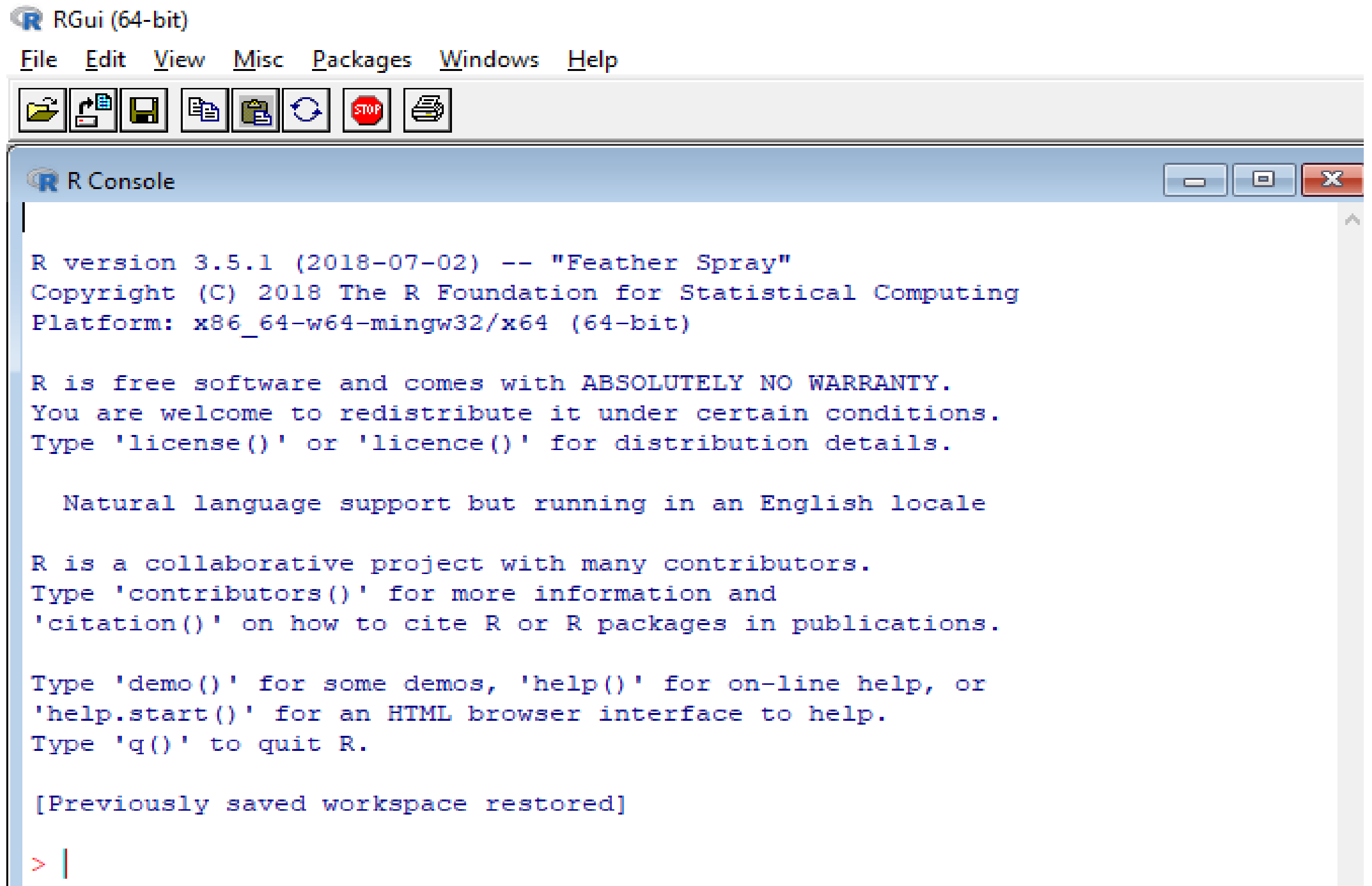
Introduction

Help, Demonstration, Packages and Libraries

Kanahaiya Kumar [Data Mining Central]

Starting with R

- To Start with R, double click on icon
- Then a GUI window getting displayed

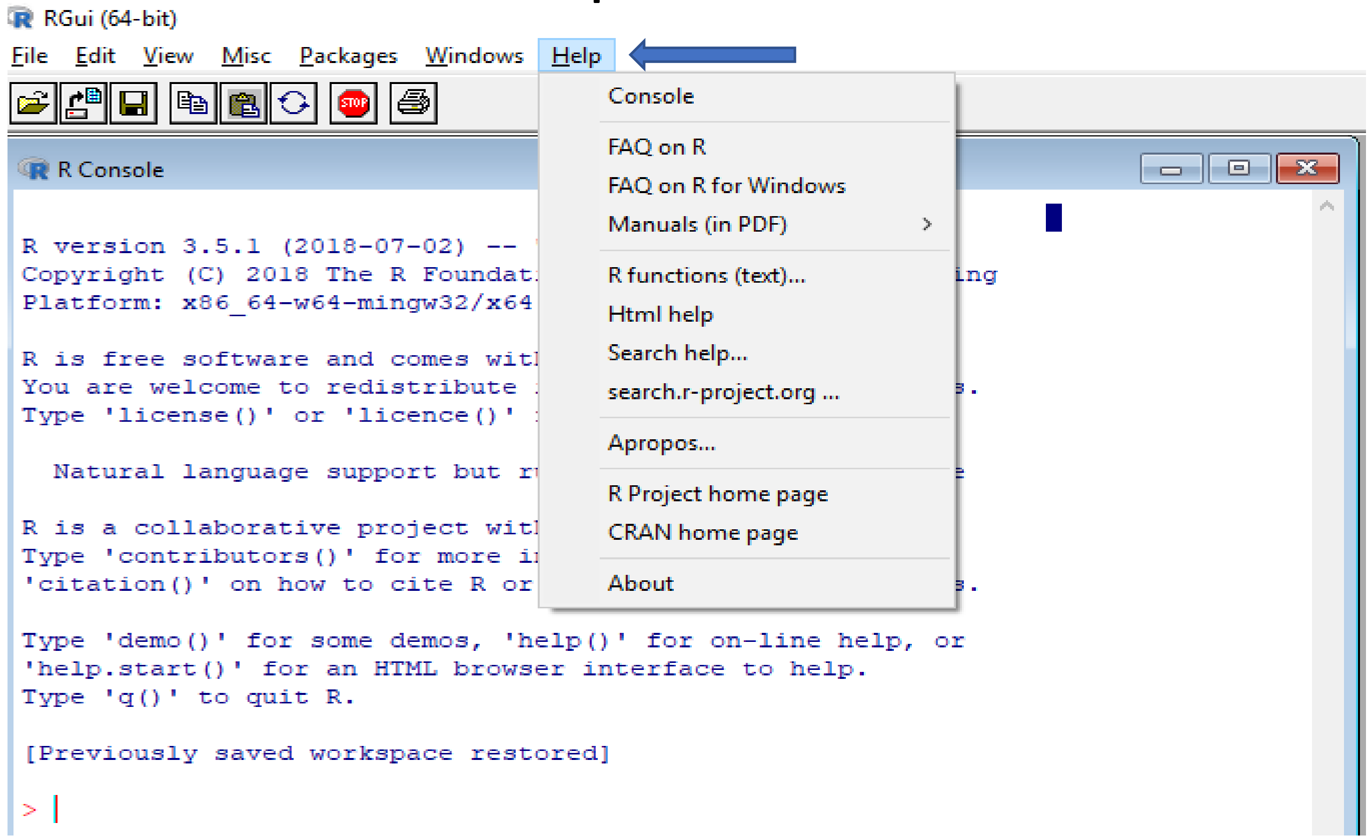


```
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

[Previously saved workspace restored]

>
```

Getting help in R

This can be done many ways

1) Start R Software and click the help button in the toolbar of the R GUI window



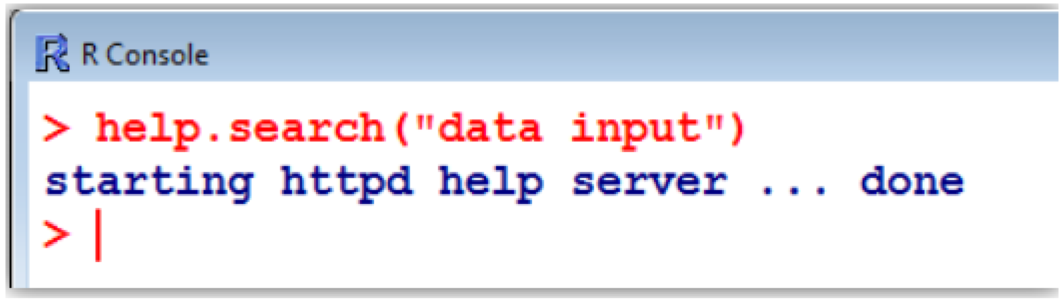
```
R version 3.5.1 (2018-07-02) --
Copyright (C) 2018 The R Foundat                                         ing
Platform: x86_64-w64-mingw32/x64

R is free software and comes wit
You are welcome to redistribute                                          s.
Type 'license()' or 'licence()'

  Natural language support but r                                         e

R is a collaborative project wit
Type 'contributors()' for more i
'citation()' on how to cite R or                                         s.

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.

[Previously saved workspace restored]

>
```

Getting Help in R

2) Search for help on 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 and get the all minor details and explanations of all arguments are given.

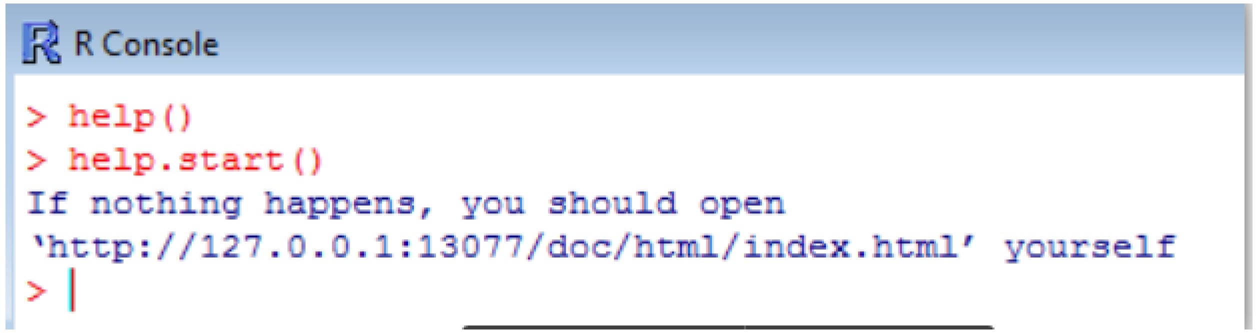
4) Sometimes, you want to search by the subject on which we want help (e.g. data input). In such case, type help.search("data input")



```
> help.search("data input")
starting httpd help server ... done
> |
```

Getting help in R

5) help() for online help, or help.start() for an HTML browser interface to help



```
> help()
> help.start()
If nothing happens, you should open
'http://127.0.0.1:13077/doc/html/index.html' yourself
> 
```

6) Other useful functions are find and apropos

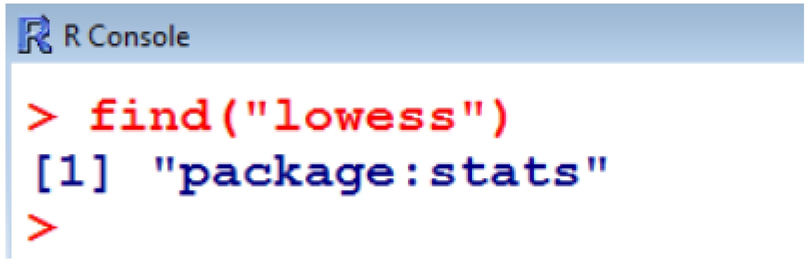
The find functions tell us what package something is in.

Getting help in R

For example,

> find(“lowess”) returns

[1] “package:stats”

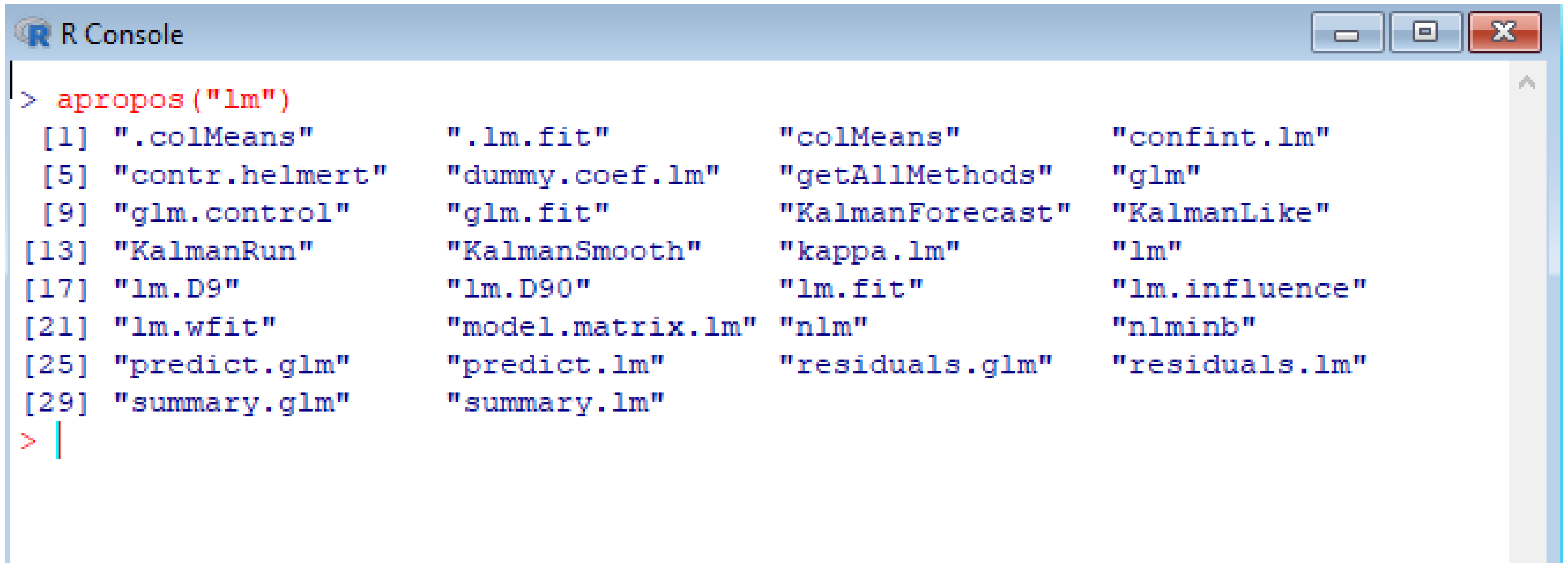


```
> find("lowess")
[1] "package:stats"
>
```

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



```
> apropos("lm")
 [1] ".colMeans"        ".lm.fit"          "colMeans"         "confint.lm"
 [5] "contr.helmert"    "dummy.coef.lm"    "getAllMethods"    "glm"
 [9] "glm.control"      "glm.fit"          "KalmanForecast"   "KalmanLike"
[13] "KalmanRun"        "KalmanSmooth"     "kappa.lm"         "lm"
[17] "lm.D9"            "lm.D90"           "lm.fit"           "lm.influence"
[21] "lm.wfit"          "model.matrix.lm"  "nlm"              "nlminb"
[25] "predict.glm"      "predict.lm"       "residuals.glm"    "residuals.lm"
[29] "summary.glm"      "summary.lm"
> 
```

