Foundations of R Software

Lecture 2

Introduction

::::

Help, Demonstration, and Examples in R

Shalabh

Department of Mathematics and Statistics

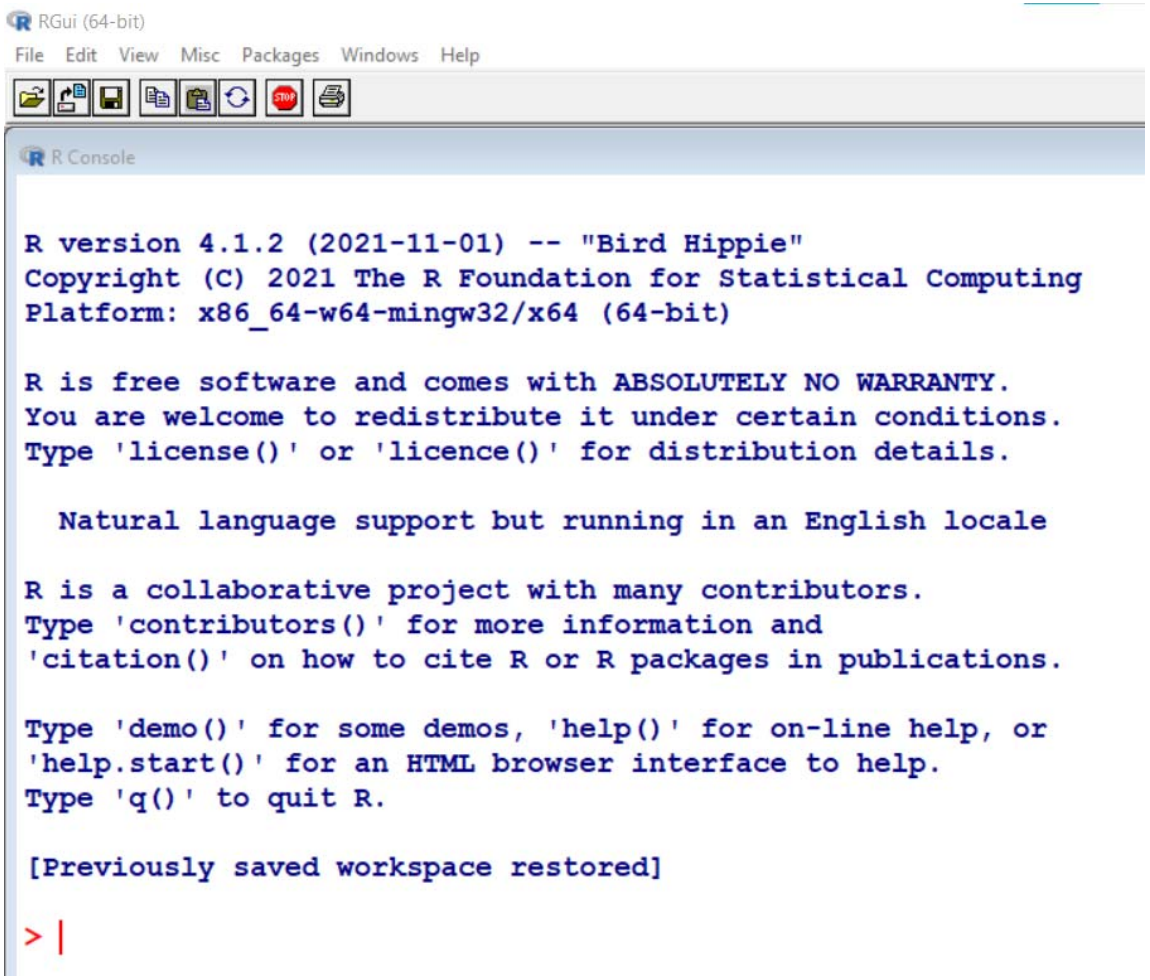
Indian Institute of Technology Kanpur

Starting with R

To start R, double click on the icon

.

