**INTRODUCTION TO R**

1. **WHY R?**

R is a data analysis programming language that let’s you analyze your data and do much more.

I have been using R for the past 8 years while 99.99% of the time R was sufficient for all my programming needs (including the ones require codes longer than thousand lines). In the past I have used many different programming languages (C, Fortran, Matlab, IDL, etc). Eventually I chose R over these languages mainly because: C and Fortran type programming languages are cumbersome (i.e., it requires much longer lines of codes to do the same thing R can do) and they often require special compilers and libraries that need to be installed separately (i.e., in some cases these libraries may not be found or installed very easily for computers using Windows operating system). Even though MATLAB and IDL programming languages are very practical (writing the codes and installation to Windows-based computers) they are not free!! This becomes a major obstacle for many people. It is true that students may install MATLAB within the campus for free using the agreement the University makes with MATLAB (= University pays!!!); but once students graduate and start working in an office environment they often do not find licensed MATLAB copy (because it is very expansive). As a result, they try to do their analysis in very primitive and cumbersome ways using other programming languages (e.g., MS Excel). On the other hand, R is completely free, it is very practical (single line of code can handle can handle lots of computations), can be installed over and over at any computer, and have almost 10,000 packages you may download for free (similar MATLAB toolboxes require additional several hundred or thousand dollars!!!) that enables you to perform different analysis, and has world-wide large user community (it is very easy to find documents or solutions from user forums). For all of these reasons, I do advise you to learn and use R so that you may use it in the future to solve engineering/data analysis problems.

1. **INSTALLING R**

Here very briefly installation instructions over Windows Operating System will be explained. For Max and Linux downloadable binaries and installation instructions are given at below link

<https://cran.r-project.org/>

For Windows-based, first click below link

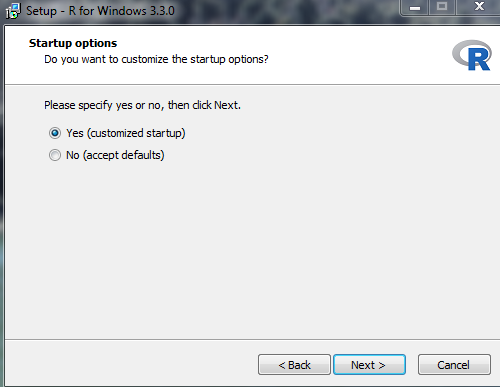
<https://cran.r-project.org/bin/windows/base/>

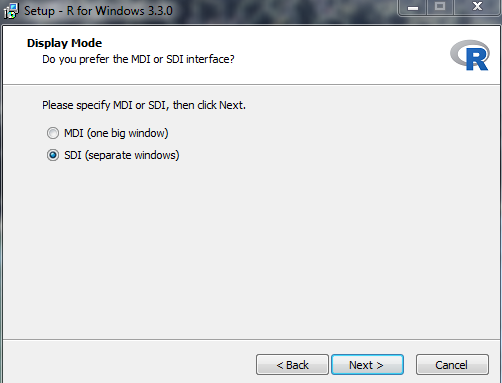
Click “[Download R 3.3.0 for Windows](https://cran.r-project.org/bin/windows/base/R-3.3.0-win.exe) (62 megabytes, 32/64 bit)” (3.3.0 is the latest version at the time this document is created). Or copy/past below link

<https://cran.r-project.org/bin/windows/base/R-3.3.0-win.exe>

Once the executable file is downloaded, double click to install it. Very familiar installation windows will pop-up. The only detail you need to pay attention during the installation is shown below:

Once R is installed, R console will have several different windows (for commend line, plotting, history, etc). Here if “No (accept defaults)” option is selected during installation, then all these windows will be embedded within a single large window, while selecting “Yes (customized startup)” option will let you split all of these windows (I always chose this customized window “SDI (separate windows)” option). Just install R twice (with and without this customized option) to see what exactly the difference between these options.





OK, just follow the steps and complete the installation. Now you should be able to write your first comment.

1. **INSTALLING PACKAGES**

R has thousands of packages (you may call toolboxes) freely available to you (if you are using MATLAB such kind to packages or toolboxes may cost thousands of dollars to you in addition to the thousands of dollars you might have already paid for the MATLAB license). All of these packages enable you to perform different analysis (just click “packages” 🡪 “install packages” to install them).