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

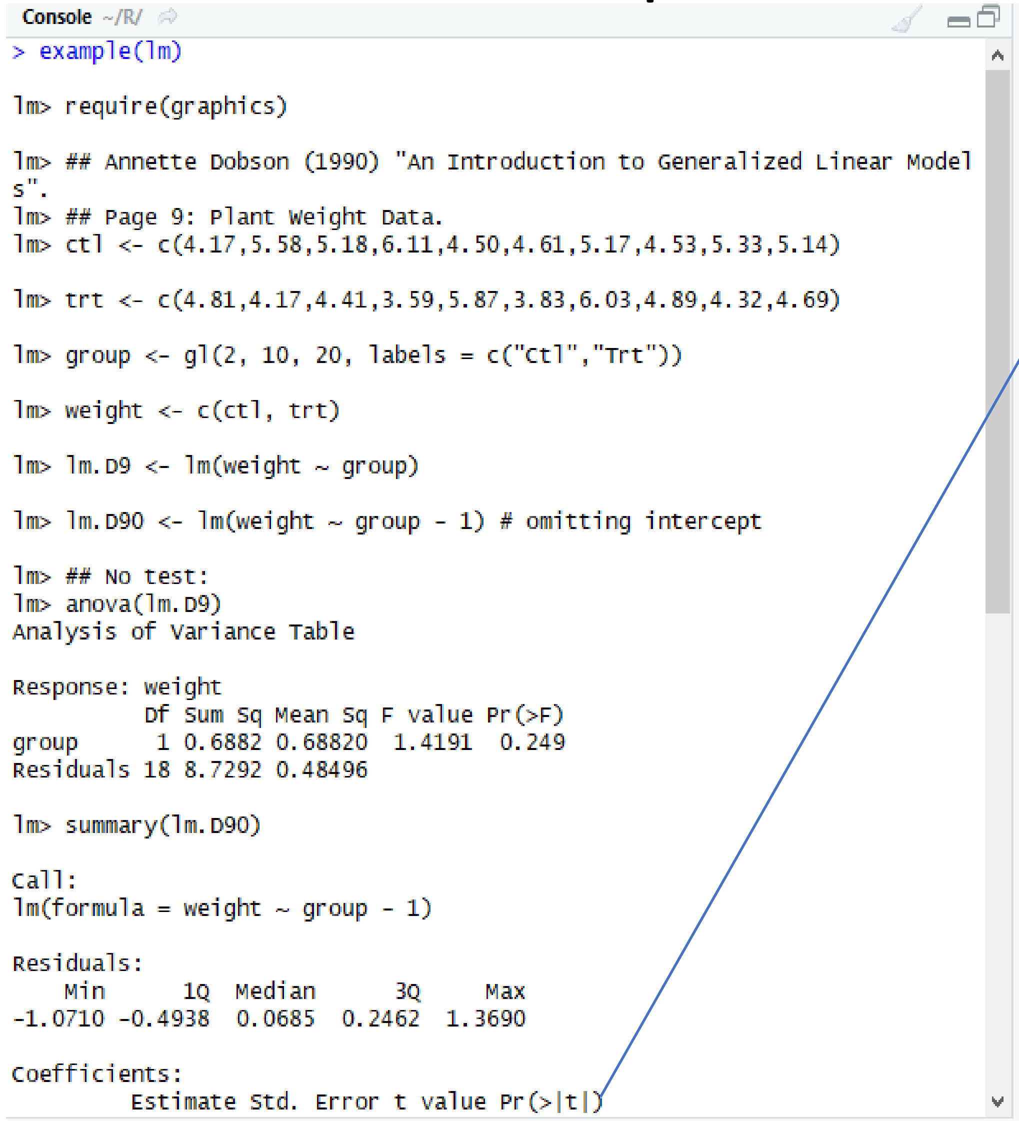
You may get error in R Studio like below while executing example(lm)

RStudio Error in plot.new() : figure margins too large

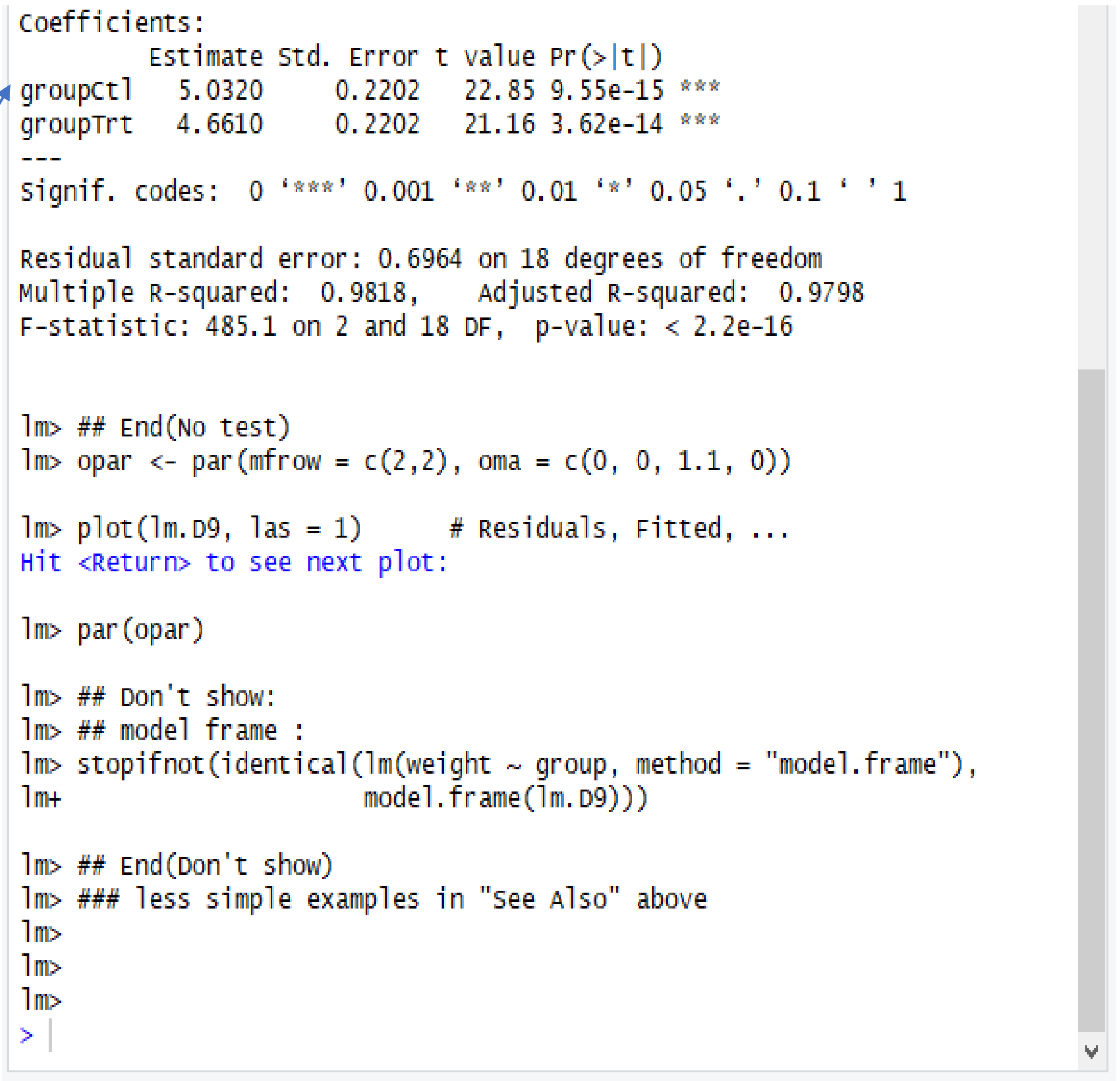
Run the below code then run example(lm)

```
graphics.off()
par(“mar”)
par(mar=c(1,1,1,1))
```

Worked examples of functions cont…

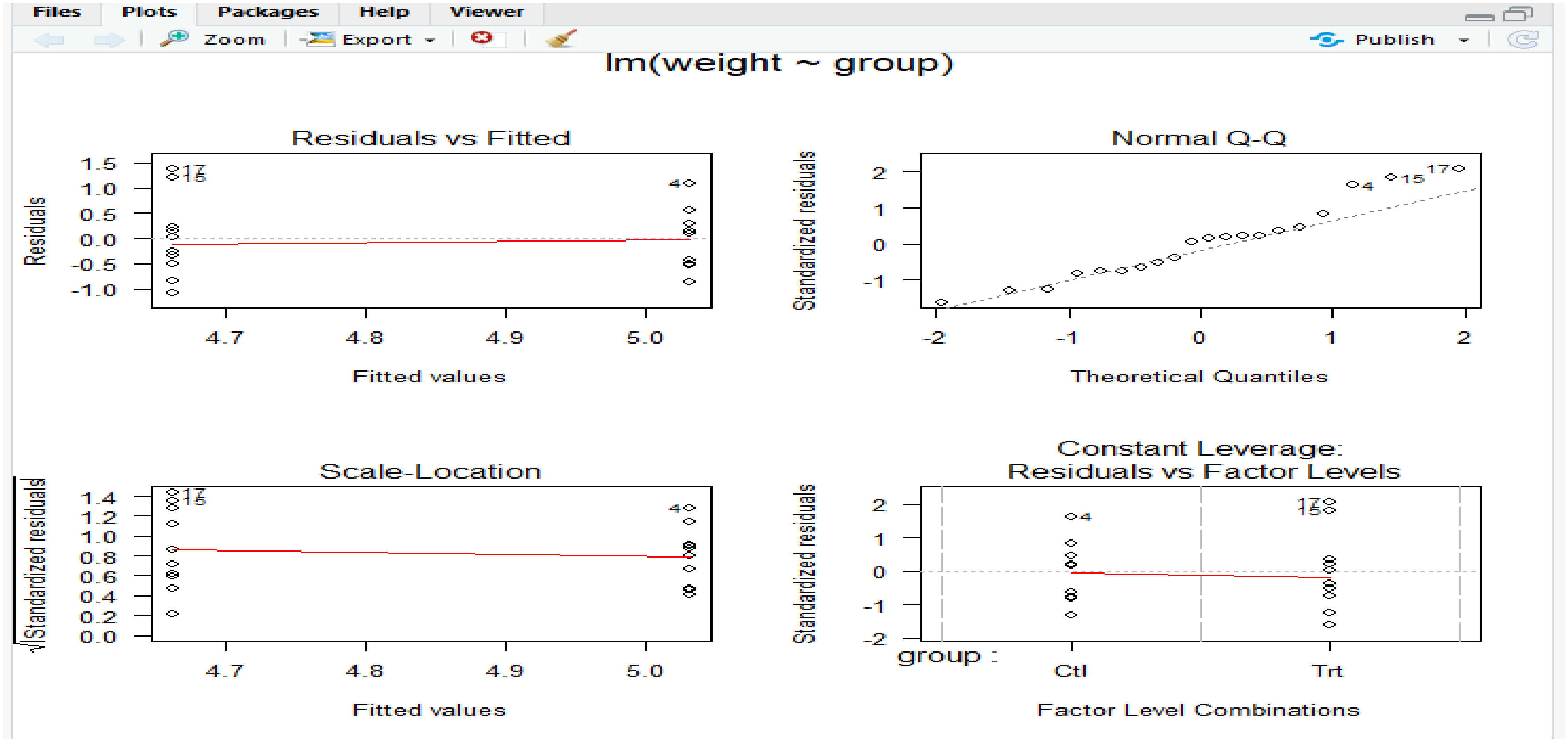


```
> example(lm)

lm> require(graphics)

lm> ## Annette Dobson (1990) "An Introduction to Generalized Linear Model
s".
lm> ## Page 9: Plant Weight Data.
lm> ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)

lm> trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)

lm> group <- gl(2, 10, 20, labels = c("Ctl","Trt"))

lm> weight <- c(ctl, trt)

lm> lm.D9 <- lm(weight ~ group)

lm> lm.D90 <- lm(weight ~ group - 1) # omitting intercept

lm> ## No test:
lm> anova(lm.D9)
Analysis of Variance Table

Response: weight
          Df Sum Sq Mean Sq F value Pr(>F)
group      1 0.6882 0.68820  1.4191  0.249
Residuals 18 8.7292 0.48496

lm> summary(lm.D90)

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|)
```



```
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, ...
Hit <Return> to see next plot:

lm> par(opar)

lm> ## Don't show:
lm> ## model frame :
lm> stopifnot(identical(lm(weight ~ group, method = "model.frame"),
lm+                     model.frame(lm.D9)))

lm> ## End(Don't show)
lm> ### less simple examples in "See Also" above
lm>
lm>
lm>
>
```