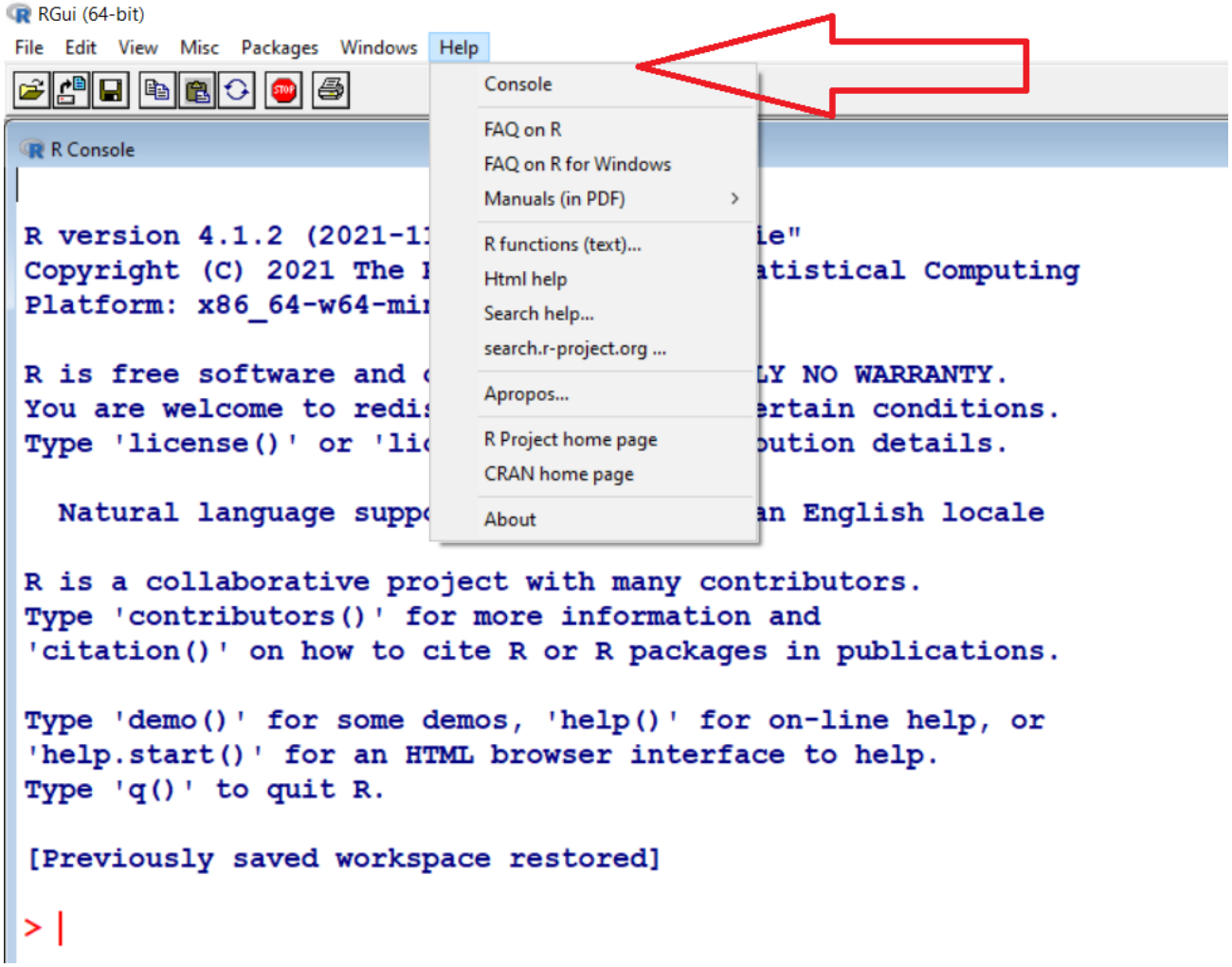
Then we get the following Gui (Graphic user interface) window screen



Getting Help in R

This can be done in one of the following ways:

