ACTSC 623 Tutorial

-Introductory session on R/RStudio

-Shirley Liu & Xintong Li

Install R/RStudio

- R is a **programming language for data analysis and statistics**. It is free, and very widely used by professional statisticians. It has many built-in functions and libraries, and is extensible, allowing users to define their own functions. It can be download and installed at http://cran.ma.imperial.ac.uk

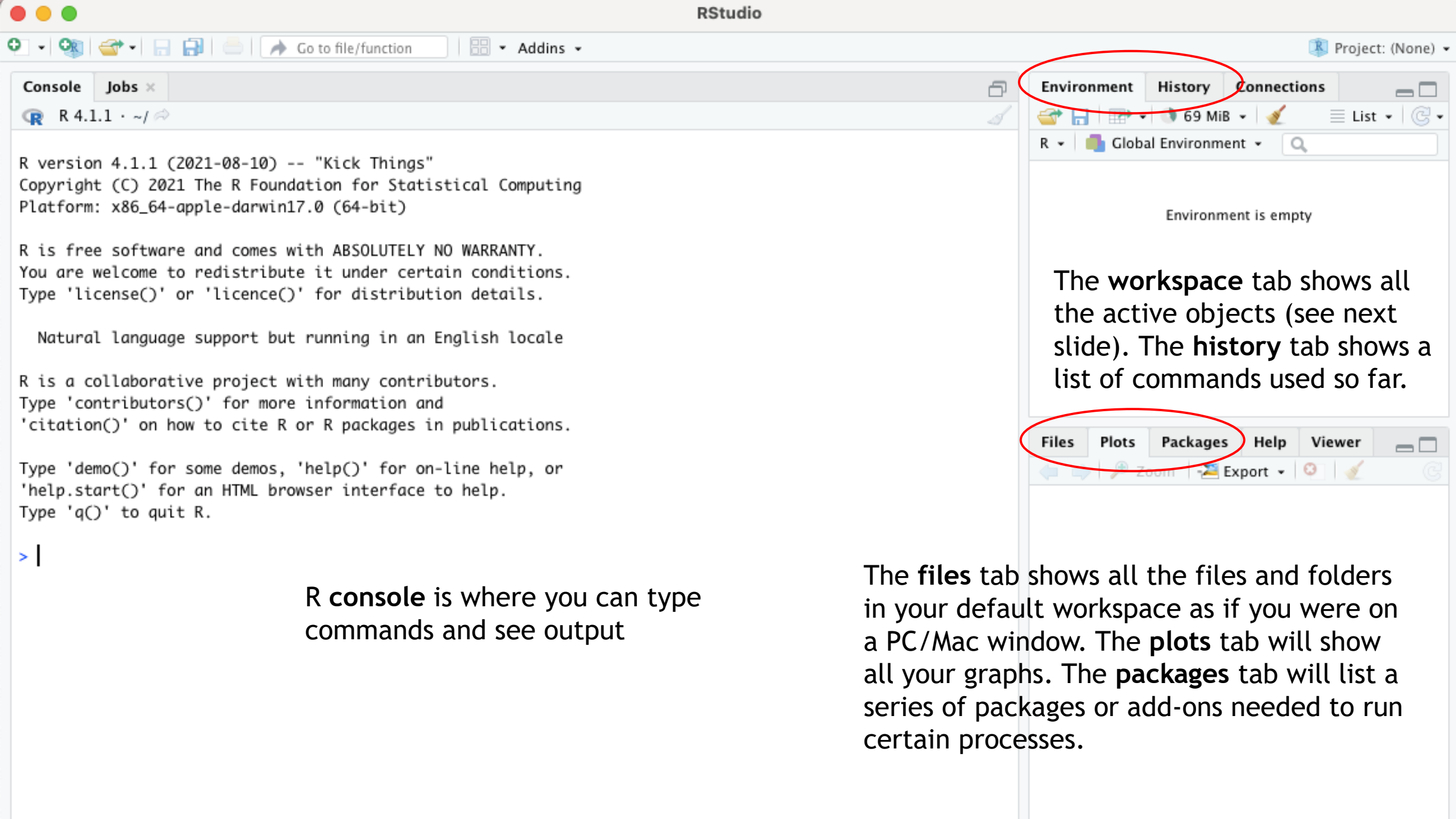
- RStudio is an integrated development environment (IDE) for R, allowing the user to run R script in a more user-friendly environment. It is open-source and available at http://www.rstudio.com/