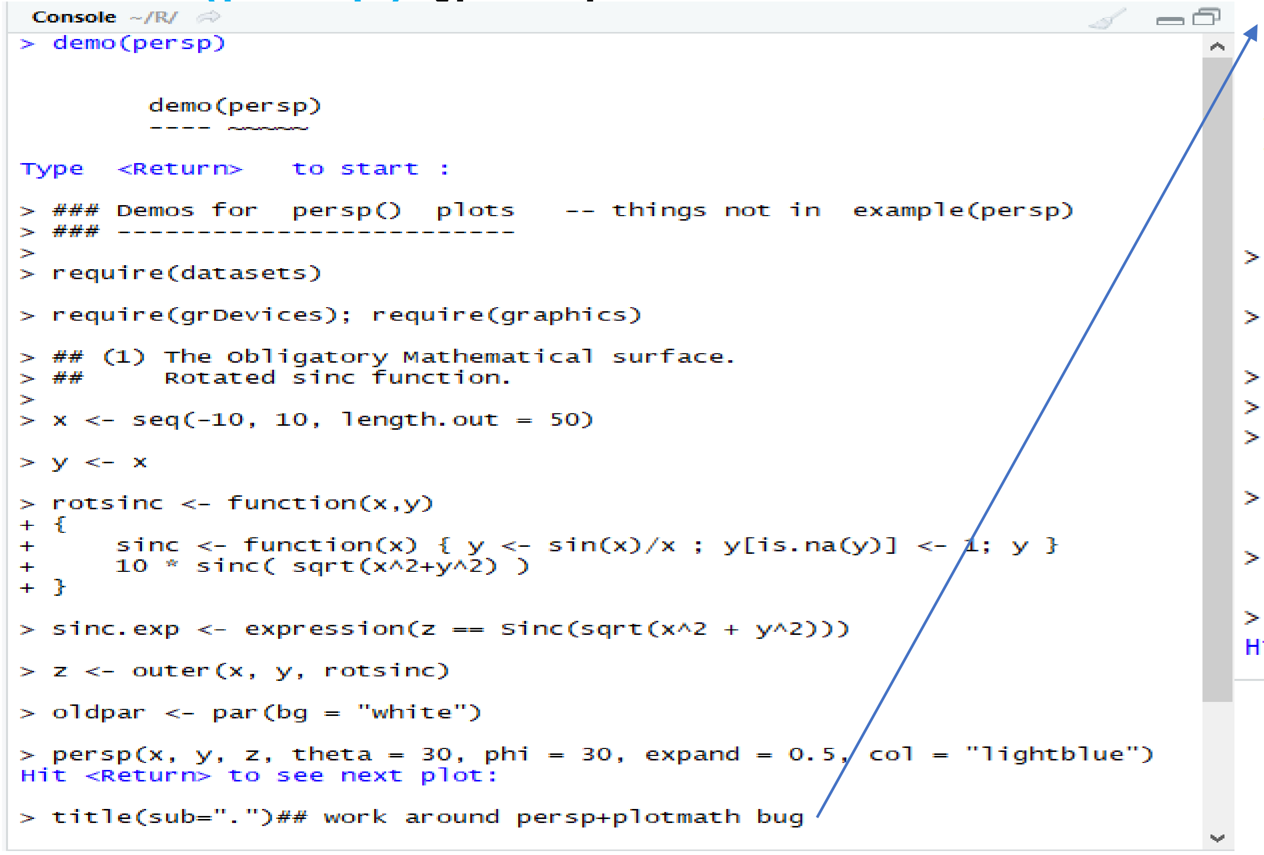
Worked examples of functions cont…



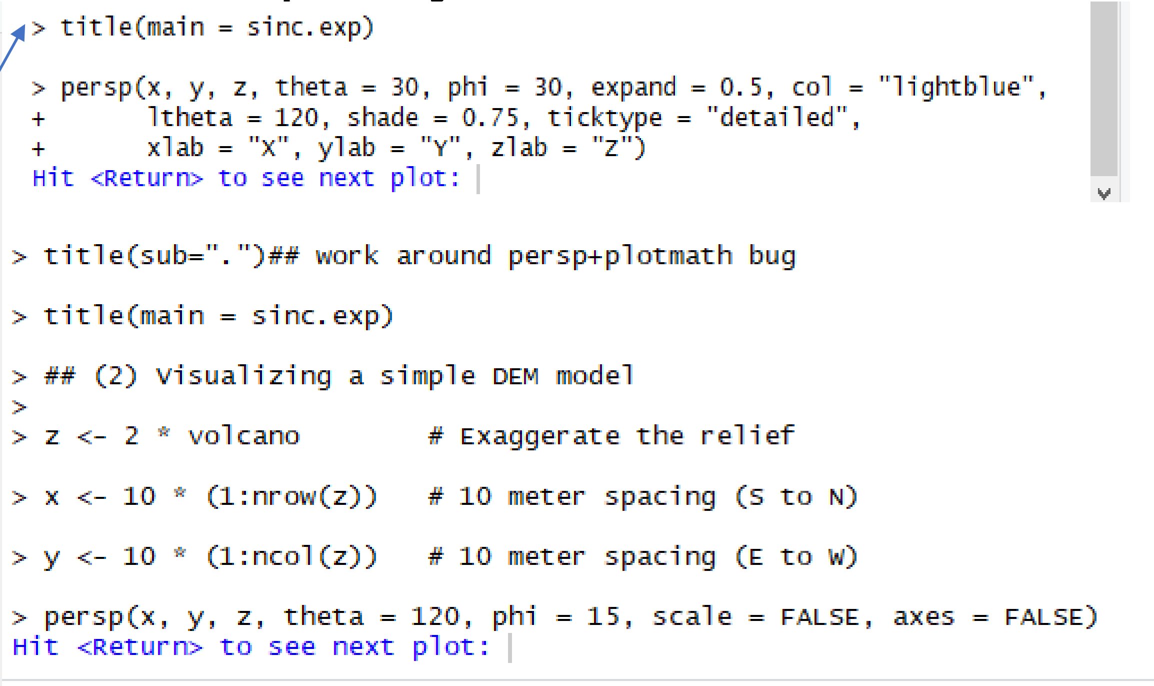
Demonstrations of R functions

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

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



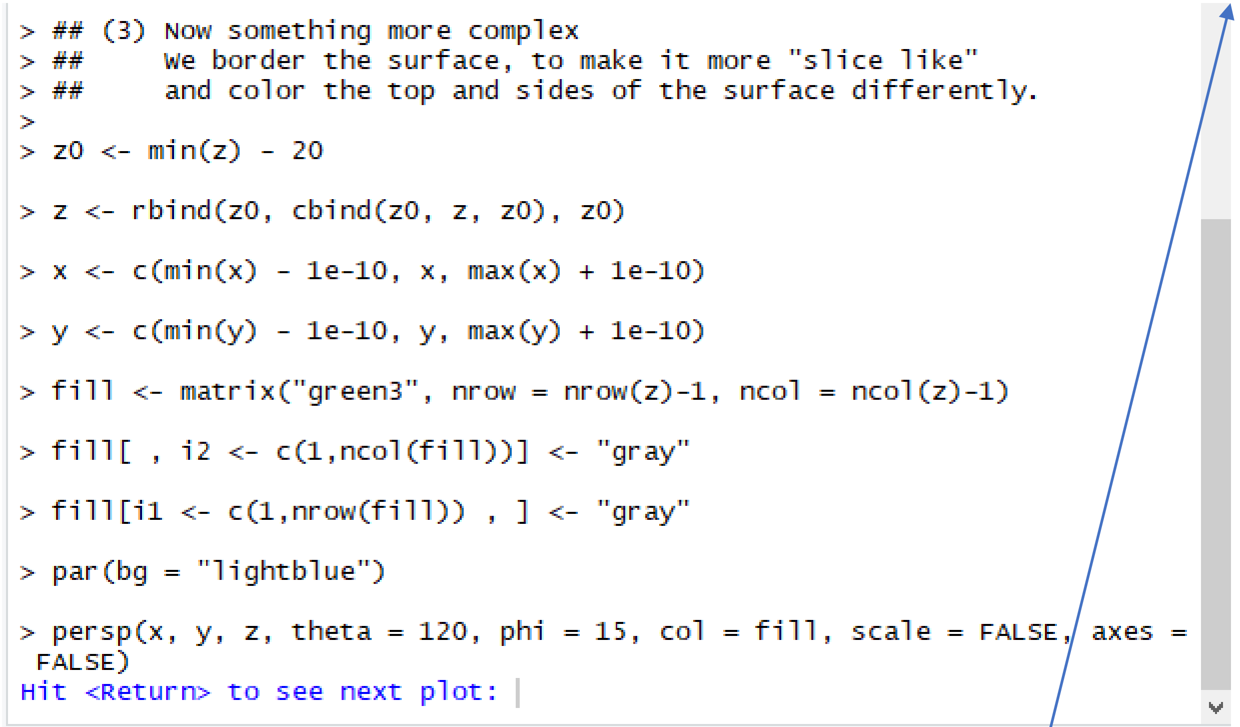
```
> demo(persp)


        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")
Hit <Return> to see next plot:

> title(sub=".")## work around persp+plotmath bug
```



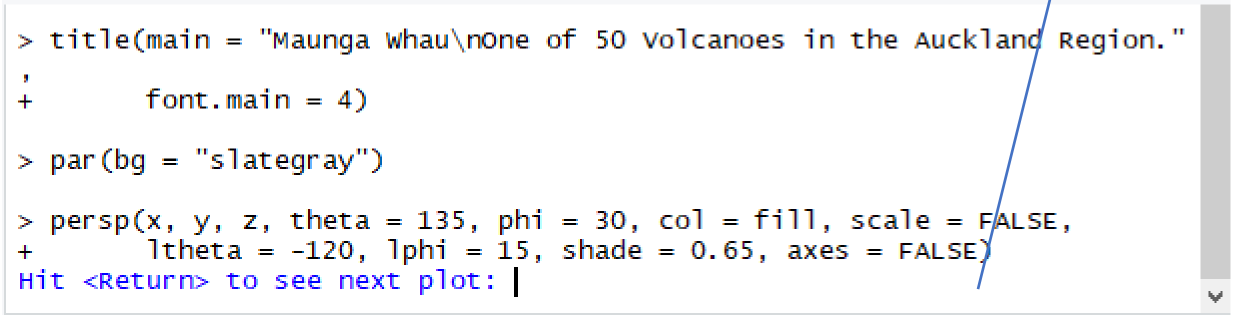
```
> title(main = sinc.exp)

> persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
+       ltheta = 120, shade = 0.75, ticktype = "detailed",
+       xlab = "X", ylab = "Y", zlab = "Z")
Hit <Return> to see next plot:
```

```
> title(sub=".")## work around persp+plotmath bug

> title(main = sinc.exp)

> ## (2) Visualizing a simple DEM model
>
> z <- 2 * volcano        # Exaggerate the relief

> x <- 10 * (1:nrow(z))   # 10 meter spacing (S to N)

> y <- 10 * (1:ncol(z))   # 10 meter spacing (E to W)

> persp(x, y, z, theta = 120, phi = 15, scale = FALSE, axes = FALSE)
Hit <Return> to see next plot:
```

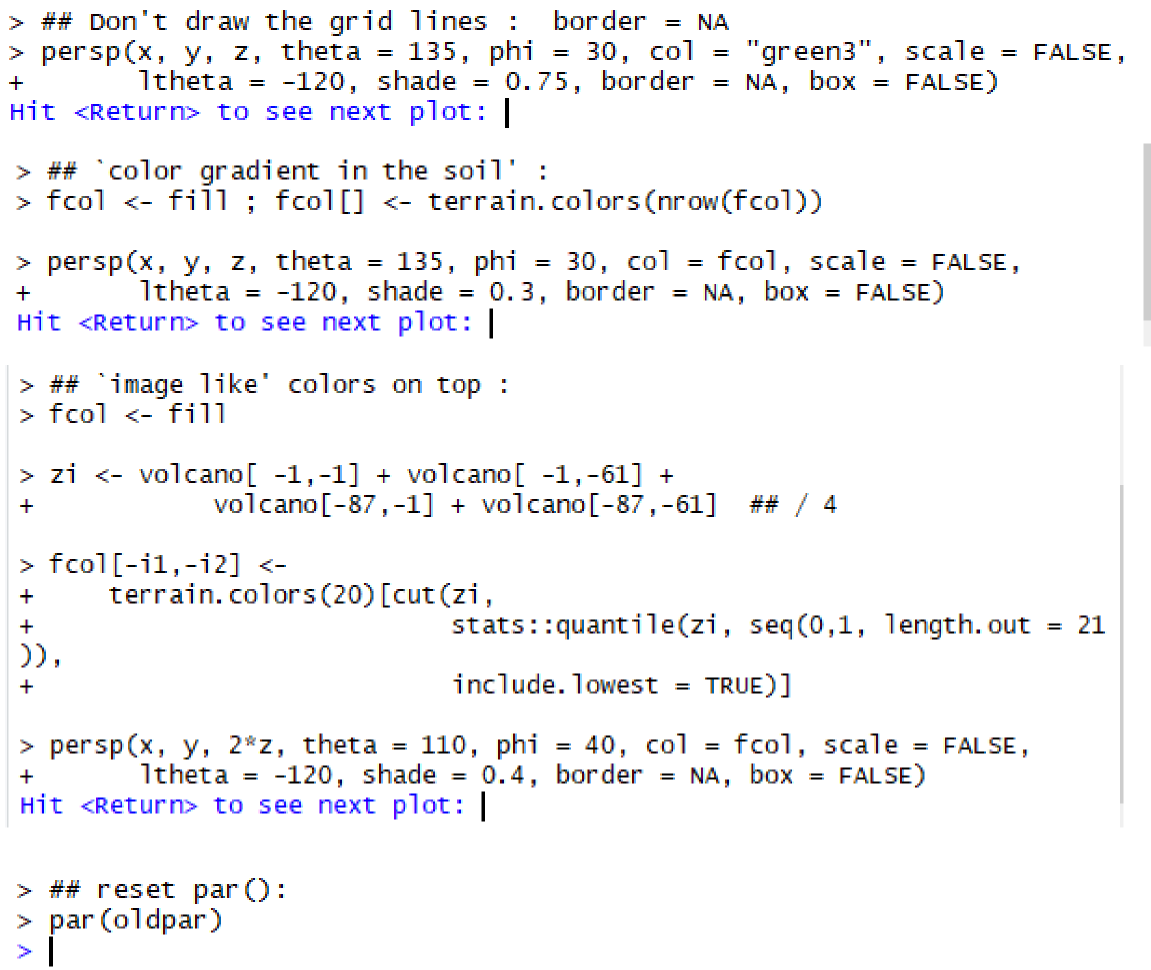
Demonstrations of R functions



```
> ## (3) Now something more complex
> ##     We border the surface, to make it more "slice like"
> ##     and color the top and sides of the surface differently.
>
> z0 <- min(z) - 20

> z <- rbind(z0, cbind(z0, z, z0), z0)

> x <- c(min(x) - 1e-10, x, max(x) + 1e-10)

> y <- c(min(y) - 1e-10, y, max(y) + 1e-10)

> fill <- matrix("green3", nrow = nrow(z)-1, ncol = ncol(z)-1)

> fill[ , i2 <- c(1,ncol(fill))] <- "gray"

> fill[i1 <- c(1,nrow(fill)) , ] <- "gray"

> par(bg = "lightblue")

> persp(x, y, z, theta = 120, phi = 15, col = fill, scale = FALSE, axes =
 FALSE)
Hit <Return> to see next plot: 
```

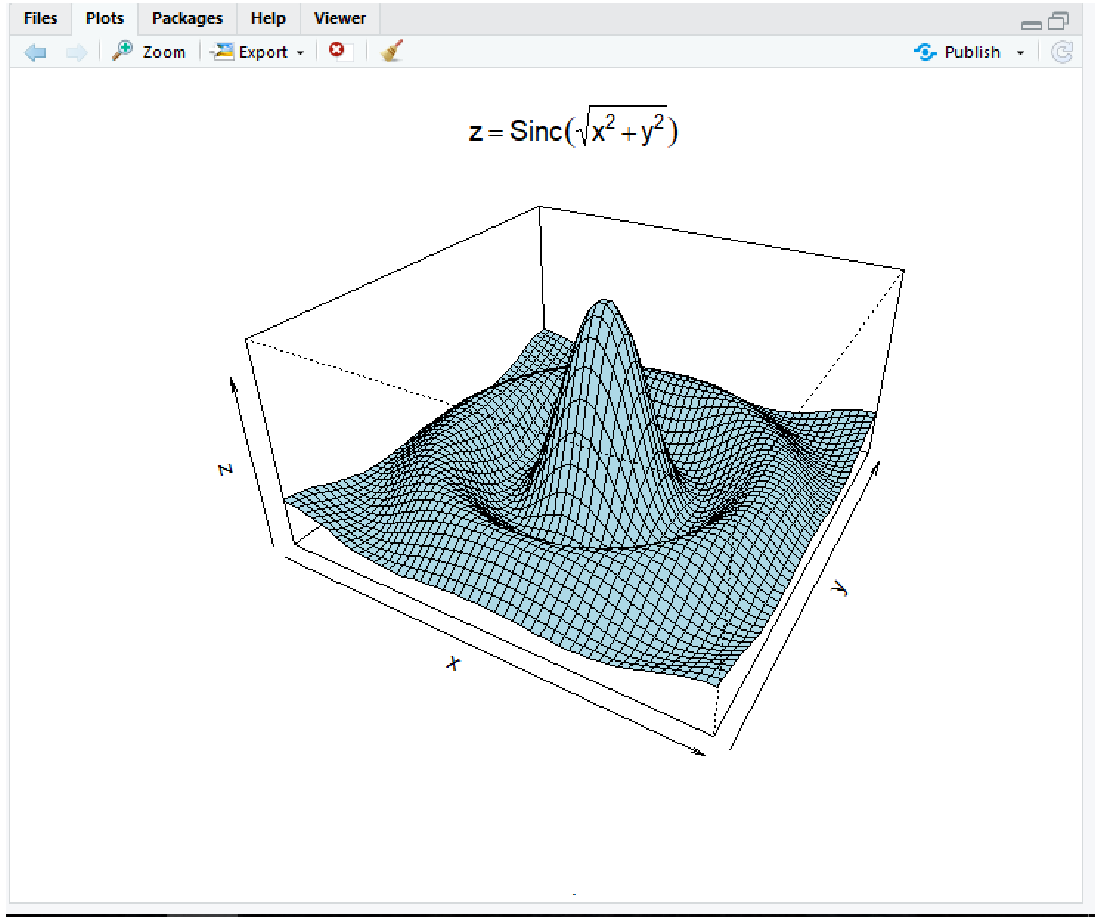


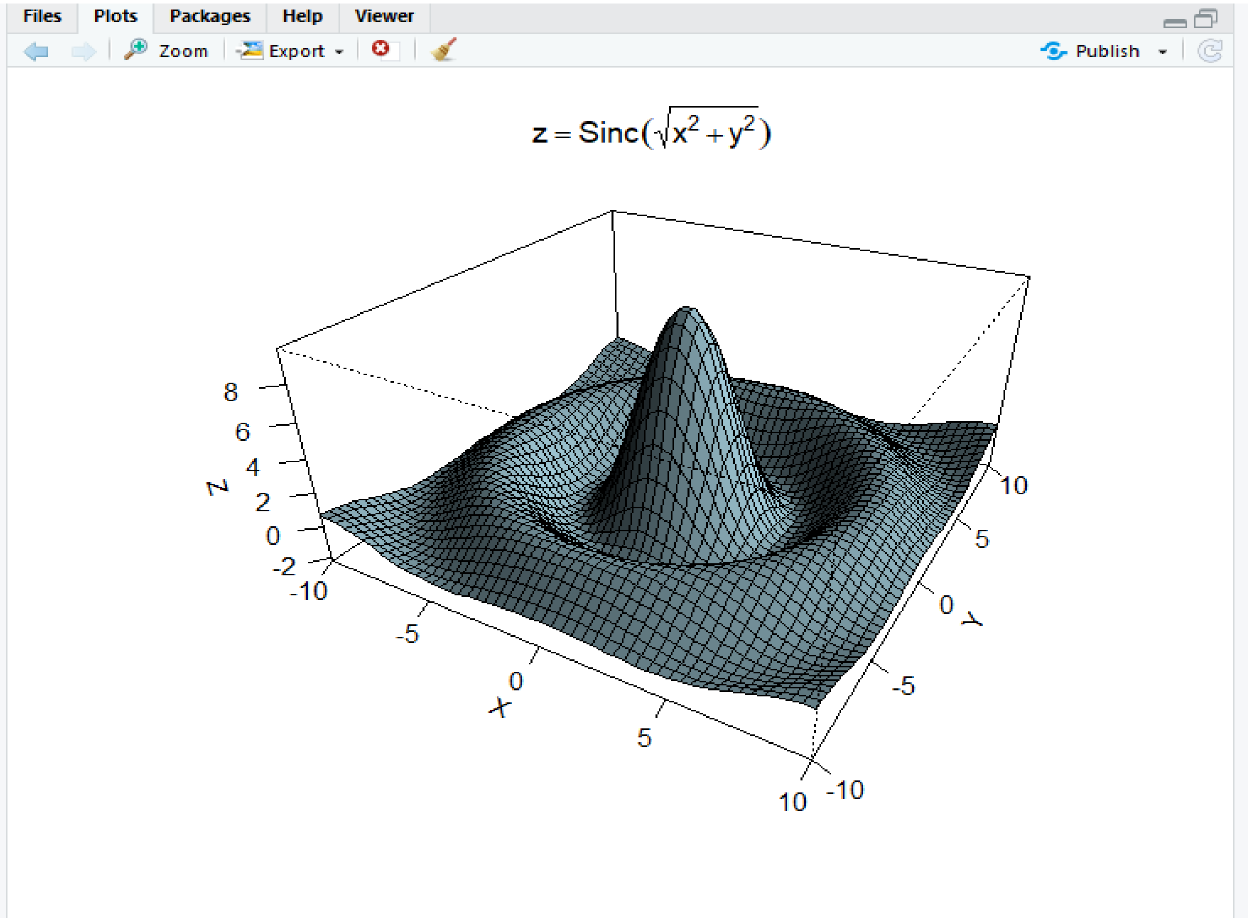
```
> title(main = "Maunga Whau\nOne of 50 Volcanoes in the Auckland Region."
,
+       font.main = 4)

> par(bg = "slategray")

> persp(x, y, z, theta = 135, phi = 30, col = fill, scale = FALSE,
+       ltheta = -120, lphi = 15, shade = 0.65, axes = FALSE)
Hit <Return> to see next plot: 
```



