R Training

A beginner’s guide to using R on Sapelo2

Goals for this Training

1. Have R ready to use on your machine either through Sapelo2 or locally
2. Recognize and create the basic R objects: dataframes,vectors,list
3. Interact with Filesystem and Load data into your environment
4. Get quick info on the data through built in functions
5. Manipulate dataframes and create new dataframes from old
6. Use other's code, I.e packages, to expand your options
7. Save/export data and figures
8. Run a batch job using R script on Sapelo2

Features of the R language

1.High-level language: Intuitive and easier to code.

2.R functions are vectorized, which allows for more efficient coding

3.Extensible via packages:Machine Learning, Genetics, High
4.Performance Computing and genetics

5.Graphics

6.Comparable to SAS, SPSS, and Stata, but FREE

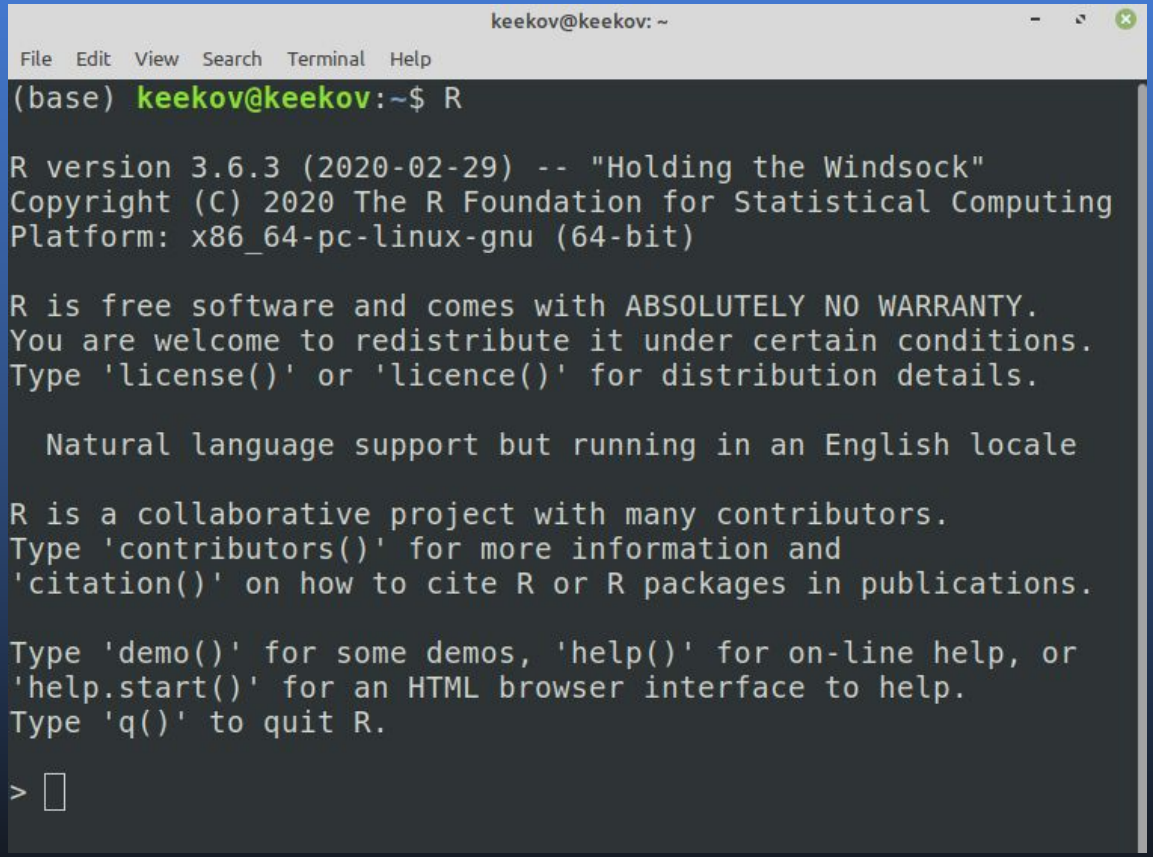
7.Easy creation of documents such as MS-Word, PDF, websites and more.