- More resources can be found at:

✓ https://www.r-project.org

✓ https://www.rstudio.com/resources/cheatsheets/

✓ https://cran.r-project.org/doc/contrib/Hiebeler-matlabR.pdf


RStudio
Go to file/function
Addins
Project: (None)
Console
Jobs
R 4.1.1 · ~/
R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
69 MiB
List
R
Global Environment
Environment is empty
The workspace tab shows all the active objects (see next slide). The history tab shows a list of commands used so far.
Files
Plots
Packages
Help
Viewer
Export
R console is where you can type commands and see output
The files tab shows all the files and folders in your default workspace as if you were on a PC/Mac window. The plots tab will show all your graphs. The packages tab will list a series of packages or add-ons needed to run certain processes.

Some basic commands

- Mathematical calculations (addition, subtraction, multiplication, and division)

```
> 3+1
[1] 4
> 2*2
[1] 4
```

- Built-in functions

```
> sqrt(25)-1
[1] 4
> min(2,4,6)
[1] 2
> sum(2,4,6)
[1] 12
> max(2,4,6)
[1] 6
```

Using functions/packages

- R has many useful functions "built in" and ready to use as soon as R is loaded.

- In addition to R standard functions, additional functionality can be loaded into R using libraries. These include specialized tools for areas such as sequence alignment, read counting etc.

- Utilize help/documentation in RStudio. If you need to see how a function works, try **?** in front of the function name.
- ✓ E.g. type ***?sqrt*** in R console, you will find documentation for function *sqrt* on the bottom right of RStudio workspace (where the File/Plots/Packages/Help tabs locate)

Defining variables

- As with other programming languages and even graphical calculators, **R** makes use of **variables**. A **variable** stores a value as a letter or word.

- In **R**, we make use of the operator **<- or =**

```
> x <- 10
```

- Now **x** holds the value of 10

```
> x
[1] 10
```

- You may also define variables as a string

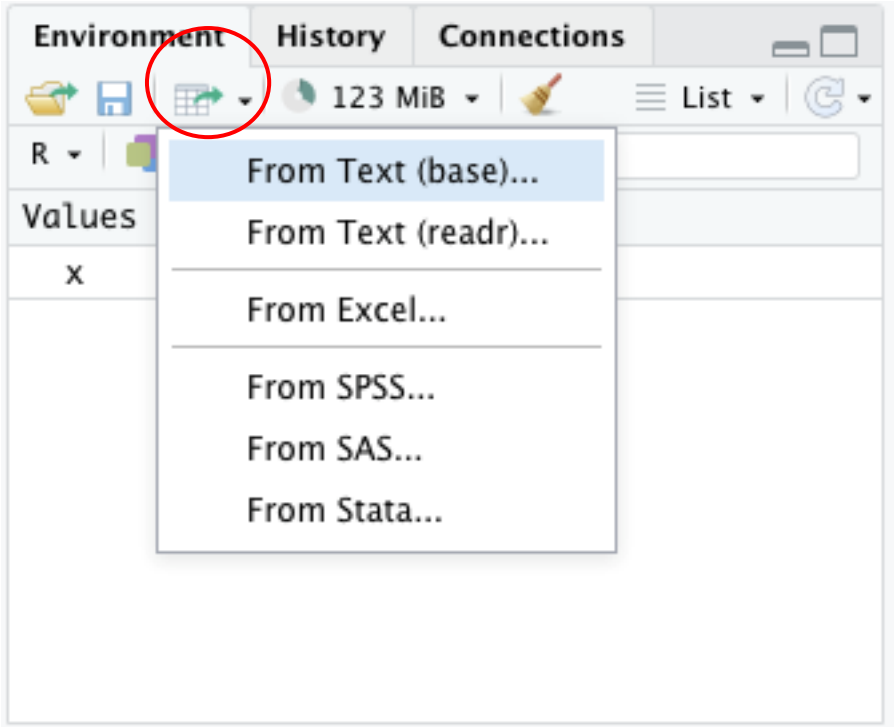
```
> x <-"good"
> x
[1] "good"
```

- ✓ Names are case sensitive
- ✓ Do not use existing function names
- ✓ Variables can be redefined. Only the latest one will be stored.