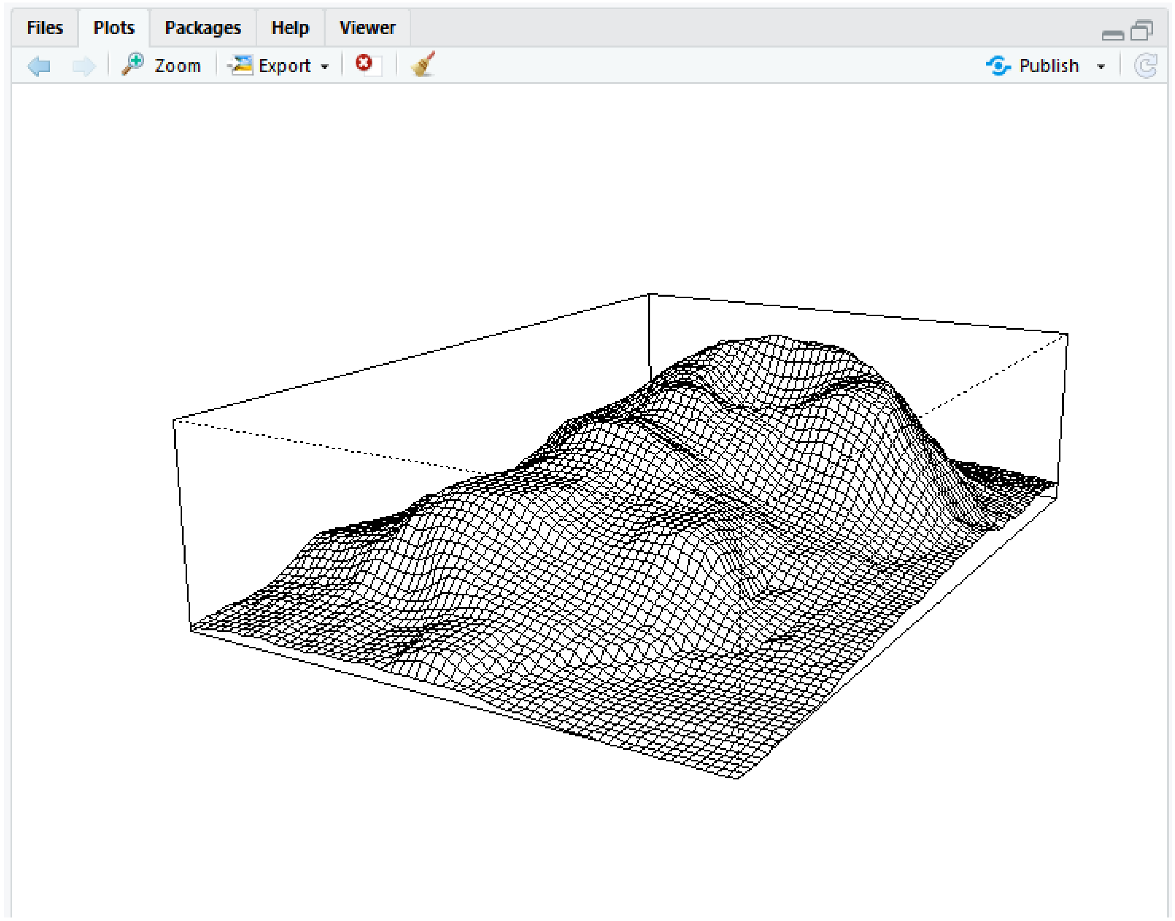
```
> ## Don't draw the grid lines :  border = NA
> persp(x, y, z, theta = 135, phi = 30, col = "green3", scale = FALSE,
+       ltheta = -120, shade = 0.75, border = NA, box = FALSE)
Hit <Return> to see next plot: 

> ## `color gradient in the soil' :
> fcol <- fill ; fcol[] <- terrain.colors(nrow(fcol))

> persp(x, y, z, theta = 135, phi = 30, col = fcol, scale = FALSE,
+       ltheta = -120, shade = 0.3, border = NA, box = FALSE)
Hit <Return> to see next plot: 

> ## `image like' colors on top :
> fcol <- fill

> zi <- volcano[ -1,-1] + volcano[ -1,-61] +
+           volcano[-87,-1] + volcano[-87,-61]  ## / 4

> fcol[-i1,-i2] <-
+     terrain.colors(20)[cut(zi,
+                        stats::quantile(zi, seq(0,1, length.out = 21
)),
+                        include.lowest = TRUE)]

> persp(x, y, 2*z, theta = 110, phi = 40, col = fcol, scale = FALSE,
+       ltheta = -120, shade = 0.4, border = NA, box = FALSE)
Hit <Return> to see next plot: 


> ## reset par():
> par(oldpar)
> 
```