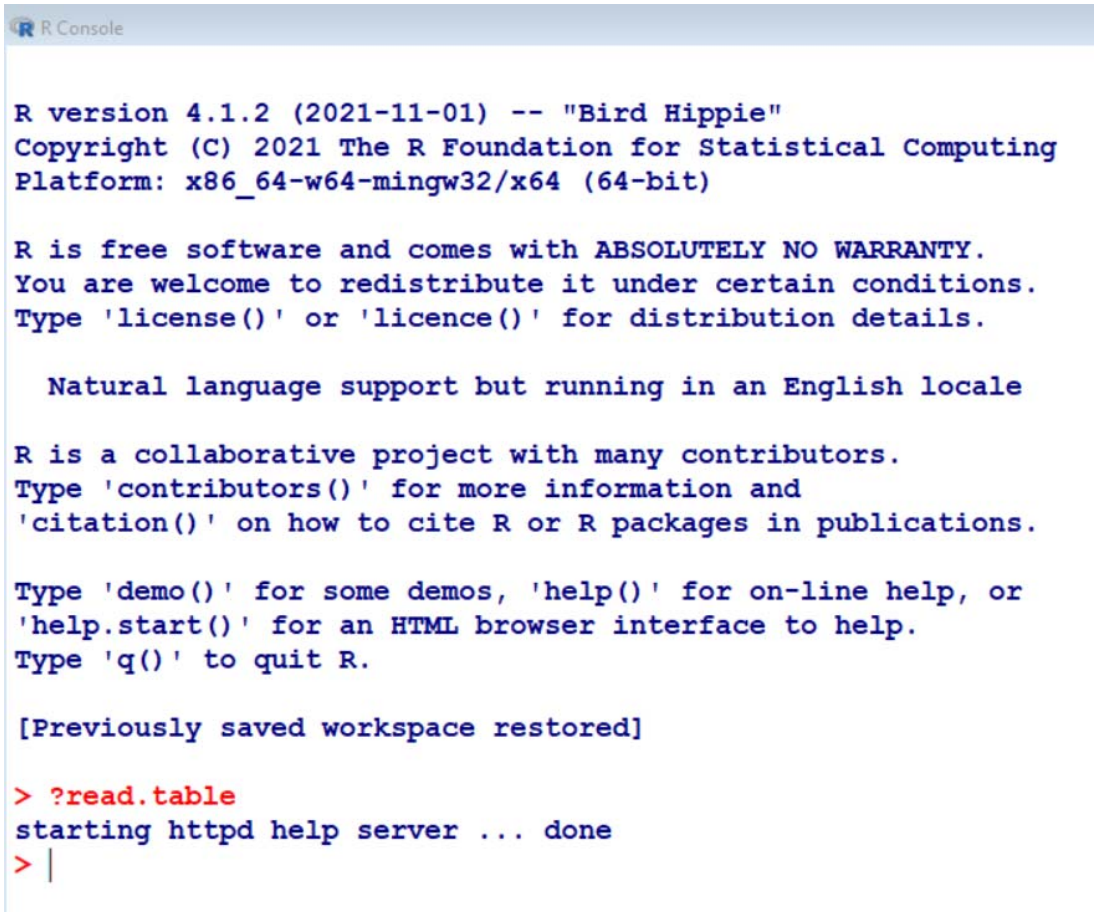
1) **Start R software and click the help button in the toolbar of the R Gui (Graphic user interface) window.**



Getting Help in R

2. Search for help in Google `www.google.com`

3. If you need help with a function, then type question mark followed by the name of the function. For example, `?read.table` to get help for function `read.table`.



read.table {utils}

Data Input

Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", row.names, col.names,
           as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)