Downloading R

The R project website has installers for Microsoft Windows, Apple OSX, and Linux

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

Once R is installed, it can be run in the command line.



Running on Sapelo2

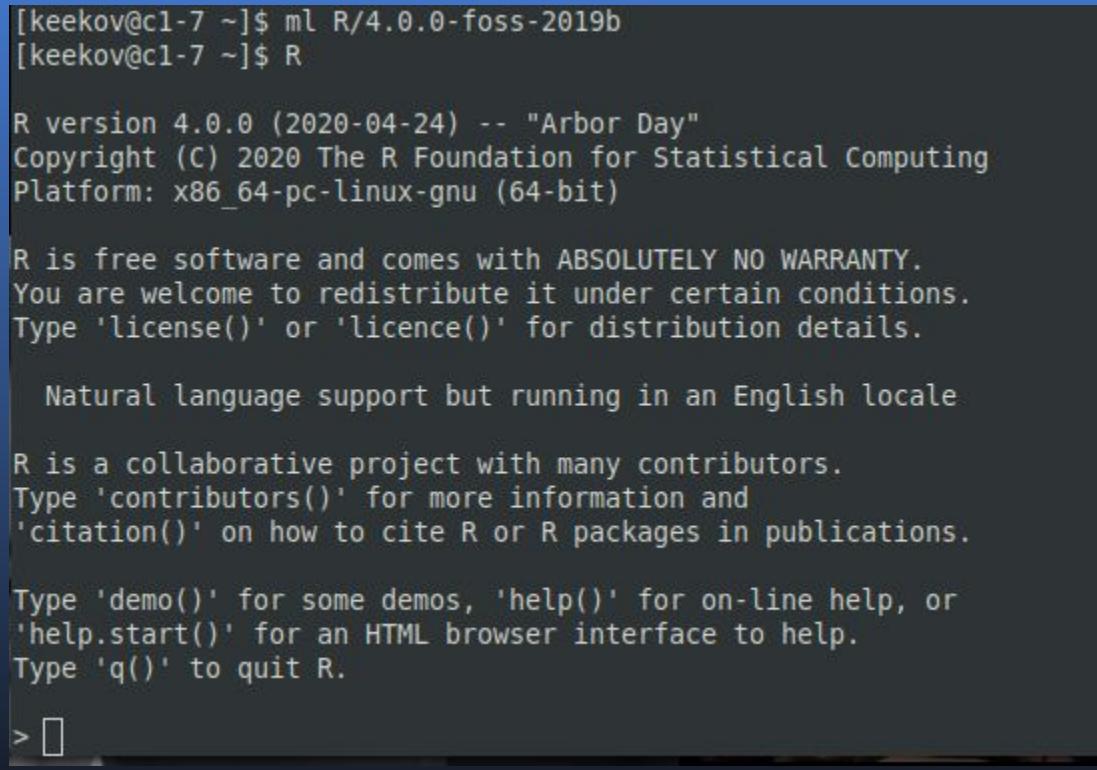
On Sapelo2, first login to the interactive node using the qlogin command.(x forwarding uses xqlogin) Then enter the command :

ml R/4.0.0-foss-2019b

to load the R module.

Then type R and press enter.

This tutorial uses xqlogin for visuals



Packages

R packages are collections of functions and data sets developed by the community.

R installs a set of packages during installation. foundational or core packages offer basic functionality in terms of data manipulation, statistical tests, analysis, and visualization and are loaded when starting R.

Other packages have to be installed and loaded explicitly.

On Sapelo: you can request a package be installed or use your home directory:

https://wiki.gacrc.uga.edu/wiki/Installing_Applications_on_Sapelo2#How_to_install_R_packages

Locally: install.packages("Package Name") followed by library("package Name")

Bioconductor is managed separately → installation of packages involves a different process

Data types

Object Oriented Language: Everything is an object, which has a type and belongs to a class.