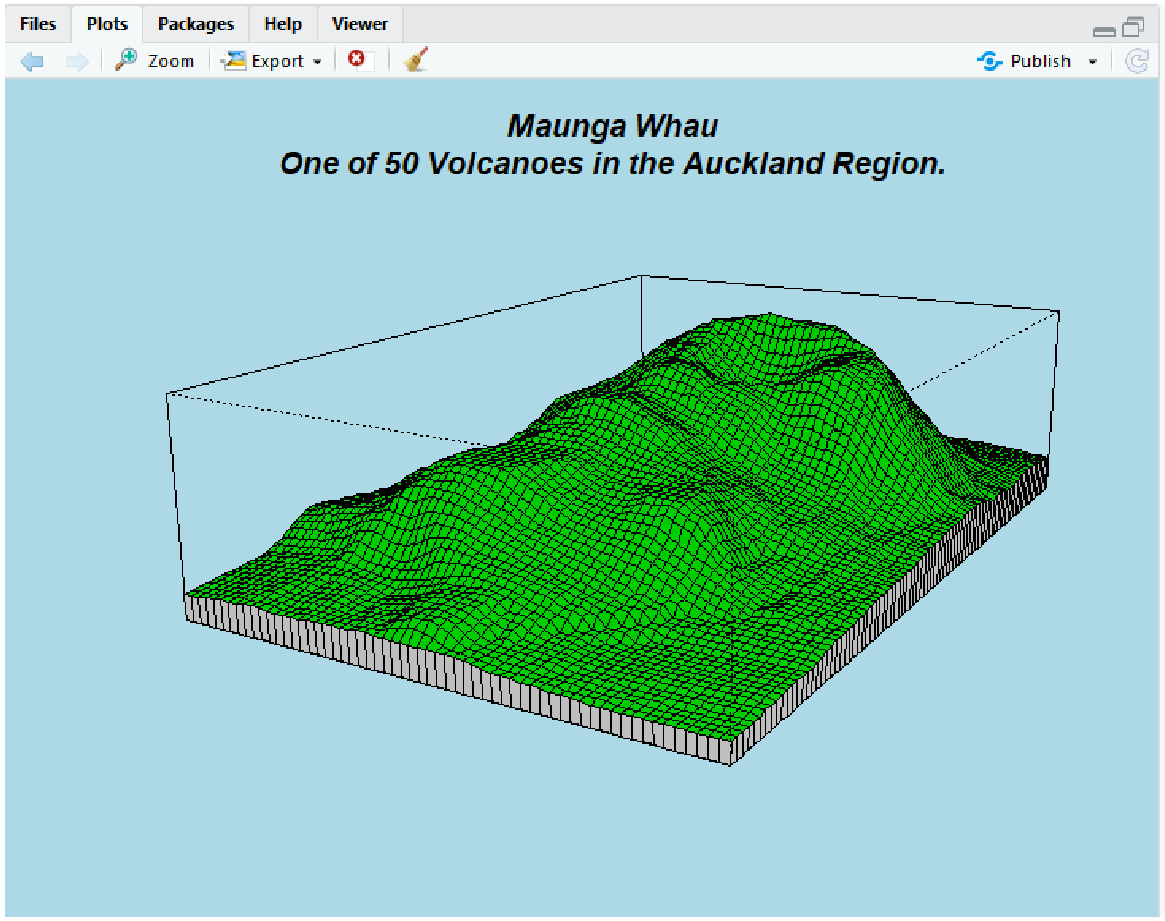
Demonstrations of R functions



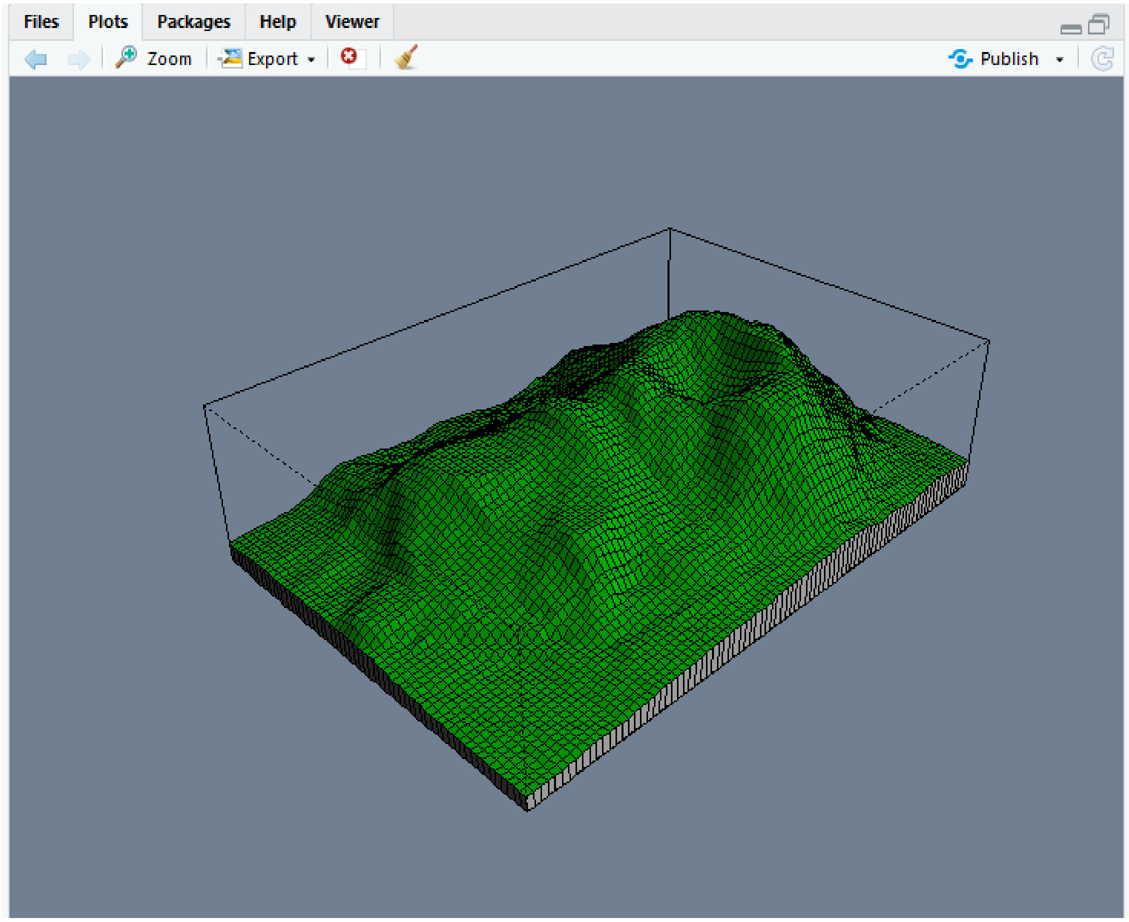


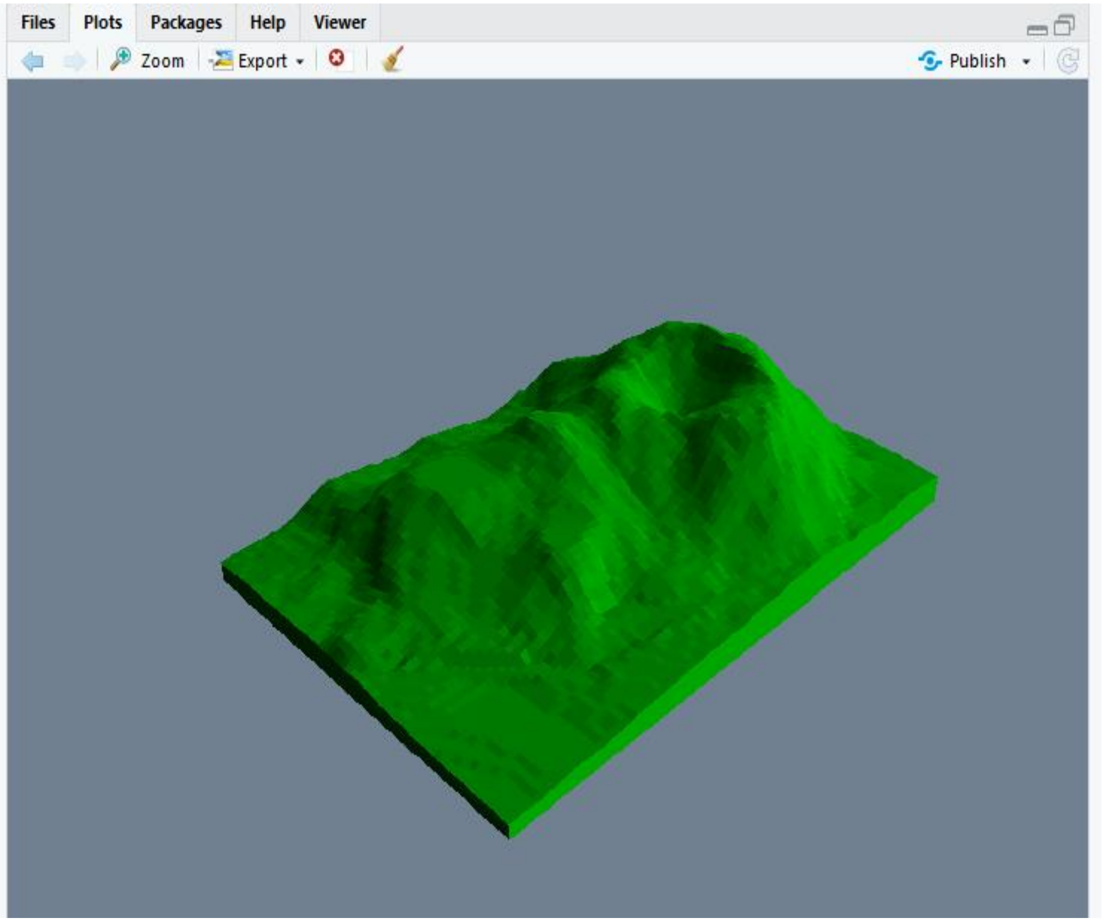
Demonstrations of R functions



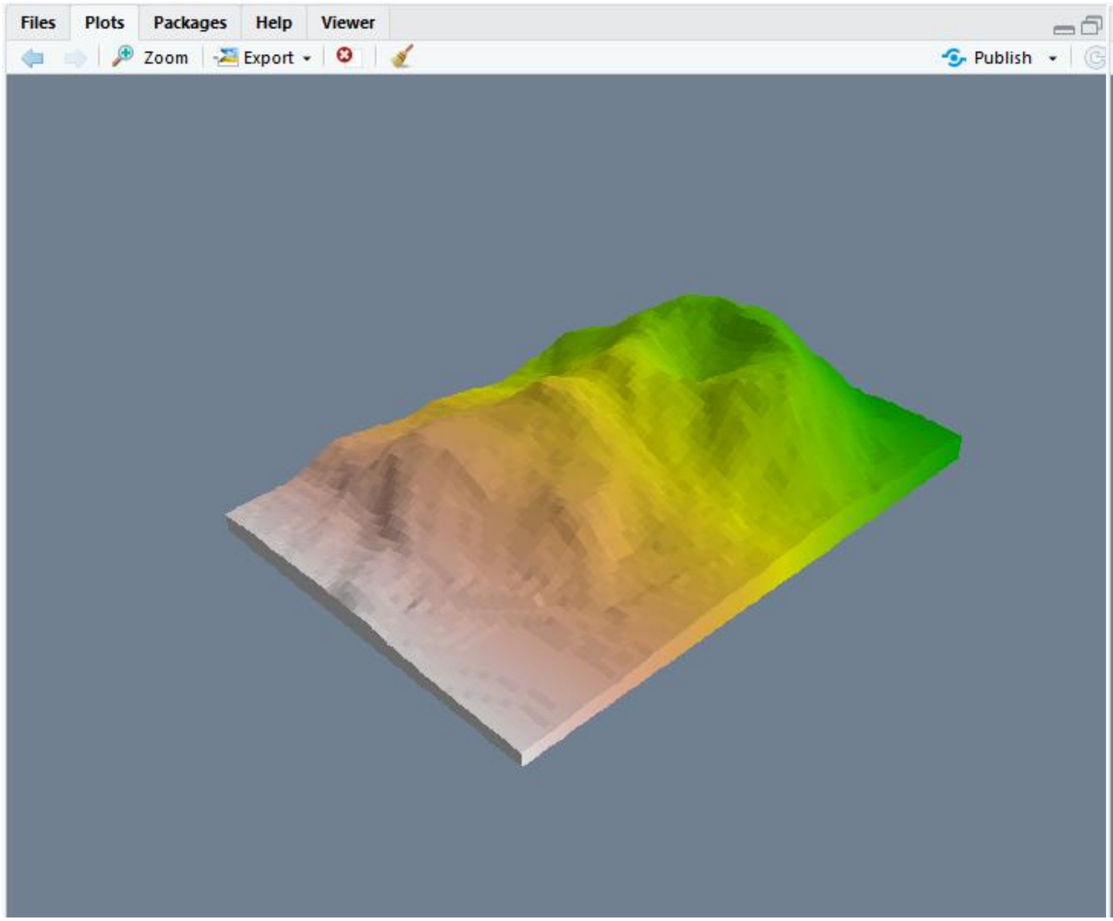


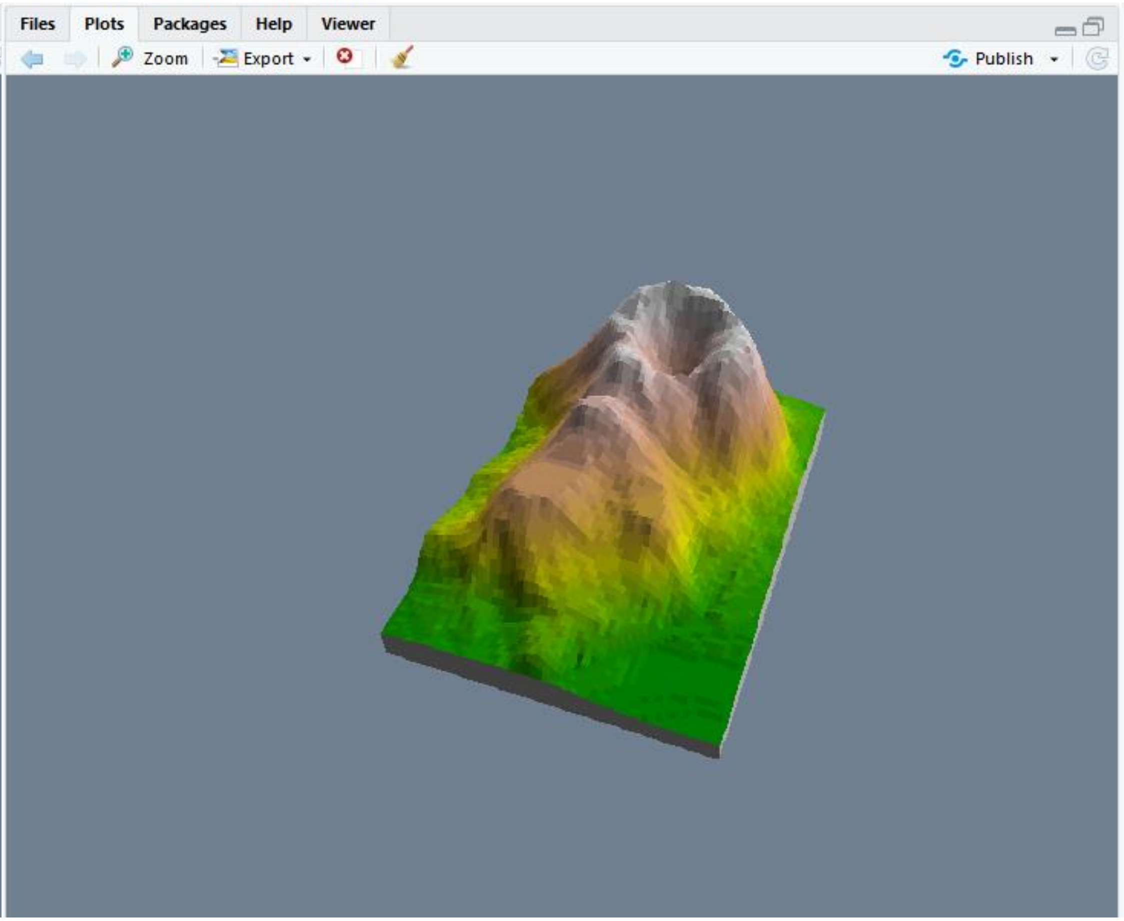
Demonstrations of R functions





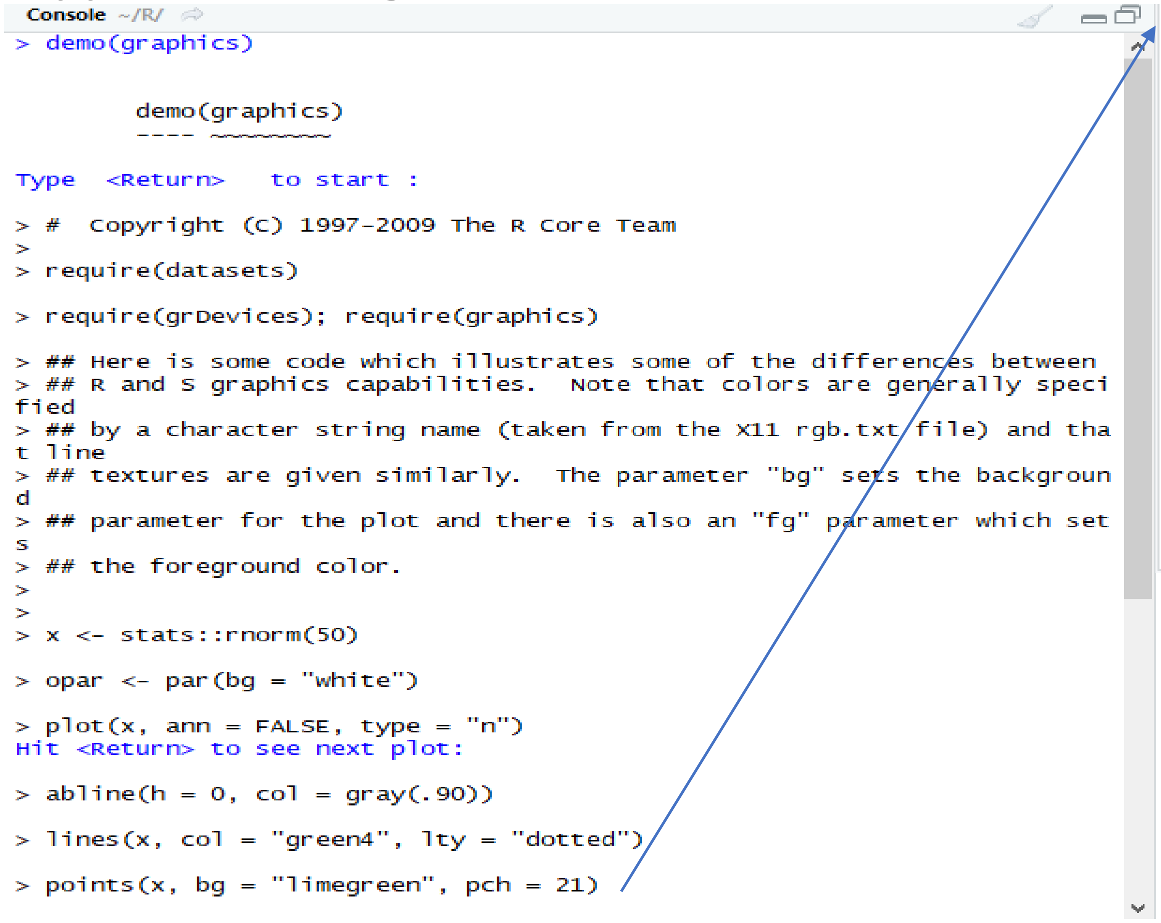
Demonstrations of R functions



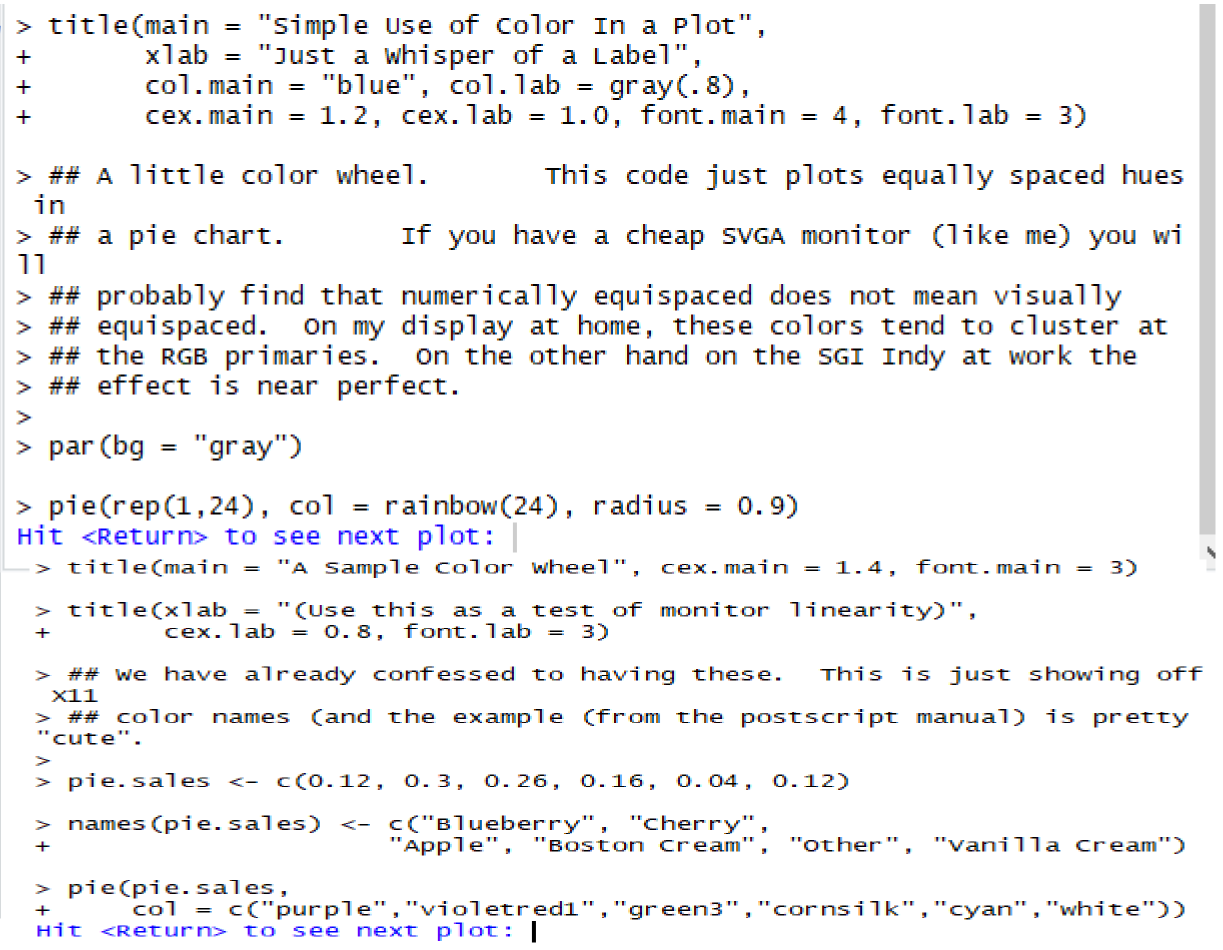


Demonstration of R functions

demo(graphics) #This can be another useful example for seeing the type of things that R can do.

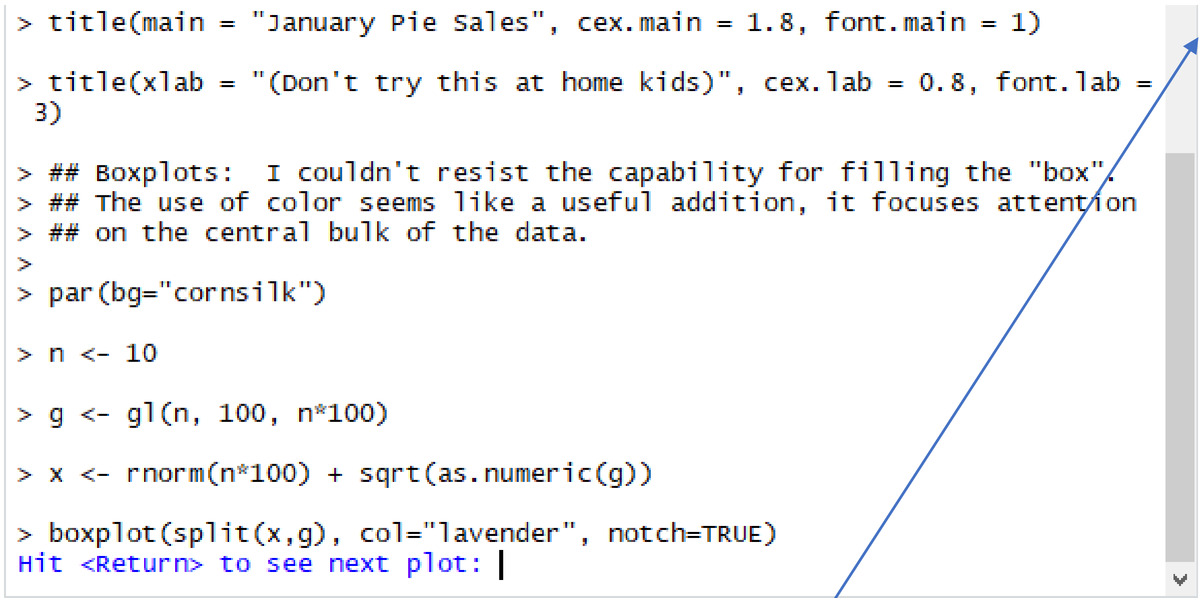


```
> demo(graphics)


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

Type  <Return>   to start :

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

> require(grDevices); require(graphics)

> ## Here is some code which illustrates some of the differences between
> ## R and S graphics capabilities.  Note that colors are generally speci
fied
> ## by a character string name (taken from the X11 rgb.txt file) and tha
t line
> ## textures are given similarly.  The parameter "bg" sets the backgroun
d
> ## parameter for the plot and there is also an "fg" parameter which set
s
> ## the foreground color.
>
>
> x <- stats::rnorm(50)

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

> plot(x, ann = FALSE, type = "n")
Hit <Return> to see next plot:

> abline(h = 0, col = gray(.90))

> lines(x, col = "green4", lty = "dotted")

> points(x, bg = "limegreen", pch = 21)
```