read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

Arguments

…continued

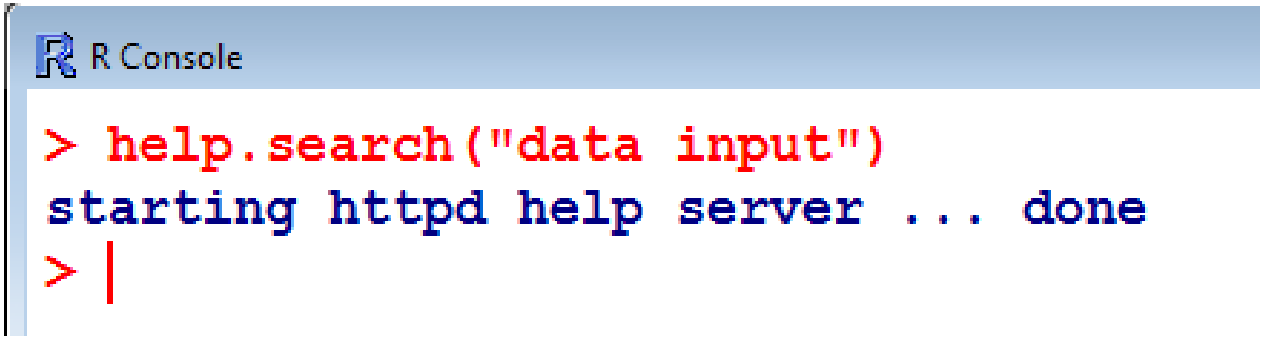
Arguments

file	the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an *absolute* path, the file name is *relative* to the current working directory, `getwd()`. Tilde-expansion is performed where supported. This can be a compressed file (see `file`). Alternatively, `file` can be a readable text-mode connection (which will be opened for reading if necessary, and if so `close`d (and hence destroyed) at the end of the function call). (If `stdin()` is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, `Ctrl-D` on Unix and `Ctrl-Z` on Windows. Any pushback on `stdin()` will be cleared before return.) `file` can also be a complete URL. (For the supported URL schemes, see the 'URLs' section of the help for `url`.)
header	a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: `header` is set to `TRUE` if and only if the first row contains one fewer field than the number of columns.
sep	the field separator character. Values on each line of the file are separated by this character. If `sep = ""` (the default for `read.table`) the separator is 'white space', that is one or more spaces, tabs, newlines or carriage returns.
quote	the set of quoting characters. To disable quoting altogether, use `quote = ""`. See `scan` for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless `colClasses` is specified.
dec	the character used in the file for decimal points.
row.names	a vector of row names. This can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names. If there is a header and the first row contains one fewer field than the number of columns, the first column in the input is used for the row names. Otherwise if `row.names` is missing, the rows are numbered. Using `row.names = NULL` forces row numbering. Missing or `NULL` `row.names` generate row names that are considered to be 'automatic' (and not preserved by `as.matrix`).
col.names	a vector of optional names for the variables. The default is to use `"V"` followed by the column number.
as.is	the default behavior of `read.table` is to convert character variables (which are not converted to logical, numeric or complex) to factors. The variable `as.is` controls the conversion of columns not otherwise specified by `colClasses`. Its value is either a vector of logicals (values are recycled if necessary), or a vector of numeric or character indices which specify which columns should not be converted to factors. Note: to suppress all conversions including those of numeric columns, set `colClasses = "character"`.

All minor details and explanations of all arguments are given.

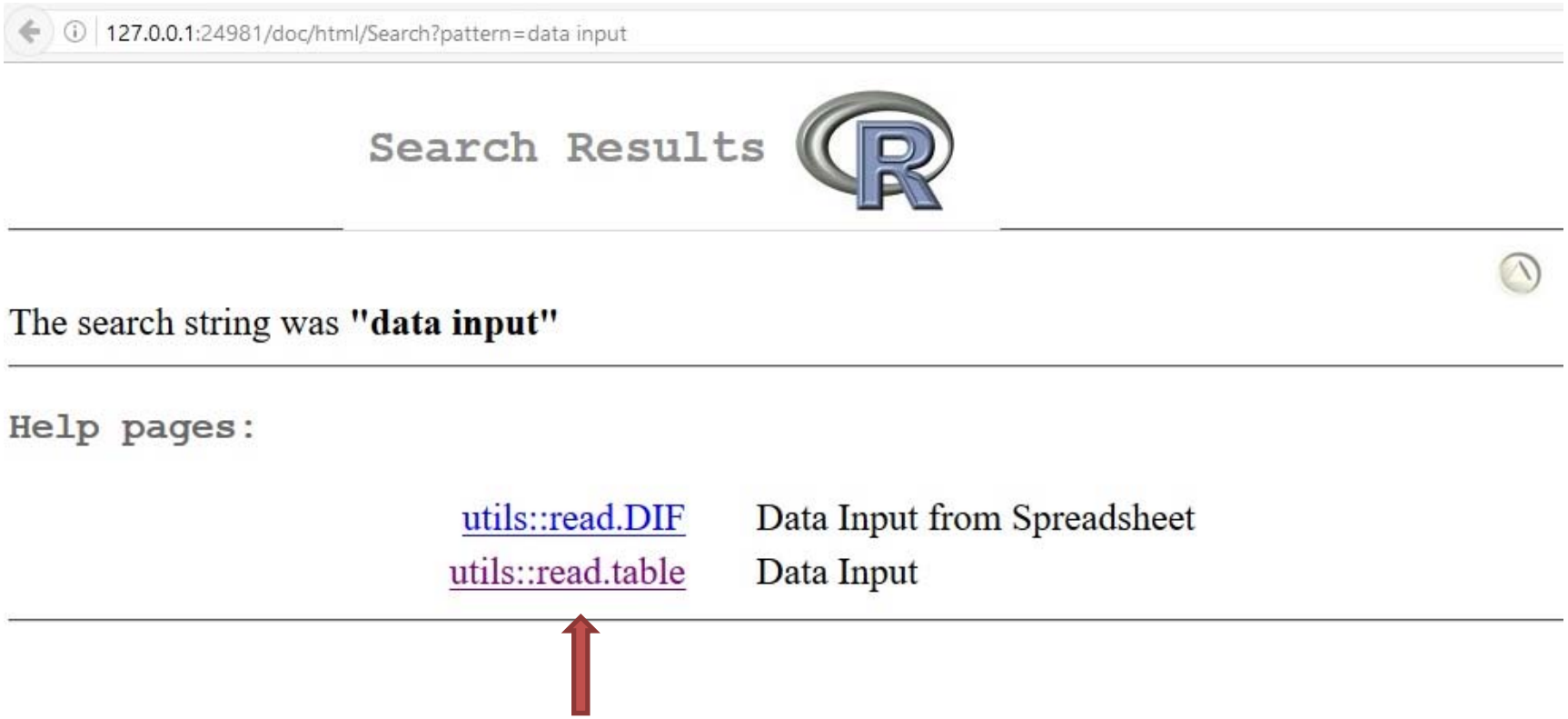
Getting Help in R

4. Sometimes, you want to search by the subject on which we want help (e.g. data input). In such a case, type `help.search("data input")`



Then we get….