Import Datasets

- 1. Use function ***read.csv***
- ✓ read.csv(file, header = TRUE, sep = ",", quote = "\"", dec = ".", fill = TRUE, comment.char = "", ...)

- 2. Use existing icon (chose a proper source type)

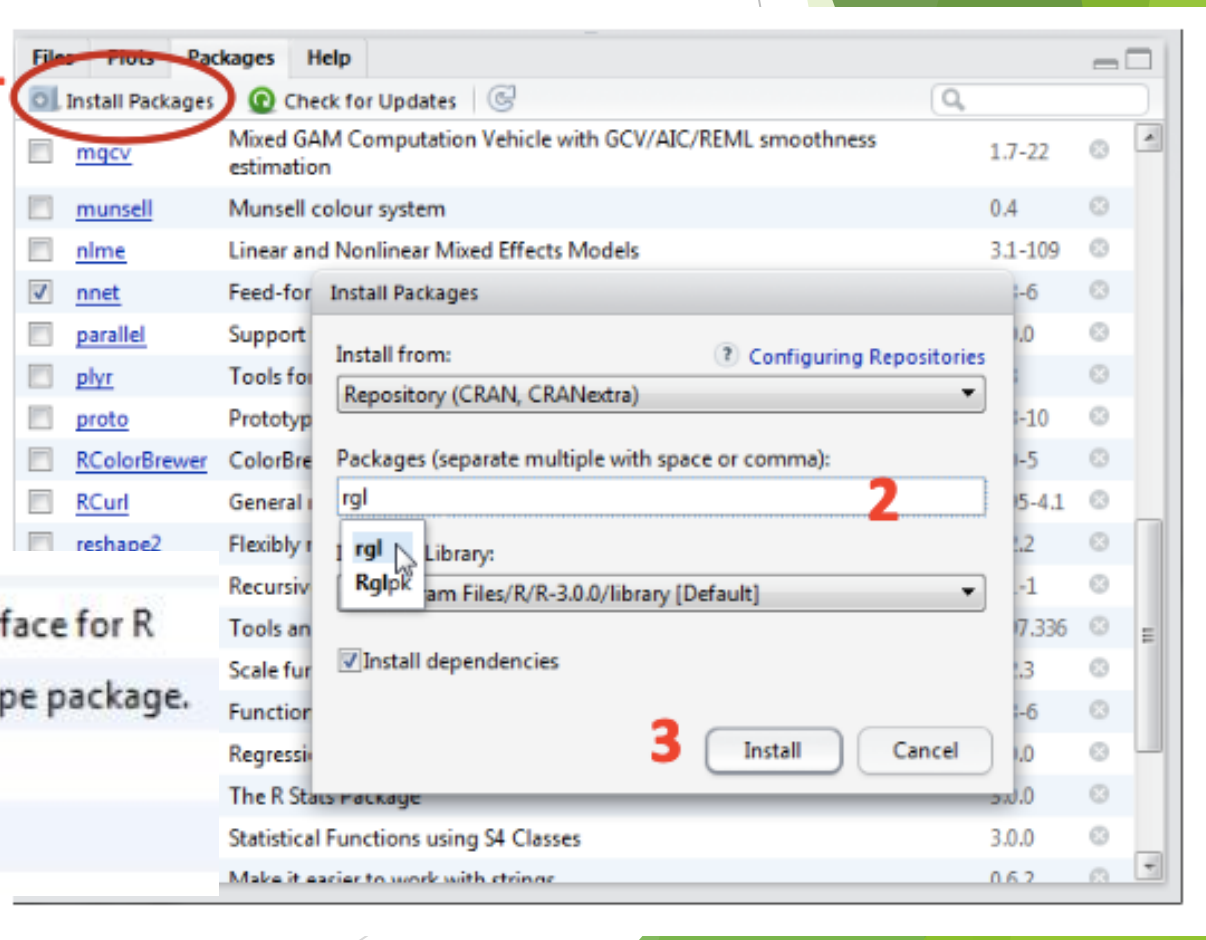


Install packages

- Sometimes, you may want to install packages in addition to the existing ones in R. Those packages can contains useful functions.
- For example, you may follow the steps shown in the screenshot to install package ‘rgl’.
- To use it, remember to check its box or type

  library(*package_name*)

  in the R console.