Objects have attributes and methods

Common Data types:

character	“Hello” , “x”
Numeric	1, 30.3, 1/3
Logical	TRUE, FALSE
vector	c(12, 34, 129, 16)
list	list(12, TRUE. “green”)

Create an Object with = or <-

Data Creation

Create an Object with = or <-

```
>
> variable1 = 3
> variable2 = 4
> variable4 <- TRUE
> variable3 <- "Hello"
> List = list(3,4,TRUE,"Hello")
>
```

Functions to explore data:

class() - what kind of object is it
length() - how long is it?
attributes() - does it have any metadata?
ls() - shows what variables you have already
object.size() - about how many bytes is object?
rm() - removes an object

```
> class(variable1)
[1] "numeric"
> class(variable3)
[1] "character"
> class(variable4)
[1] "logical"
> class(List)
[1] "list"
> length(variable1)
[1] 1
> length(List)
[1] 4
>
```

Let’s make a dataframe

Most common data structure for tabular data is the Dataframe

Combine vectors with data.frame()

```
> Animal = c("snake","chicken","human")
> NumberOfHands = c(0,0,2)
> WarmBlooded = c(FALSE,TRUE,TRUE)
> ExampleDataFrame = data.frame(Animal,NumberOfHands,WarmBlooded)
> ExampleDataFrame
   Animal NumberOfHands WarmBlooded
1   snake             0       FALSE
2 chicken             0        TRUE
3   human             2        TRUE
>
```

Add and reference columns with $

```
> ExampleDataFrame$Animal
[1] snake   chicken human
Levels: chicken human snake
> ExampleDataFrame$MultipleSpecies = c(TRUE,TRUE,FALSE)
> ExampleDataFrame
   Animal NumberOfHands WarmBlooded MultipleSpecies
1   snake             0       FALSE            TRUE
2 chicken             0        TRUE            TRUE
3   human             2        TRUE           FALSE
>
```

Use ? in front on a function to open documentation

Close the documentation by typing q.

Functions can have arguments(inputs). Specify arguments by position, by complete name, or by partial name.

The following are the same:

mean(exampledata,0.5,FALSE)
mean(x=exampledata,trim=0.5,na.rm=FALSE)
mean(trim=0.5,x=exampledata,na.rm=FALSE)

The arguments can only be out of order if they are named!

Example: ?mean

```
mean                    package:base                    R Documentation

Arithmetic Mean

Description:

     Generic function for the (trimmed) arithmetic mean.

Usage:

     mean(x, ...)

     ## Default S3 method:
     mean(x, trim = 0, na.rm = FALSE, ...)

Arguments:

       x: An R object.  Currently there are methods for numeric/logical
          vectors and date, date-time and time interval objects.
          Complex vectors are allowed for 'trim = 0', only.

    trim: the fraction (0 to 0.5) of observations to be trimmed from
          each end of 'x' before the mean is computed.  Values of trim
          outside that range are taken as the nearest endpoint.

   na.rm: a logical value indicating whether 'NA' values should be
          stripped before the computation proceeds.

     ...: further arguments passed to or from other methods.
```

Interacting with the file system

getwd() , setwd() - get and set the current directory

list.files() - List the files in the current directory or a specified location.

Example: list.files("/usr/local/training/")

file.remove(), file.rename(), file.copy(), dir.create() - file manipulation

Example: file.copy("/usr/local/training/R/testRscript.R","./")

We will need ML_Data.csv, testRscript.R and Rsub.sh

Importing data

The most common type of data is csv

read.delim() for txt files, read.csv() for csv , read_excel() for excel.