Then we get….

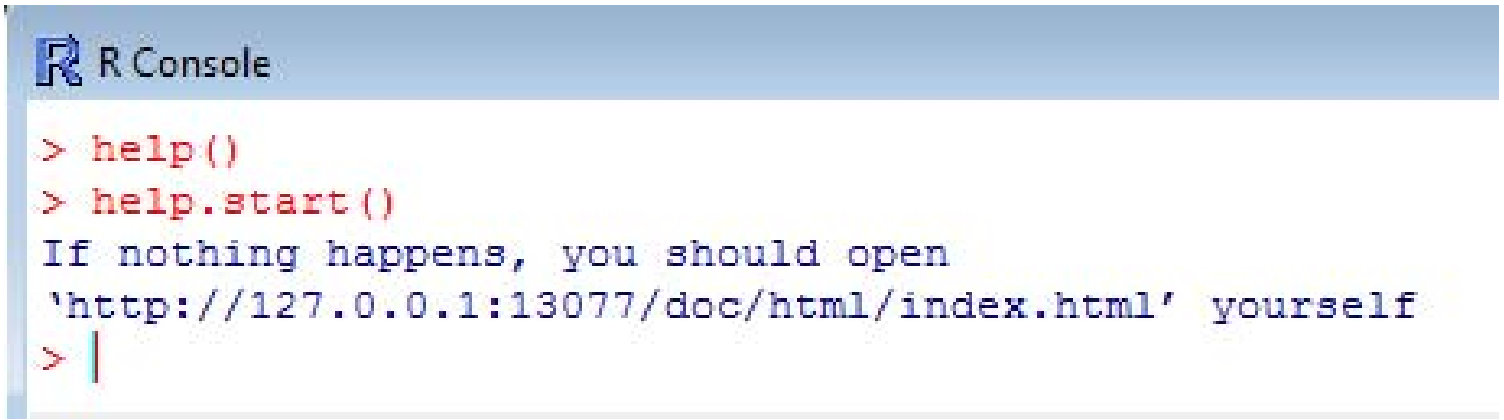


Clicking over the link give required information

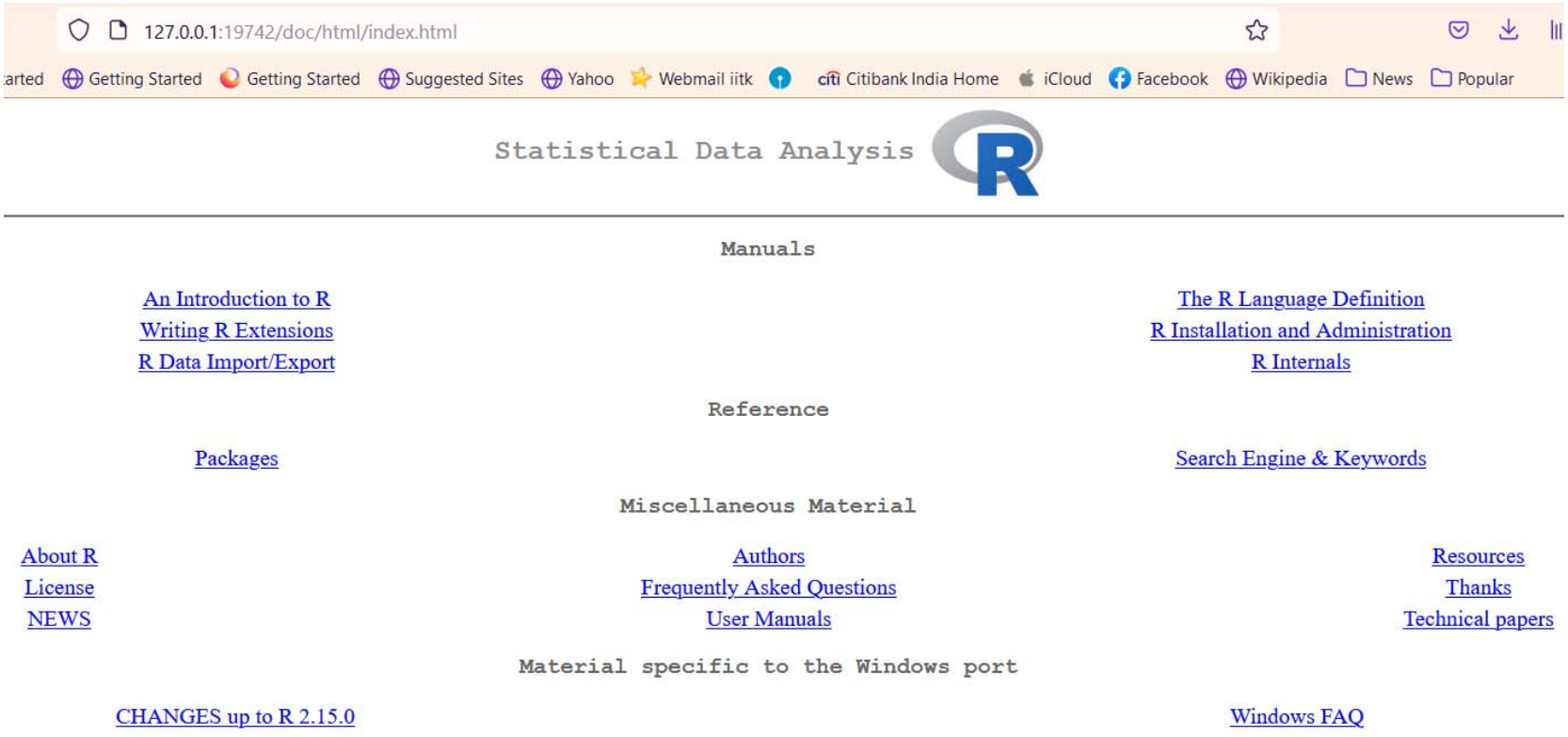
Getting Help in R

4. '`help()`' for on-line help,

or '`help.start()`' for an HTML browser interface to help.



Then we get....


127.0.0.1:19742/doc/html/index.html
arted
Getting Started
Getting Started
Suggested Sites
Yahoo
Webmail iitk
Citibank India Home
iCloud
Facebook
Wikipedia
News
Popular
Statistical Data Analysis
Manuals
