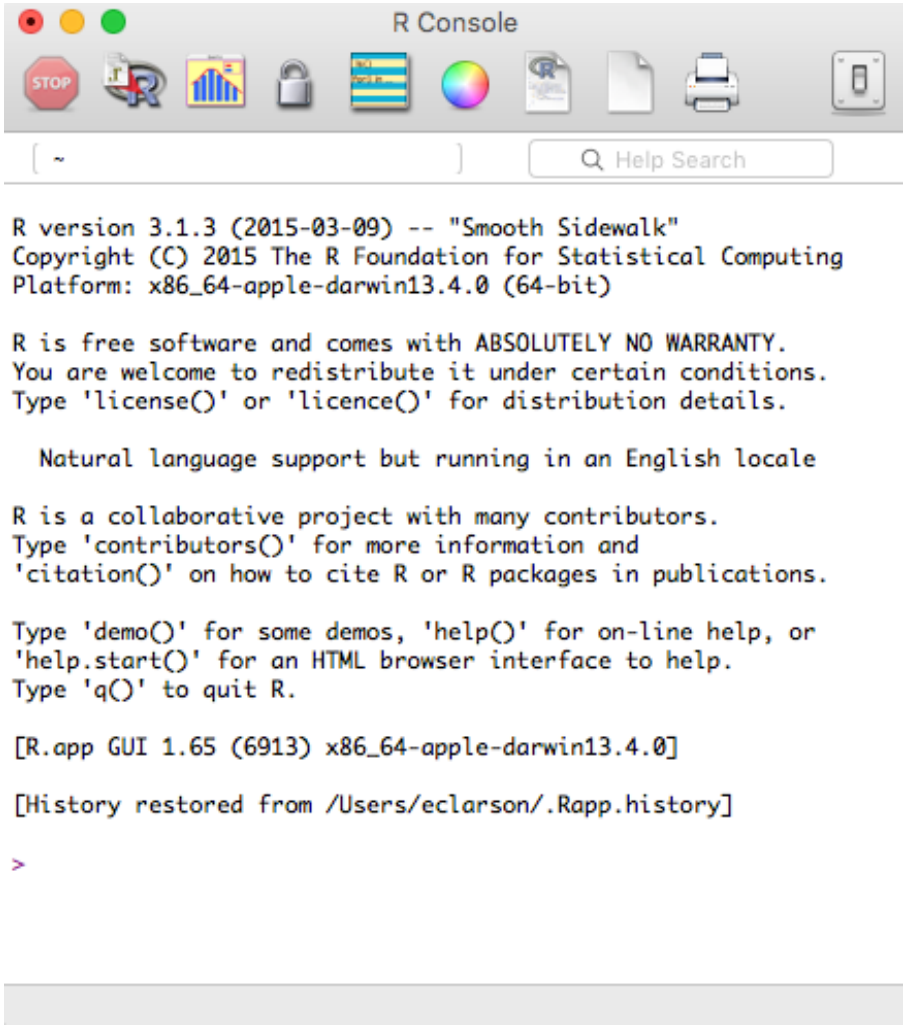
SMU Interdisciplinary Master's Degree in Data Science

Unit 3 - III. basic R syntax

Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University

installing R

- download and install from
  - https://www.r-project.org
  - use a CRAN mirror
- can use console
  - this is a “Session”
  - session has workspace
  - all memory saved in workspace
    - *a lot like jupyter does*



comments, variables, types in R


```
> x = 10 # this is assignment
> class(x) # what is the type of x?
[1] "numeric"
>
> y = as.integer(10)
> class(y)
[1] "integer"

> z = 10 + 0i
> class(z)
[1] "complex"

> j = sqrt(-1)
Warning message:
In sqrt(-1) : NaNs produced
> j = sqrt(as.complex(-1))
> j
[1] 0+1i
```

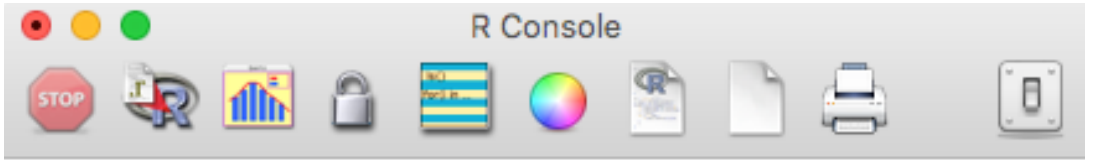
Comment given by #


```
> u=TRUE
> v=FALSE
> class(u)
[1] "logical"
> u&v
[1] FALSE
> u|v
[1] TRUE
> !v
[1] TRUE

> fname="Eric"
> lname="Larson"
> paste(fname,lname)
[1] "Eric Larson"
```

vectors in R

- Denoted by “c”


```
> vec = c(10,5,7)
> class(vec)
[1] "numeric"
> length(vec)
[1] 3

> comb_vec[1]
[1] "a"
> comb_vec[0]
character(0)
```




```
> str_vec = c("a","b","c")
> num_vec = c(10,5,7)
> comb_vec = c(str_vec,num_vec)
> comb_vec
[1] "a"  "b"  "c"  "10" "5"  "7"
> comb_vec[-2]
[1] "a"  "c"  "10" "5"  "7"
> series = c(1.0,5.6,3600)
> names(series) = c("Day","Temp","Sec")
> series
  Day   Temp    Sec
  1.0    5.6 3600.0
```

remove second element

conditionals and loops in R


```
> x = 1:10
> z = c()
> for(xi in 1:10) {
+     if(x[i] < 5) {
+         z = c(z, x[i] - 1)
+     } else {
+         z = c(z, x[i] / 2)
+     }
+ }
> z
 [1] 0.0 1.0 2.0 3.0 2.5 3.0 3.5 4.0 4.5 5.0
```


```
> if(1==0) {
+     print(1)
+ } else {
+     print(2)
+ }
[1] 2
```

highly similar to python syntax

defining functions R


```
> myfct = function(x1, x2=5) {
+      z1 = x1/x1
+      z2 = x2*x2
+      myvec = c(z1, z2)
+      return(myvec)
+ }
> myfct
function(x1, x2=5) {
     z1 = x1/x1
     z2 = x2*x2
     myvec = c(z1, z2)
     return(myvec)
}
> myfct(x1=2, x2=5)
[1]  1 25
> myfct(2, 5)
[1]  1 25
> myfct(x2=5, x1=2)
[1]  1 25
```


```
> myfct2 = function(x1=5, opt_arg) {
+      if(missing(opt_arg)) { #missing?
+           z1 = 1:10
+      } else {
+           z1 = opt_arg
+      }
+      cat("my function returns:", "\n")
+      return(z1/x1)
+ }
```

installing and using libraries in R

calling R from python: Rpy2

using R to extend the calculator