Data located at /usr/local/training/RTraining/ML_Data.csv

We will use data from the Kaggle Machine Learning & Data Science Survey. Kaggle is useful for practice data. The page for the data we will be using can be found here:

https://www.kaggle.com/c/kaggle-survey-2019/overview

Quick Analysis

Useful Data Frame Functions:

head() - shows first 6 rows

tail() - shows last 6 rows

dim() - returns the dimensions of data frame

ncol() - number of columns

str() - structure of data frame - name, type and preview of data in each column

names() or colnames() - both show the names attribute for a data frame

table() - builds a contingency table of a column

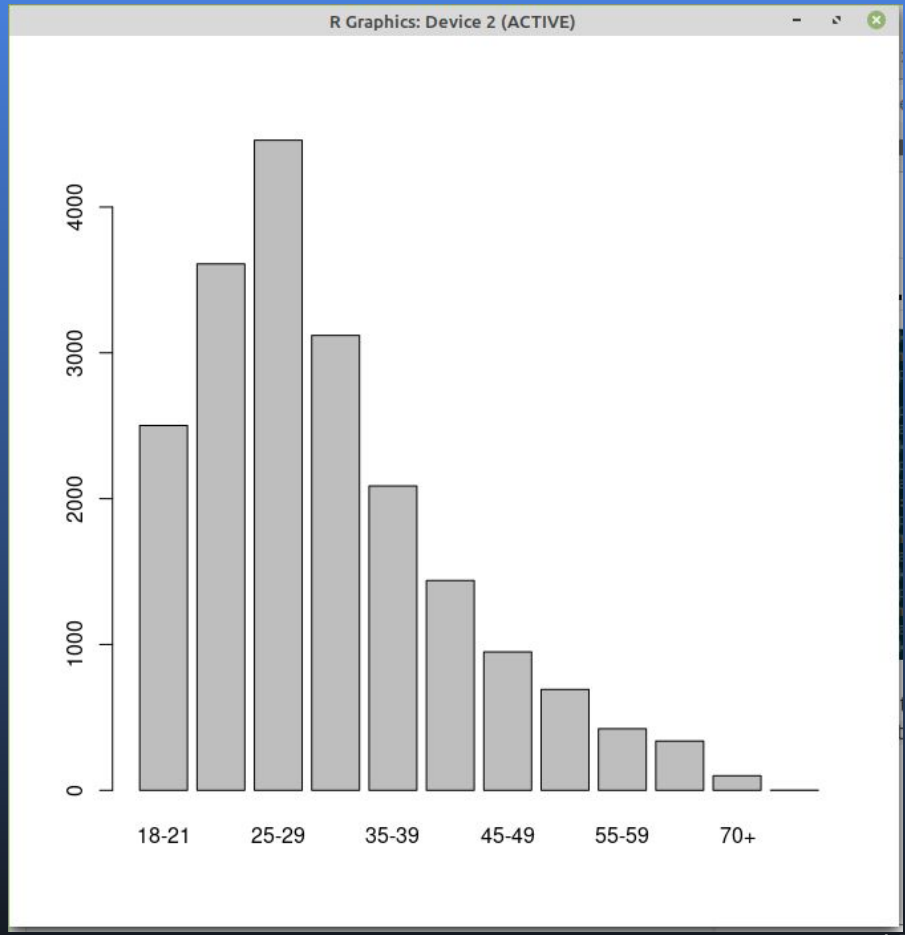
Checking data distribution

Subset using brackets:

Dataframe[rows,columns]

Plot the distribution of ages(Q2):