1. **INTRODUCTION TO R DOCUMENTS**

There are many different sources that you may learn R from. To start with “cran.r-project.org” web page offers many documents. For a comprehensive note see the document at below link

[**https://cran.r-project.org/doc/manuals/R-intro.pdf**](https://cran.r-project.org/doc/manuals/R-intro.pdf)

For step-by-step instructions and hands-on experience, please complete the exercises offered at **tryr.codeschool.com/**

Or for a free R course just register at below link and follow their classes

[**https://www.coursera.org/learn/r-programming**](https://www.coursera.org/learn/r-programming)

1. **CODES IN COLOR**

Codes are written in R-editor. R-editor can be called by pressing “File – New Script” or alternatively “File – Open Script” for existing codes. However, R-editor does not have different colors for functions, variables, etc. Some experienced programmers may prefer such a colorful editor because it helps debugging the code. Alternative to R-editor, Notepad++ can be used as an editor as well. To form this link between Notepad++ and R, an additional software called “Npp2R” is used. In this way the codes written in Notepad++ can be run directly over Notepad++ and the comments will pass to R. Notepad++ and Npp2R are third party software (are not products of the group introduces R software).

Notepad++ can be downloaded from below link

<https://notepad-plus-plus.org/repository/6.x/6.9.2/npp.6.9.2.Installer.exe>

NppR2 can be downloaded from below link

<https://sourceforge.net/projects/npptor/files/latest/download>

Once again, Notepad++ and Npp2R are optional, if you would like you might as well use R-editor to write your codes.

1. **FIRST CODES**

To run the commends line-by-line, you may write above comments to **Notepad++** and hit **F8** function or alternatively write in **R-editor** and hit **ctrl+r.** Alternatively, long codes could be written all together by pressing **ctrl+F8** in **Notepad++** or “**Edit – Run All**” in **R-Editor**.

Many programming language introductions start with below comment, so we will do the same. Just type “hello world” to the console.

**"hello world"**

**[1] "hello world"**

You may assign this text to a variable as well and then print it to the screen in two different ways:

**varname = "hello world"**

**varname**

**[1] "hello world"**

**print(varname)**

**[1] "hello world"**

Now, let’s try some basic math expressions (+ summation and \* multiplication operators). Just type

**1+1**

**[1] 2**

**5\*9**

**[1] 45**

Next, lets’ try some logical expressions (TRUE / FALSE)

**5 > 9**

**[1] FALSE**

**a = 20**

**a > -3**

**[1] TRUE**

**5\*6 == 40**

**[1] FALSE**

**(1+2) == 3**

**[1] TRUE**

**T == TRUE**

**[1] TRUE**

Next, lets’ try some basic functions

**x = c(1,2,3,4,5,6)**

**y = c(2,4,6,8,12,10)**

**mean(x)**

**[1] 3.5**

**sum(x)**

**[1] 21**

**sqrt(x)**

**[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490**

**sd(x)**

**[1] 1.870829**

**var(x)**

**[1] 3.5**

**cov(x,y)**

**[1] 6.6**

**cor(x,y)**

**[1] 0.9428571**

**rep(10, times=5)**

**[1] 10 10 10 10 10**

**c(1:10)**

**[1] 1 2 3 4 5 6 7 8 9 10**

1. **HELP MENU**

How functions are used may not be immediately clear in many cases. To see what kind of parameters are used, or to see the syntax of these functions type

**?var**

**starting httpd help server ... done**

The pop-up web page will show 1) Description, 2) Arguments, 3) Details, 4) Value, 5) Note, 6) References, 7) See Also, 8) Examples

1. **CONSOLE**

If there are too many items in your console, you may clean it by first clicking the R console and then hitting **ctrl+L**

To see which variables are stored in your R type

**ls()**

**[1] "a" "varname" "x" "y"**

1. **RUNNING A CODE**

Sometimes the codes are very long, or you may need to call other scripts written in your computer. To run any other code written in your computer type

**codeloc1 = “C:/Users/user/Desktop/CE204/test.R”**

**codeloc2 = “C:\\Users\\user\\Desktop\\CE204\\test.R”**

**source(codeloc1)**

**source(codeloc2)**

Here the only thing you need to pay attention is the direction of slashes. In Windows the folder locations are saved using common slash (\) while R (similar to Mac and Linux systems) use backslash (/) to distinguish the folder locations. Briefly, to locate the files in computer in R, either use backslash (/) or double common slash (\\) between the folders.

1. **READING A FILE FROM COMPUTER**

When a task is assigned to you (e.g., analyzing some data), you often need to read the data from a file stored in your computer. Before you may start the analysis, the first thing you need to do is to “open” and read it to your computer.

As an example, 2015-2016 Spring semester CE204 Midterm Exam I and Exam II results are used.