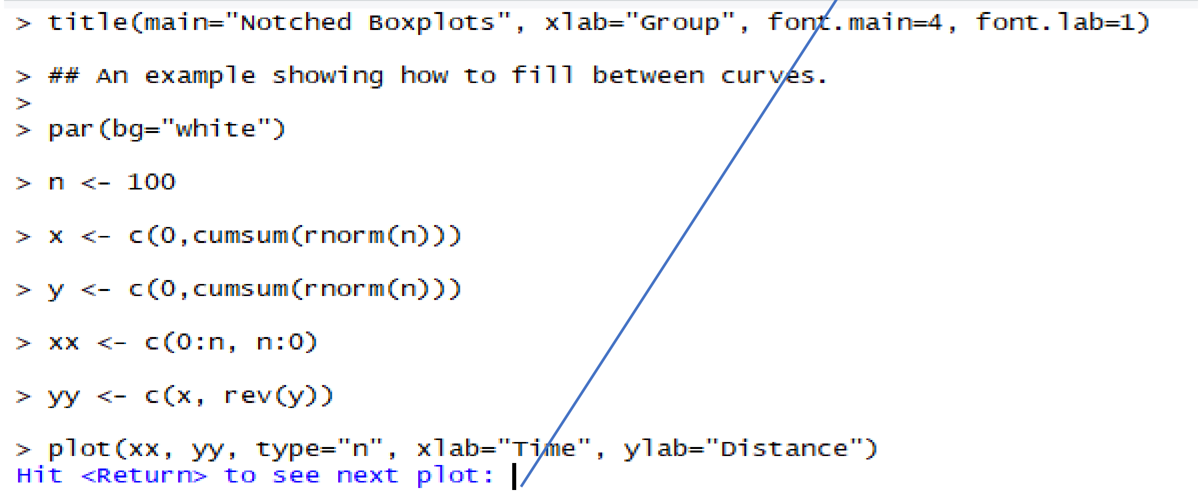


```
> title(main = "Simple Use of Color In a Plot",
+       xlab = "Just a Whisper of a Label",
+       col.main = "blue", col.lab = gray(.8),
+       cex.main = 1.2, cex.lab = 1.0, font.main = 4, font.lab = 3)

> ## 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 wi
ll
> ## 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)
Hit <Return> to see next plot:
 > 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".
 >
 > pie.sales <- c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)

 > names(pie.sales) <- c("Blueberry", "Cherry",
 +                      "Apple", "Boston Cream", "Other", "Vanilla Cream")

 > pie(pie.sales,
 +     col = c("purple","violetred1","green3","cornsilk","cyan","white"))
 Hit <Return> to see next plot:
```

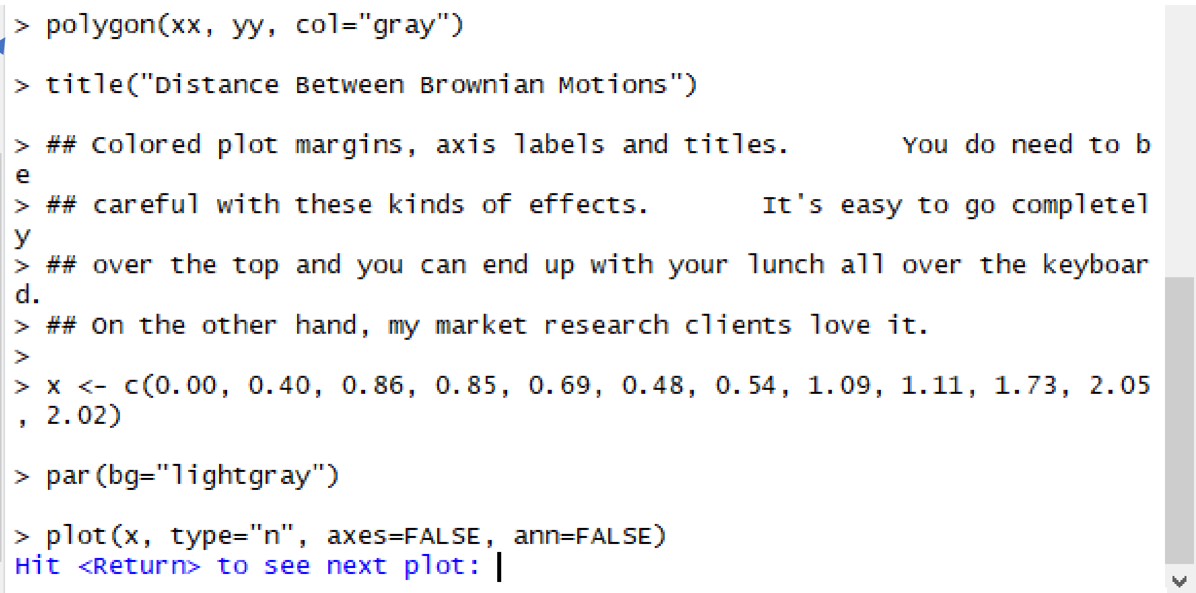
Demonstration of R functions



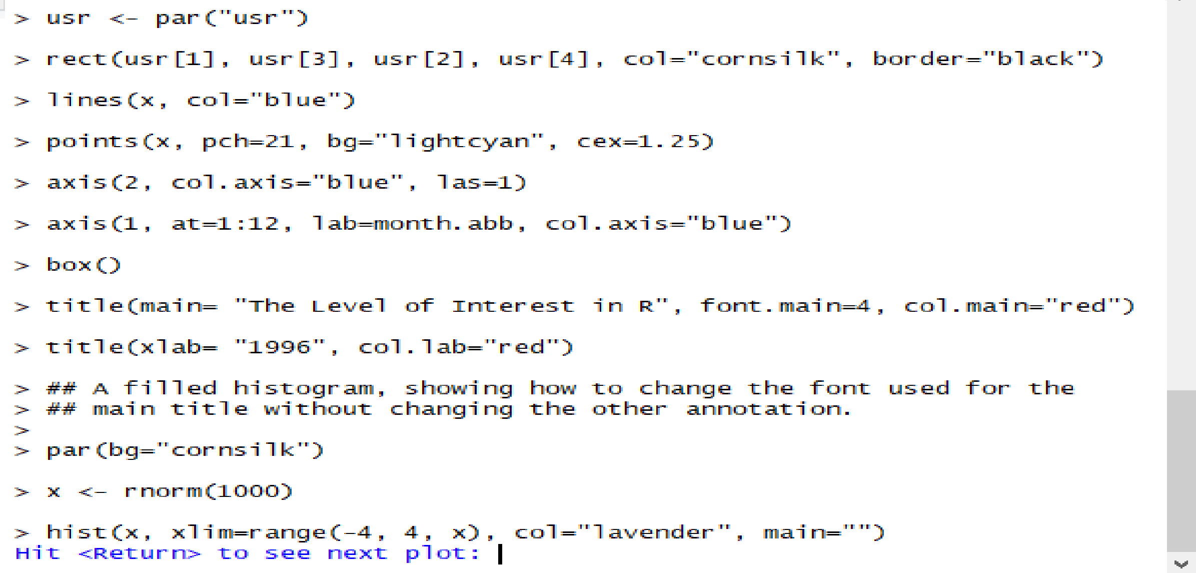
```
> title(main = "January Pie Sales", cex.main = 1.8, font.main = 1)

> title(xlab = "(Don't try this at home kids)", cex.lab = 0.8, font.lab =
 3)

> ## Boxplots:  I couldn't resist the capability for filling the "box".
> ## The use of color seems like a useful addition, it focuses attention
> ## on the central bulk of the data.
>
> par(bg="cornsilk")

> n <- 10

> g <- gl(n, 100, n*100)

> x <- rnorm(n*100) + sqrt(as.numeric(g))

> boxplot(split(x,g), col="lavender", notch=TRUE)
Hit <Return> to see next plot: 
```



```
> title(main="Notched Boxplots", xlab="Group", font.main=4, font.lab=1)

> ## An example showing how to fill between curves.
>
> par(bg="white")

> n <- 100

> x <- c(0,cumsum(rnorm(n)))

> y <- c(0,cumsum(rnorm(n)))

> xx <- c(0:n, n:0)

> yy <- c(x, rev(y))

> plot(xx, yy, type="n", xlab="Time", ylab="Distance")
Hit <Return> to see next plot: 
```

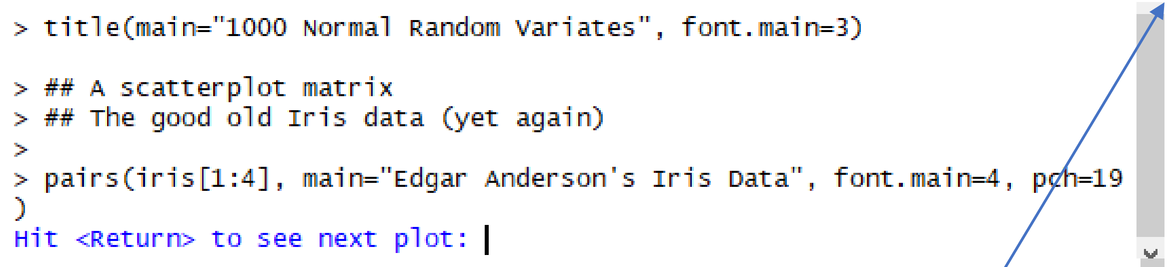


```
> polygon(xx, yy, col="gray")

> title("Distance Between Brownian Motions")

> ## Colored plot margins, axis labels and titles.      You do need to b
e
> ## careful with these kinds of effects.      It's easy to go completel
y
> ## over the top and you can end up with your lunch all over the keyboar
d.
> ## On the other hand, my market research clients love it.
>
> x <- c(0.00, 0.40, 0.86, 0.85, 0.69, 0.48, 0.54, 1.09, 1.11, 1.73, 2.05
, 2.02)

> par(bg="lightgray")

> plot(x, type="n", axes=FALSE, ann=FALSE)
Hit <Return> to see next plot: 
```

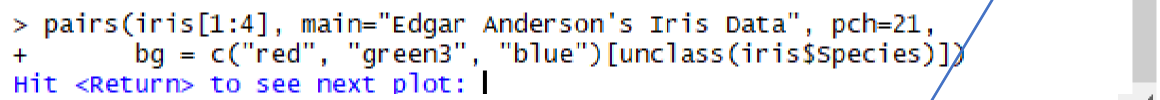


```
> usr <- par("usr")

> rect(usr[1], usr[3], usr[2], usr[4], col="cornsilk", border="black")

> lines(x, col="blue")

> points(x, pch=21, bg="lightcyan", cex=1.25)

> axis(2, col.axis="blue", las=1)

> axis(1, at=1:12, lab=month.abb, col.axis="blue")

> box()

> title(main= "The Level of Interest in R", font.main=4, col.main="red")

> title(xlab= "1996", col.lab="red")

> ## A filled histogram, showing how to change the font used for the
> ## main title without changing the other annotation.
>
> par(bg="cornsilk")

> x <- rnorm(1000)

> hist(x, xlim=range(-4, 4, x), col="lavender", main="")
Hit <Return> to see next plot: 
```

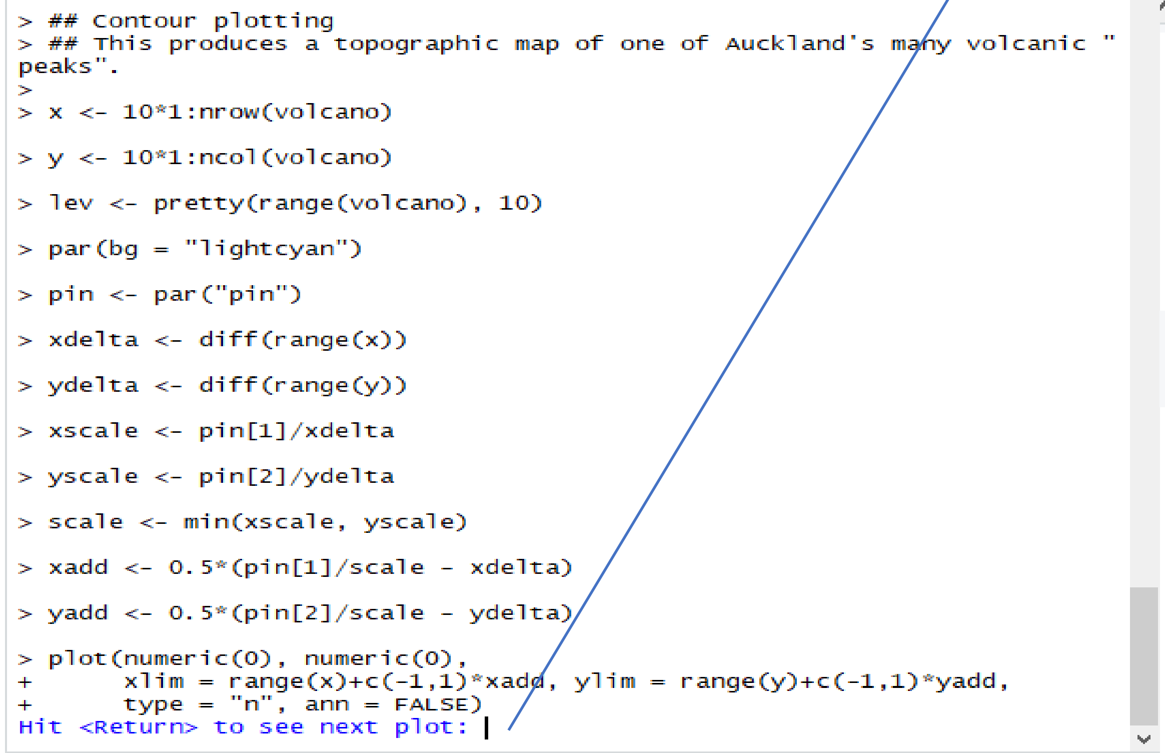
Demonstration of R functions



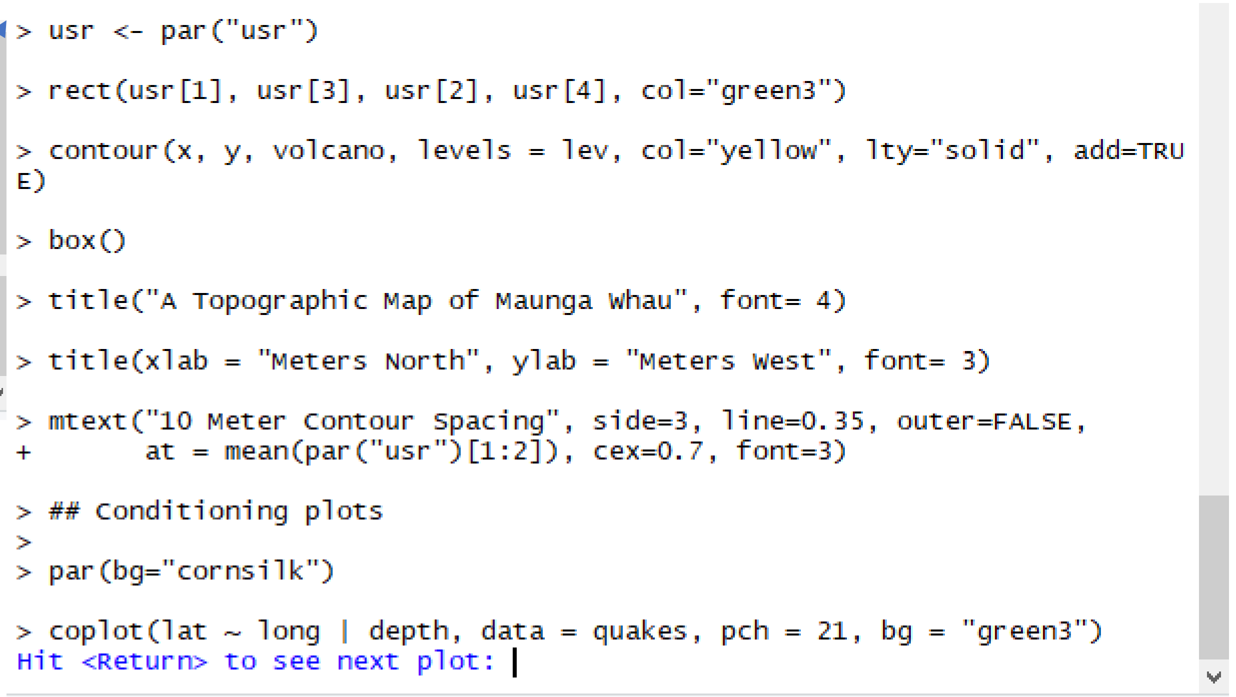
```
> title(main="1000 Normal Random Variates", font.main=3)

> ## A scatterplot matrix
> ## The good old Iris data (yet again)
>
> pairs(iris[1:4], main="Edgar Anderson's Iris Data", font.main=4, pch=19
)
Hit <Return> to see next plot: |
```