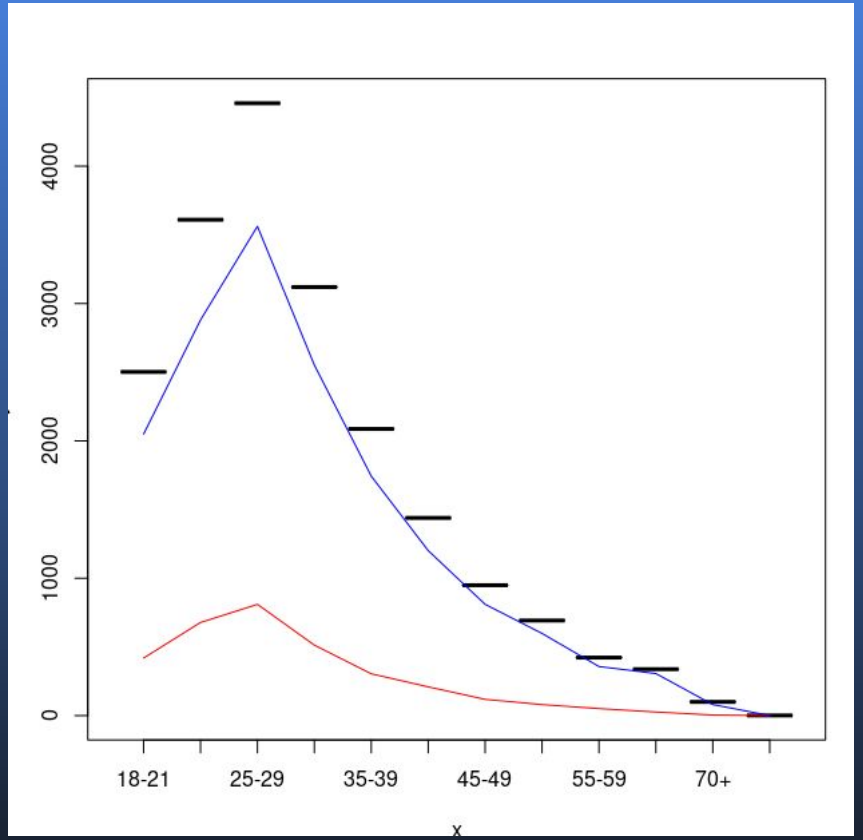
```
plot(responses[,2])
```



DPLYR

- `select`: return a subset of the columns of a data frame, using a flexible notation
- `filter`: extract a subset of rows from a data frame based on logical conditions
- `arrange`: reorder rows of a data frame
- `rename`: rename variables in a data frame
- `mutate`: add new variables/columns or transform existing variables
- `summarise` / `summarize`: generate summary statistics of different variables in the data frame
- `%>%`: the “pipe” operator is used to connect multiple verb actions together into a pipeline

```
>
> plot(gendersummary$Q1,gendersummary$count)
> lines(gendersummary$Q1,gendersummary$Females,col="red")
> lines(gendersummary$Q1,gendersummary$Males,col="blue")
>
```



Exporting Figures and Data

Figure output types include PDF, JPEG, PNG, SVG

Saving pdf Example:
pdf(“rplot.pdf”)
hist(yfData[,5])
dev.off()
write.csv() saves data as .csv

The workspace is your current R working environment and includes any user-defined objects. At the end of an R session, the user can save an image of the current workspace. They can also save their workspace or load a workspace using