**inname1 = "C:/Users/user/Desktop/CE204/data/R-MID-01.txt"**

**inname2 = "C:/Users/user/Desktop/CE204/data/R-MID-02.txt"**

**mid01 = read.table(inname1, skip=1)**

**mid02 = read.table(inname2, skip=1)**

1. **CALCULATING STATISTICS**

**GRADE01 = mid01[,2:4]**

**GRADE02 = mid02[,2:5]**

**TOTAL01 = rowSums( GRADE01 )**

**TOTAL02 = rowSums( GRADE02 )**

**mean(TOTAL01)**

**mean(TOTAL01, na.rm=T)**

**mean(TOTAL02, na.rm=T)**

**cor(TOTAL01,TOTAL02)**

**cor(TOTAL01,TOTAL02, use="na.or.complete")**

**hist(TOTAL01)**

**br = (0:110)**

**hist(TOTAL01, breaks=br, xlim=c(0,110))**

**br = (0:22)\*5**

**hist(TOTAL01, breaks=br, xlim=c(0,110))**

**t.test(TOTAL01 , TOTAL02)**

1. **PLOTTING**

**plot(TOTAL01, TOTAL02)**

**plot(TOTAL01, TOTAL02, xlim=c(35,100))**

**plot(TOTAL01, TOTAL02, xlim=c(35,100), col='red')**

**plot(TOTAL01, TOTAL02, xlim=c(35,100), col='red', xlab='Mid-I', ylab='Mid-II')**

**plot(TOTAL01, TOTAL02, xlim=c(35,100), col='red', xlab='Mid-I', ylab='Mid-II', col='red', lwd=2.0)**

**plot (TOTAL01[ 1:50], TOTAL02[1:50], pch=1, col='red', lwd=2.0, xlim=c(35,105), ylim=c(10,100), xlab='Mid-I', ylab='Mid-II')**

**points(TOTAL01[51:100], TOTAL02[51:100], pch=2, col='green', lwd=2.0)**

**points(TOTAL01[101:150], TOTAL02[101:150], pch=3, col='blue', lwd=2.0)**

**points(TOTAL01[151:200], TOTAL02[151:200], pch=4, col='cyan', lwd=2.0)**

**points(TOTAL01[201:241], TOTAL02[201:241], pch=5, col='black', lwd=2.0)**

**You may save the plot area either**

**- using “File – Save as”**

**- writing a code**

**outname = "C:/Users/user/Desktop/CE204/data/R-MID-01.jpg"**

**jpeg(filename=outname,units="cm",res =1024, width = 20, height = 12, quality = 1000)**

**op <- par(mar=c(5,5,1,1), oma=c(0,0,0,0), cex.lab = 1.4, cex.axis = 1.4) ###### [B,L,T,R]**

**plot (TOTAL01[ 1:50], TOTAL02[1:50], pch=1, col='red', lwd=2.0, xlim=c(35,105), ylim=c(10,100), xlab='Mid-I', ylab='Mid-II')**

**points(TOTAL01[51:100], TOTAL02[51:100], pch=2, col='green', lwd=2.0)**

**points(TOTAL01[101:150], TOTAL02[101:150], pch=3, col='blue', lwd=2.0)**

**points(TOTAL01[151:200], TOTAL02[151:200], pch=4, col='cyan', lwd=2.0)**

**points(TOTAL01[201:241], TOTAL02[201:241], pch=5, col='black', lwd=2.0)**

**par(op)**

**dev.off()**

1. **RUNNING CODES WITHIN OTHER CODES**

**codename = "C:/Users/user/Desktop/CE204/data/R-MID01-PLOT.R"**

**source(codename)**

1. **LIBRARIES (PACKAGE or TOOLBOXES)**

R has thousands (~10,000) packages freely available. These packages should be installed only once, but they should be called each time they will be used:

To install, click “Packages” at R Console, and then click “Install Packages”, then select CRAN mirror (any location is fine, I usually select the default “0-Cloud”), click OK.

As an example, let’s install “**fields**” package. Select “fields” from the list, and then click OK. This step is not necessary each time you run R, if you install this package once that is enough, you don’t need to install again. But you need to call it each time you will use this package in R. You may call it via:

**library(fields) # required for image.plot**

**rd = rnorm(10000,24,4)**

**rd = 1:10000**

**dim(rd) = c(100,100)**

**image(rd, col=rainbow(500))**

**image(rd, col=heat.colors(500))**

**image(rd, col=terrain.colors(500))**

**image.plot(rd, col=terrain.colors(500))**

**text(0.5,0.6,'ANY TEXT IS OK', col='blue', cex=2.0, lwd=2.0)**

1. **WRITING FUNCTIONS INSIDE R**

You may write your own function, if your code requires certain calculations to be performed very frequently. Below example writes a function to calculate standard deviation of a data.

**mystdev <- function(indata){**

**n = length(indata)**

**stdev = sqrt( sum( (indata - mean(indata))^2 )/(n-1) )**

**return(stdev)**

**}**

**randat = rnorm(100,10,15)**

**mystdev(randat)**

**sd(randat)**

1. **POWER OF GOOGLE**

**REMEMBER, IF YOU CAN NOT FIGURE OUT SOMETHING, JUST GOOGLE IT!!! I AM CONFIDENT THAT SIMILAR QUESTIONS HAVE BEEN ASKED AND ANSWERRED BEFORE…**

**DON’T FORGET TO PRACTICE PROGRAMMING!!**

**THIS SKILL CANNOT BE GAINED IN ANY OTHER WAY..**