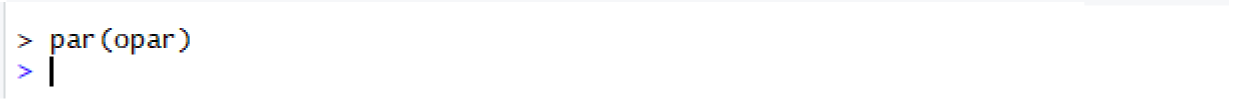
```
> pairs(iris[1:4], main="Edgar Anderson's Iris Data", pch=21,
+       bg = c("red", "green3", "blue")[unclass(iris$Species)])
Hit <Return> to see next plot: |
```



```
> ## Contour plotting
> ## This produces a topographic map of one of Auckland's many volcanic "
peaks".
>
> x <- 10*1:nrow(volcano)

> y <- 10*1:ncol(volcano)

> lev <- pretty(range(volcano), 10)

> par(bg = "lightcyan")

> pin <- par("pin")

> xdelta <- diff(range(x))

> ydelta <- diff(range(y))

> xscale <- pin[1]/xdelta

> yscale <- pin[2]/ydelta

> scale <- min(xscale, yscale)

> xadd <- 0.5*(pin[1]/scale - xdelta)

> yadd <- 0.5*(pin[2]/scale - ydelta)

> plot(numeric(0), numeric(0),
+      xlim = range(x)+c(-1,1)*xadd, ylim = range(y)+c(-1,1)*yadd,
+      type = "n", ann = FALSE)
Hit <Return> to see next plot: |
```

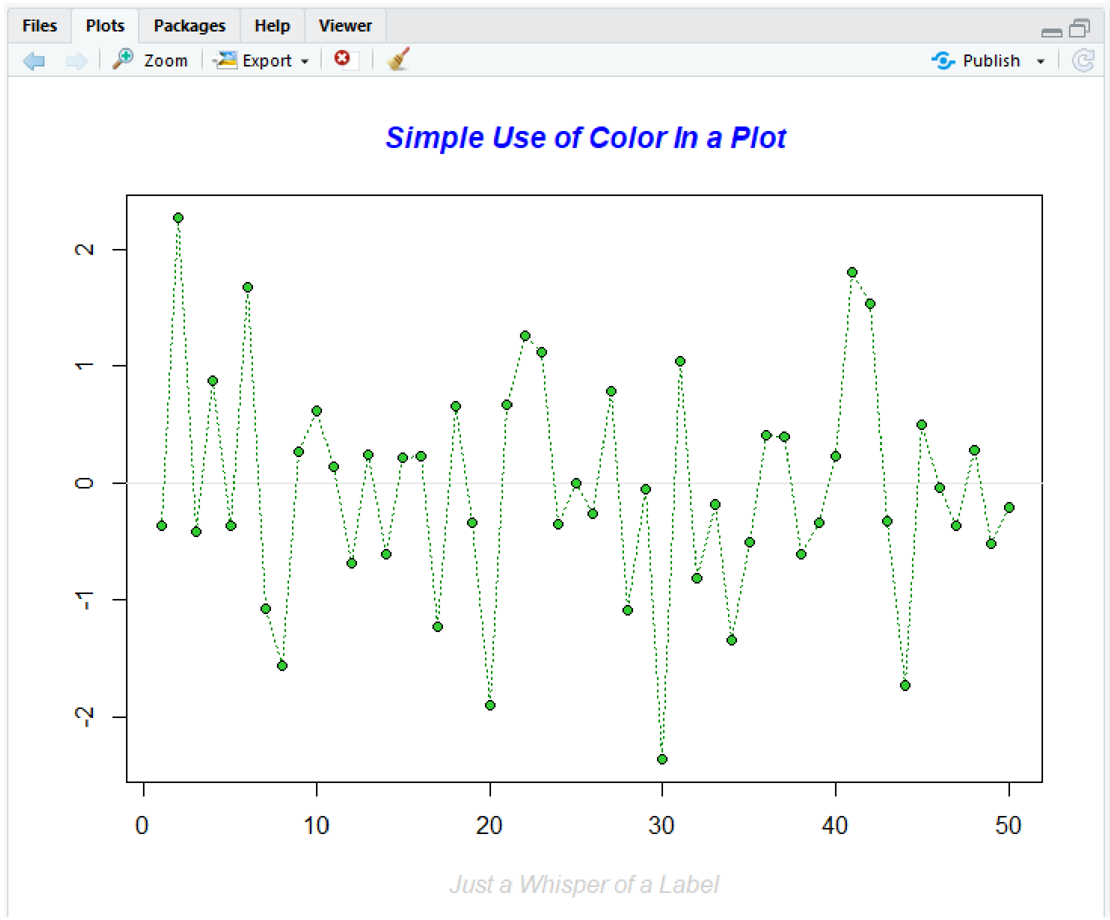


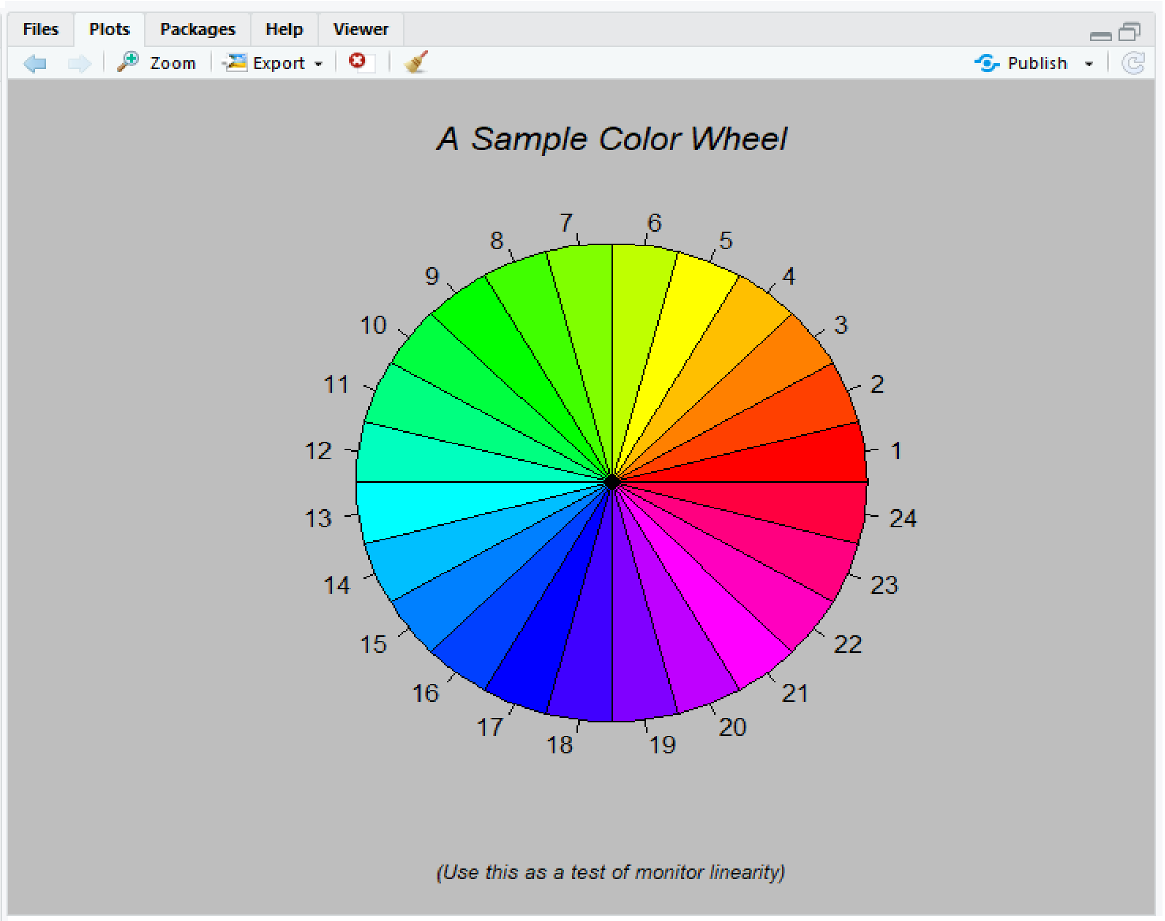
```
> usr <- par("usr")

> rect(usr[1], usr[3], usr[2], usr[4], col="green3")

> contour(x, y, volcano, levels = lev, col="yellow", lty="solid", add=TRU
E)

> box()

> title("A Topographic Map of Maunga Whau", font= 4)

> title(xlab = "Meters North", ylab = "Meters West", font= 3)

> mtext("10 Meter Contour Spacing", side=3, line=0.35, outer=FALSE,
+       at = mean(par("usr")[1:2]), cex=0.7, font=3)

> ## Conditioning plots
>
> par(bg="cornsilk")

> coplot(lat ~ long | depth, data = quakes, pch = 21, bg = "green3")
Hit <Return> to see next plot: |
```