An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals
Reference
Packages
Search Engine & Keywords
Miscellaneous Material
About R
License
NEWS
Authors
Frequently Asked Questions
User Manuals
Resources
Thanks
Technical papers
Material specific to the Windows port
CHANGES up to R 2.15.0
Windows FAQ

Getting Help in R

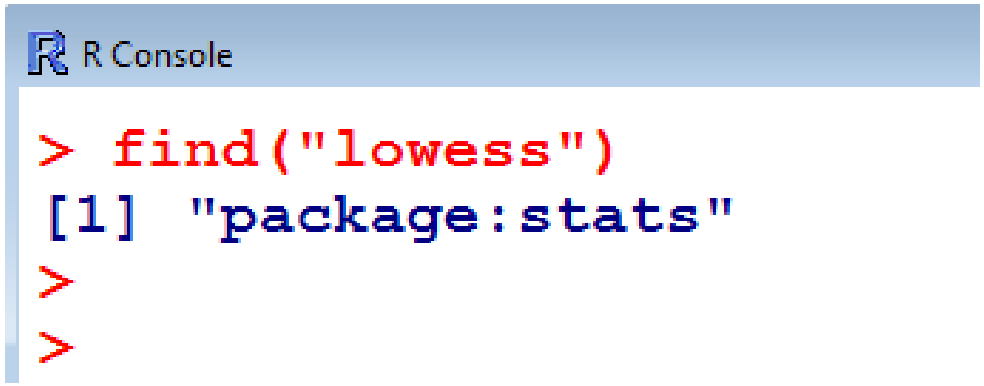
5) Other useful functions are `find` and `apropos`.

6) The `find` function tells us what package something is in.