Different types of data

- Vector
- List
- Matrix
- Data frame

1. Vector

- A vector is an ordered collection of values. It can be defined by

```
> x <- c(10,3,5,6,8,2,9)
> length(x)
[1] 7
```

- Other functions such as seq() and rep() can also be used to create vectors. Each value in the vector has an index, which can be used as references.

```
> x[4]
[1] 6
> x[c(4,5)]
[1] 6 8
> x[-1]
[1] 3 5 6 8 2 9
> x[-c(1,2)]
[1] 5 6 8 2 9
```

- Basic arithmetic operations still works for vectors (vector addition and subtraction requires same vector length)

2. List

- A list is collection of elements which can be of different types.
- To create a list, we can simply use the list() function with arguments specifying the data we wish to include in the list.

```
> l <- list(x = 1:5, y = c("a", "b", "c"))
> l
$x
[1] 1 2 3 4 5

$y
[1] "a" "b" "c"
```

- To retrieve an element from a list in R , use two square brackets [[]].

```
> l[[1]]
[1] 1 2 3 4 5
```

- The $ sign may be used to extract named elements from a list.

```
> l$y
[1] "a" "b" "c"
```

3. Matrix

- A matrix can be created using the matrix() function with the arguments of nrow and ncol specifying the number of rows and columns, respectively.

```
> m <- matrix(1:9, nrow = 3, ncol = 3)
> m
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

- Use dim(), nrow(), ncol() functions to find the dimension of a matrix.

```
> dim(m)
[1] 3 3
```

- A matrix can be created from multiple vectors or other matrices. Use cbind() function to attach data to a matrix as columns.

```
> x = seq(100, 200, by = 50)
> cbind(m, x)
                x
[1,] 1 4 7 100
[2,] 2 5 8 150
[3,] 3 6 9 200
```

4. Data frame

- Data frame is a special case of a list where all elements are the same length. To create a data frame we can simply use the data.frame() function.

```
> df <- data.frame(name = c("John", "Mary", "Ben"), Age = c(20, 18, 25))
> df
  name Age
1 John  20
2 Mary  18
3  Ben  25
```

- Data frames may be indexed with [].

```
> df[2, ]
  name Age
2 Mary  18
> df[, 2]
[1] 20 18 25
```

- We can use advanced index to filter data with $ to specify column.

```
> df[df$Age >= 20, ]
  name Age
1 John  20
3  Ben  25
```

Probability distribution

- For distribution xxx, we can calculate the following:

- ✓ dxxx(x,) returns the density or the value on the y-axis of a probability distribution for a discrete value of x
- ✓ pxxx(q,) returns the cumulative density function (CDF) or the area under the curve to the left of a quantile q on a probability distribution curve
- ✓ qxxx(p,) returns the quantile value given probability p.

```
> qnorm(0.975, mean = 0, sd = 1)
[1] 1.959964
> pnorm(1.959964, mean = 0, sd = 1)
[1] 0.975
> dnorm(0.975, mean = 0, sd = 1)
[1] 0.2480187
```

Simulation

- R is able to simulate values under specific distributions/models using the built-in functions.
- E.g. To simulate 1,000 values with an Uniform (0, 1)

```
> runif(1000, min = 0, max = 1)
```

- Similarly, you can do simulations using rnorm, rlnorm, rpois, rbinom …… for various distributions.

- Keep in mind that simulations are randomly generated every time you run the codes. If you want to obtain same set of values, use the following command BEFORE your simulations

```
> set.seed(123)
```

- Different seed values will generate different outcome. But same seed will always give same results.

Some useful resources

- An Introduction to R https://intro2r.com
- R Tutorial http://www.r-tutor.com/r-introduction
- Documentation browser https://rdrr.io/r/
- Official website for R/RStudio

✓ https://www.r-project.org

✓ https://www.rstudio.com/resources/cheatsheets/

✓ https://cran.r-project.org/doc/contrib/Hiebeler-matlabR.pdf

Q&A?

Good luck and have a great term!