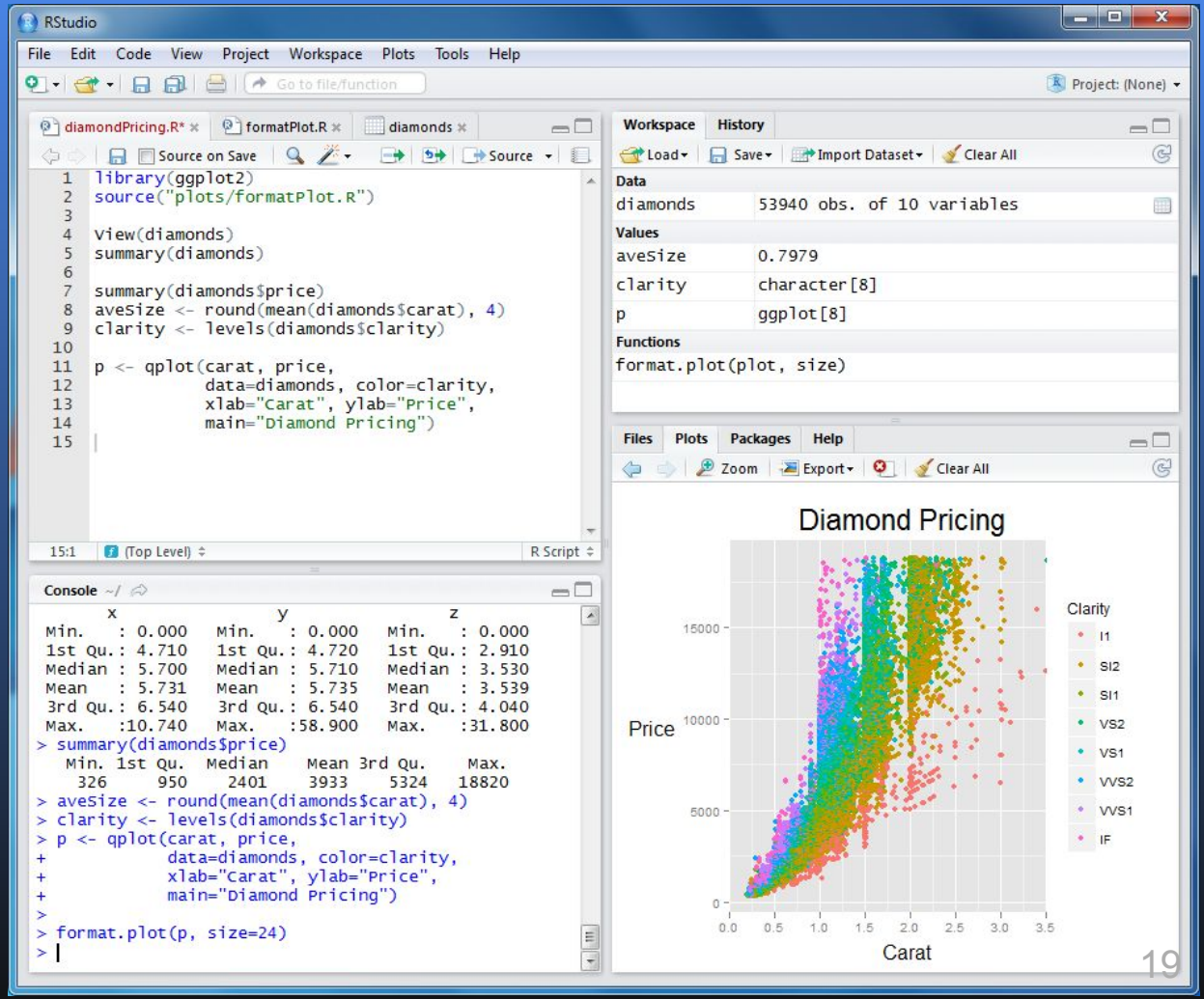
```
save.image(“myfile.RData) If unnamed will be hidden file! Show with ls -a

load("myfile.RData")
```

Example Script located at /usr/local/training/R/Rsub.sh

Includes console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

UI for connecting to git, managing packages and project management.

R studio desktop on Sap2



R markdown supports outputs such as HTML, PDFs, MS-Word documents, applications, websites and more.

chunks.Rmd

Knit HTML | Chunks

```
1  R Code Chunks
2  ==========
3
4  With R Markdown, you can insert R code
   chunks including plots:
5
6  ```{r qplot, fig.width=4, fig.height=3,
   message=FALSE}
7  # quick summary and plot
8  library(ggplot2)
9  summary(cars)
10 qplot(speed, dist, data=cars) +
11       geom_smooth()
12 ```
13
```

RStudio: Preview HTML

Preview: ~/chunks.html | Save As | Publish

R Code Chunks

With R Markdown, you can insert R code chunks including plots:

```
# quick summary and plot
library(ggplot2)
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2
##  1st Qu.:12.0   1st Qu.: 26
##  Median :15.0   Median : 36
##  Mean   :15.4   Mean   : 43
##  3rd Qu.:19.0   3rd Qu.: 56
##  Max.   :25.0   Max.   :120
```

```
qplot(speed, dist, data = cars) + geom_smooth()
```