For example

`> find("lowess")` returns

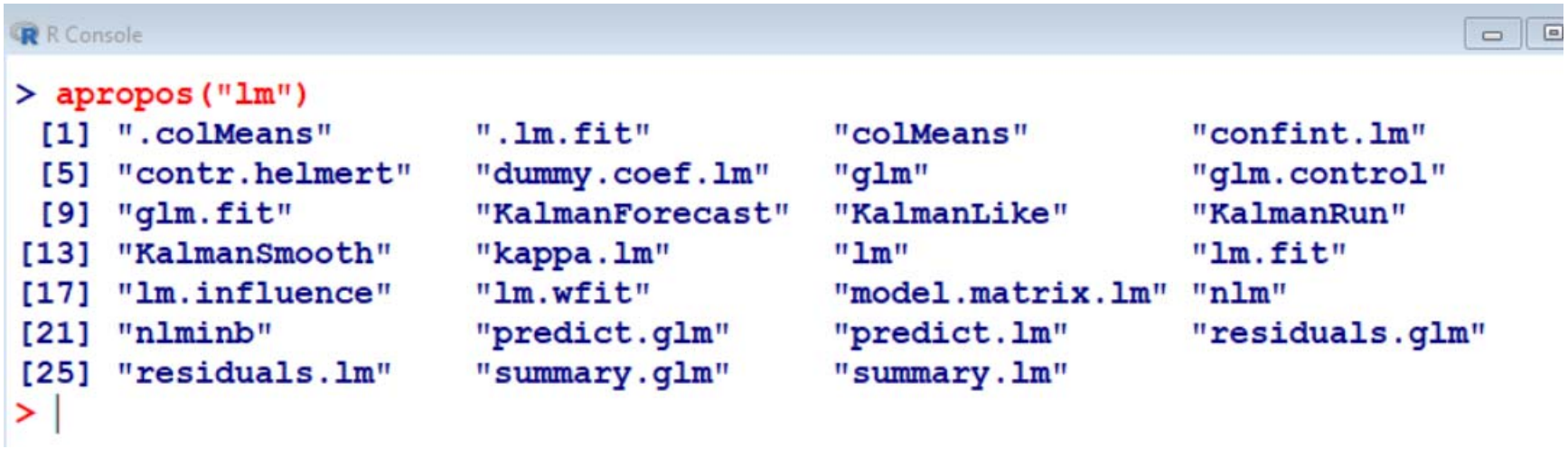
```
[1] "package:stats"
```



Getting Help in R

7) The apropos returns a character vector giving the names of all objects in the search list that match your enquiry.

`apropos("lm")` **returns**



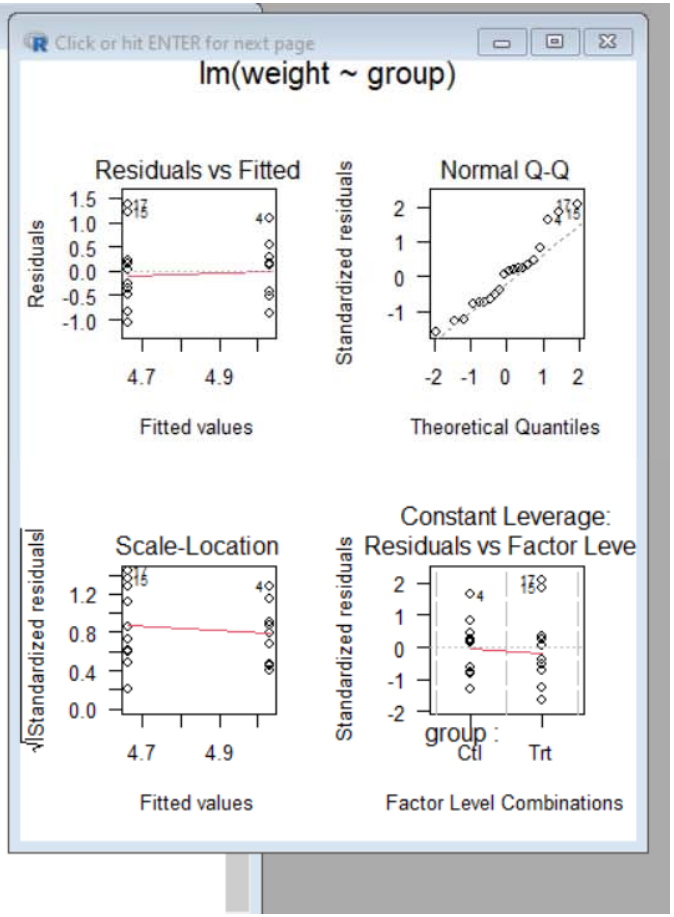
Worked Examples of Functions

To see a worked `example` just type the function name, e.g., `lm` for linear models:

```
example(lm)
```

and we see the printed and graphical output produced by the `lm` function.

```
Call:
lm(formula = weight ~ group - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0710 -0.4938  0.0685  0.2462  1.3690

Coefficients:
         Estimate Std. Error t value Pr(>|t|)
groupCtl   5.0320     0.2202   22.85 9.55e-15 ***
groupTrt   4.6610     0.2202   21.16 3.62e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6964 on 18 degrees of freedom
Multiple R-squared:  0.9818,    Adjusted R-squared:  0.9798
F-statistic: 485.1 on 2 and 18 DF,  p-value: < 2.2e-16


lm> ## End(No test)
lm> opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))

lm> plot(lm.D9, las = 1)      # Residuals, Fitted, ...
Waiting to confirm page change...
```



…and other details follow further

Demonstration of R Functions

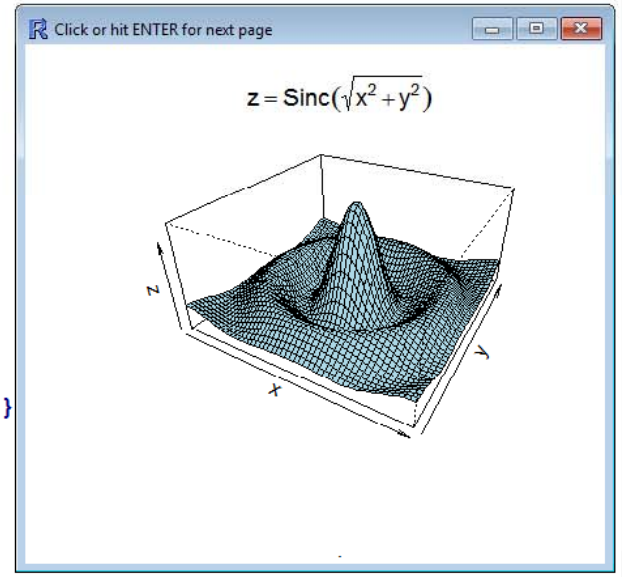
This can be useful for seeing the type of things that R can do.

`demo(persp)` [`persp` is a command for 3d surface plots]

R Console

```
> demo(persp)


        demo(persp)
        ---- ~~~~~

Type  <Return>   to start :

> ### Demos for  persp()  plots   -- things not in  example(persp)
> ### ------------------------
>
> require(datasets)

> require(grDevices); require(graphics)

> ## (1) The Obligatory Mathematical surface.
> ##     Rotated sinc function.
>
> x <- seq(-10, 10, length.out = 50)

> y <- x

> rotsinc <- function(x,y)
+ {
+     sinc <- function(x) { y <- sin(x)/x ; y[is.na(y)] <- 1; y }
+     10 * sinc( sqrt(x^2+y^2) )
+ }

> sinc.exp <- expression(z == Sinc(sqrt(x^2 + y^2)))

> z <- outer(x, y, rotsinc)

> oldpar <- par(bg = "white")

> persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
Waiting to confirm page change...
```

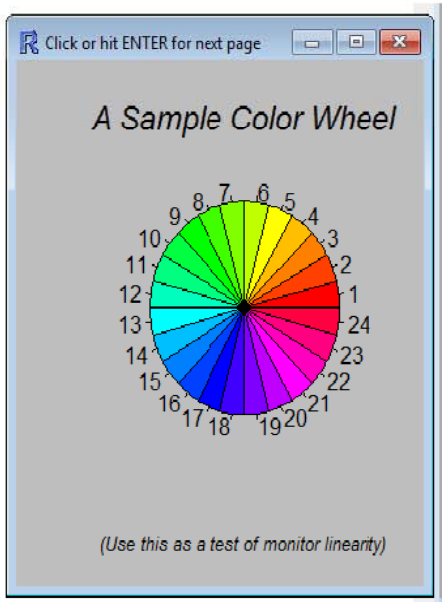
Click or hit ENTER for next page



…and it continues

Demonstration of R Functions

This can be useful for seeing the type of things that R can do.

demo(graphics)

```
R Console
> demo(graphics)

        demo(graphics)
        ---- ~~~~~~~~

Type  <Return>   to start :

> #  Copyright (C) 1997-2009 The R Core Team
>
> require(datasets)

> require(grDevices); require(graphics)

> ## A little color wheel.       This code just plots equally spaced hues in
> ## a pie chart.       If you have a cheap SVGA monitor (like me) you will
> ## probably find that numerically equispaced does not mean visually
> ## equispaced.  On my display at home, these colors tend to cluster at
> ## the RGB primaries.  On the other hand on the SGI Indy at work the
> ## effect is near perfect.
>
> par(bg = "gray")

> pie(rep(1,24), col = rainbow(24), radius = 0.9)
Waiting to confirm page change...

> title(main = "A Sample Color Wheel", cex.main = 1.4, font.main = 3)

> title(xlab = "(Use this as a test of monitor linearity)",
+       cex.lab = 0.8, font.lab = 3)

> ## We have already confessed to having these.  This is just showing off X11
> ## color names (and the example (from the postscript manual) is pretty "cute".
>
```