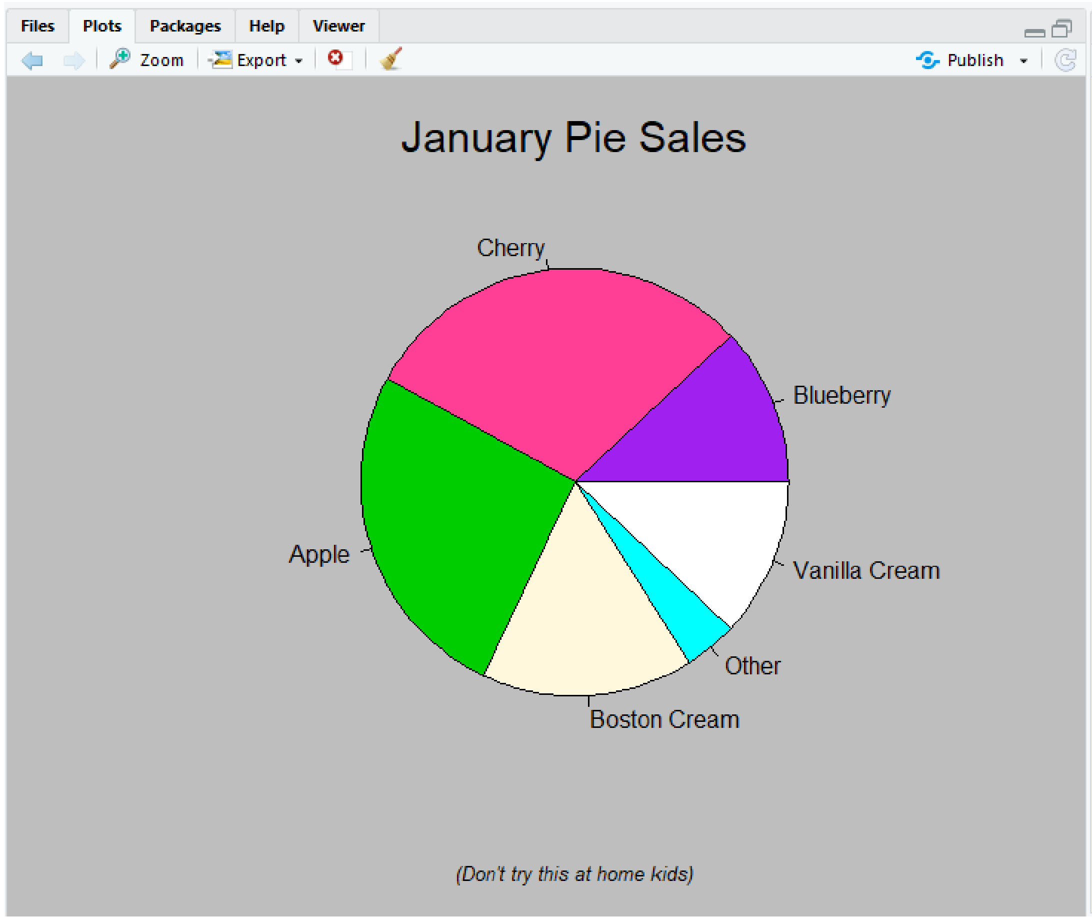
```
> par(opar)
> |
```

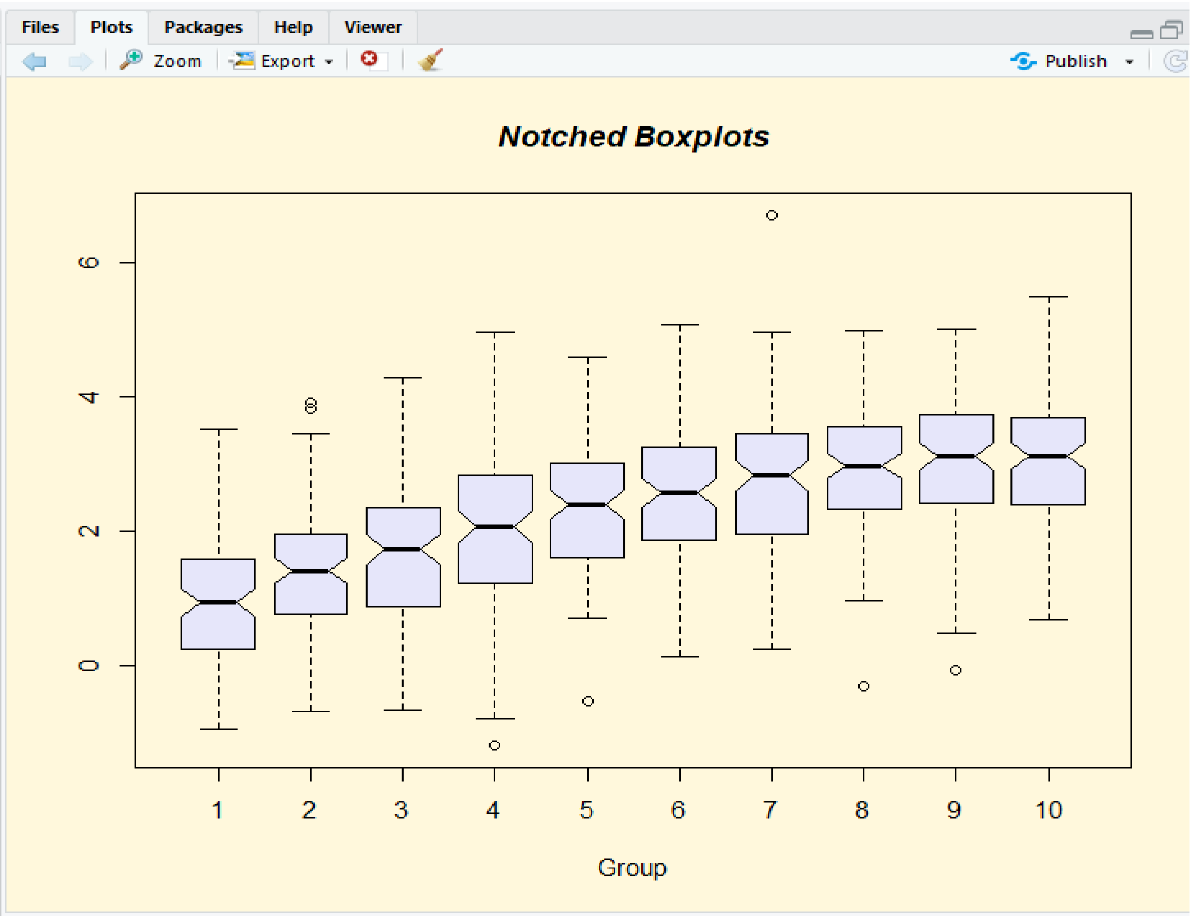
Demonstration of R functions



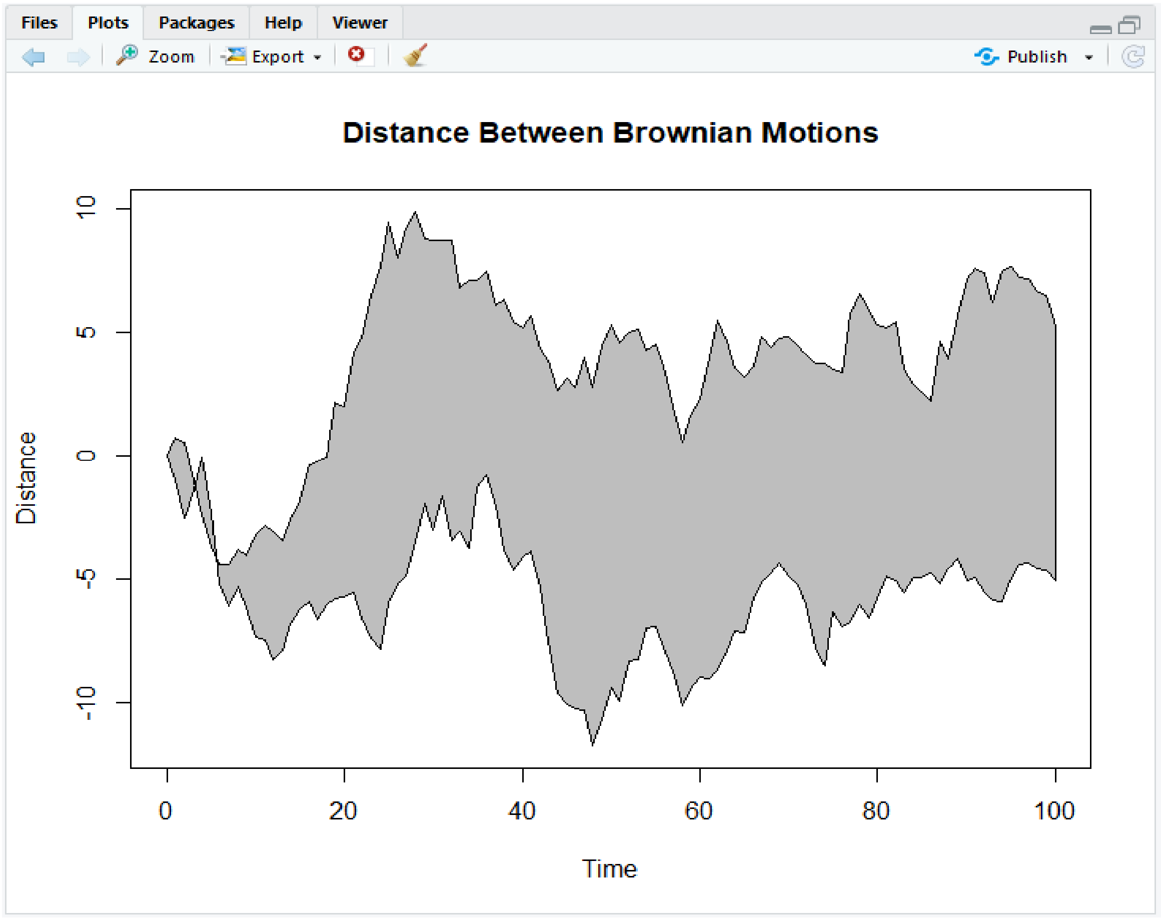


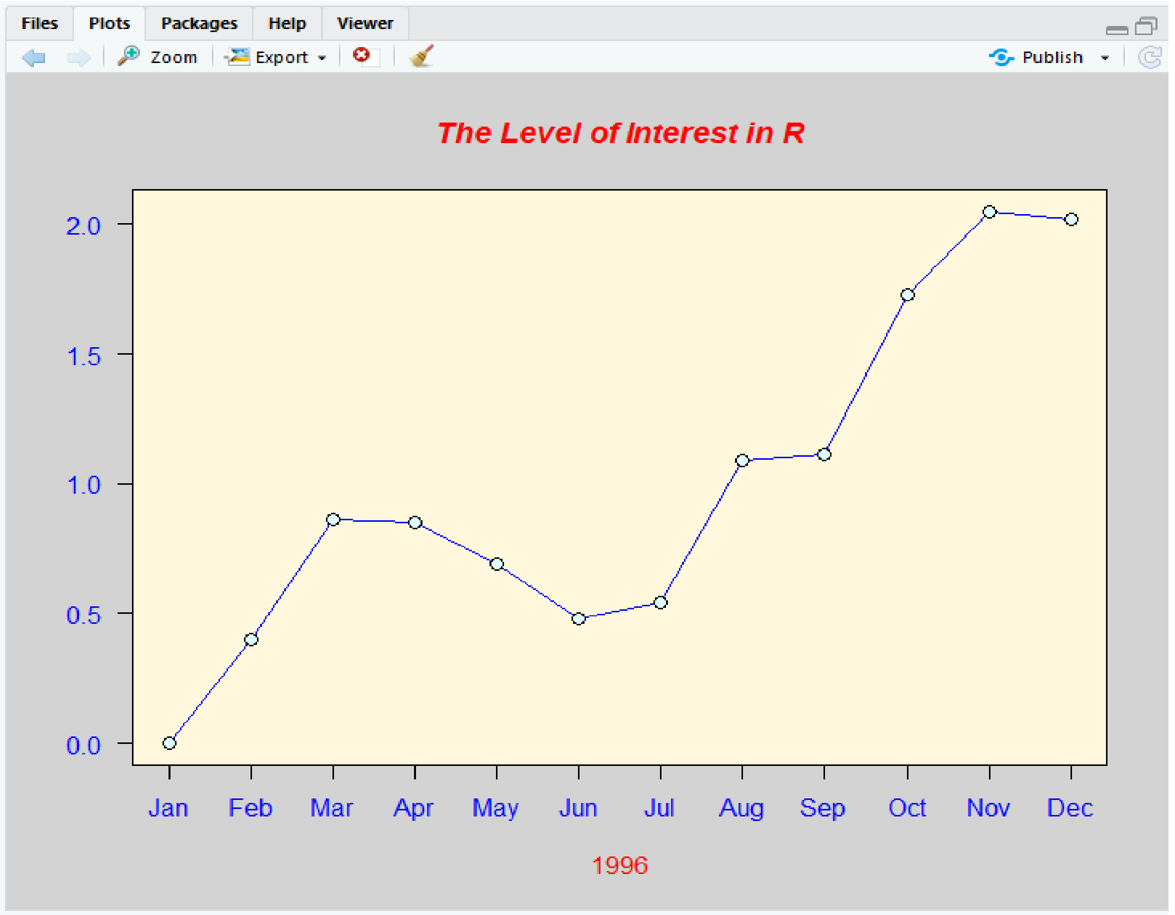
Demonstration of R functions



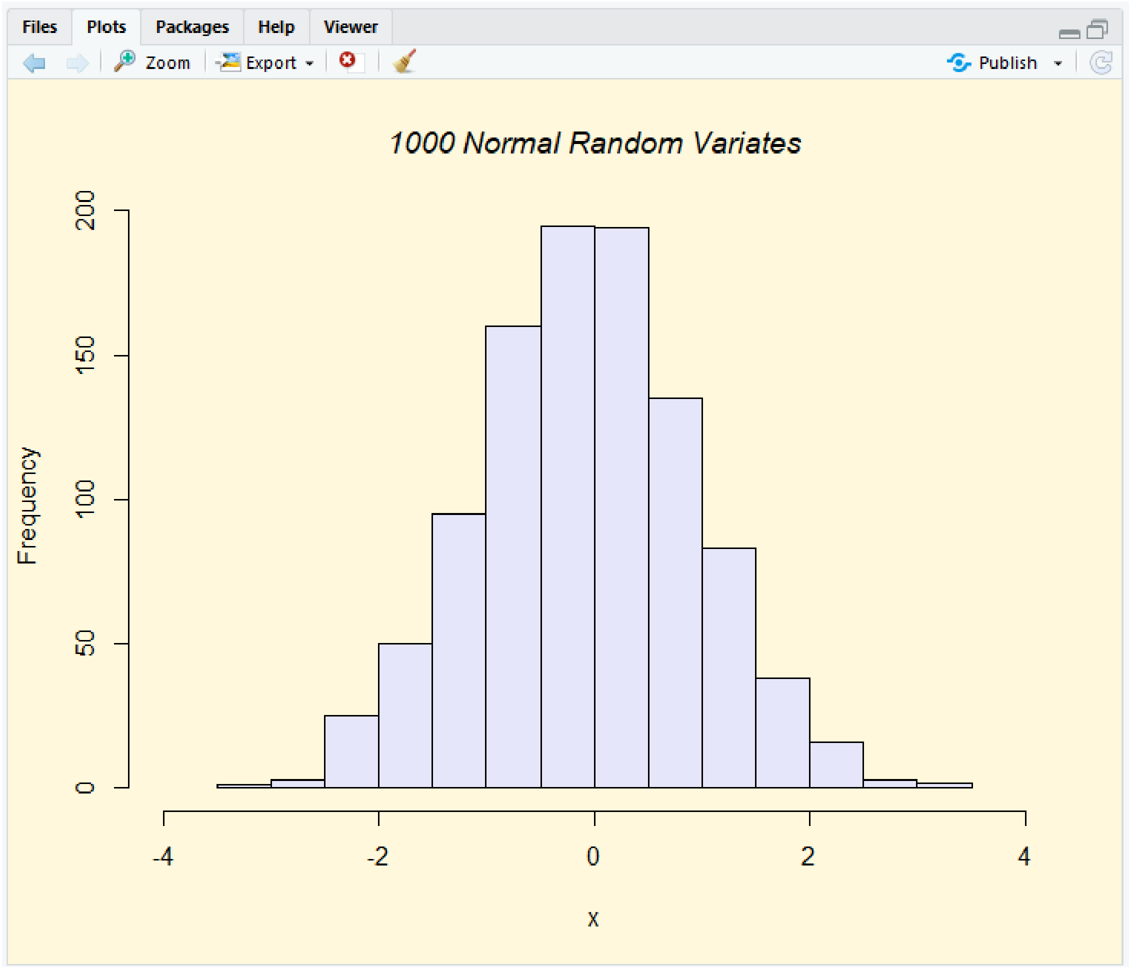


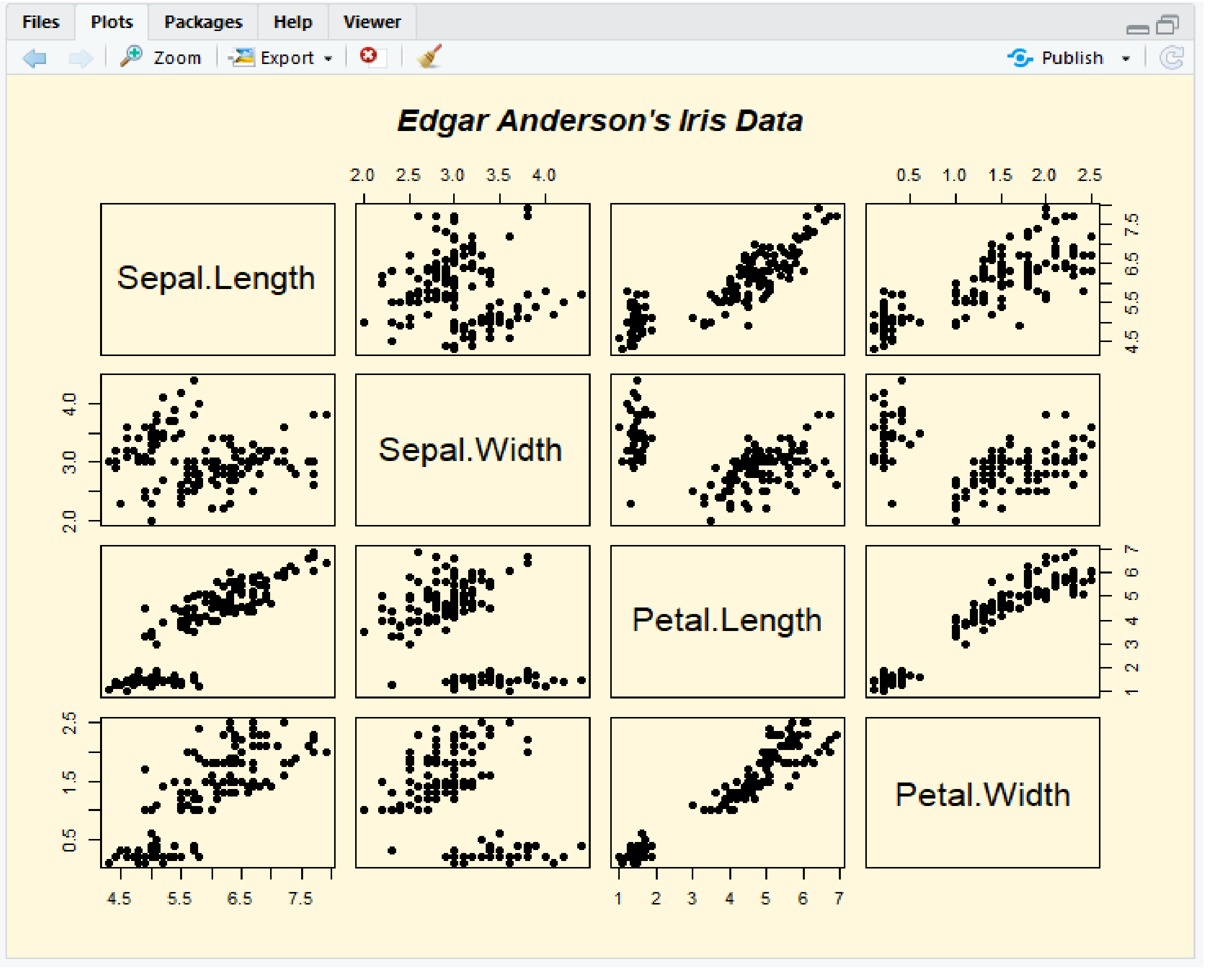
Demonstration of R functions



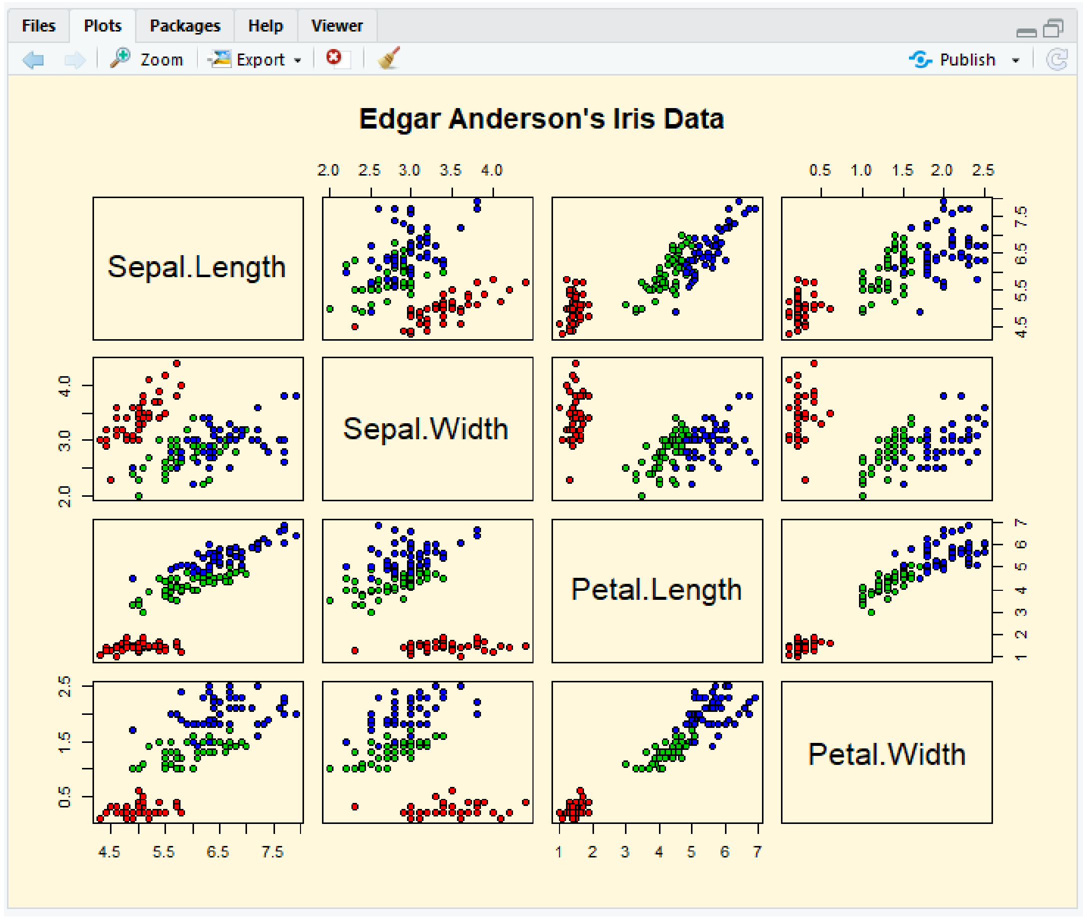


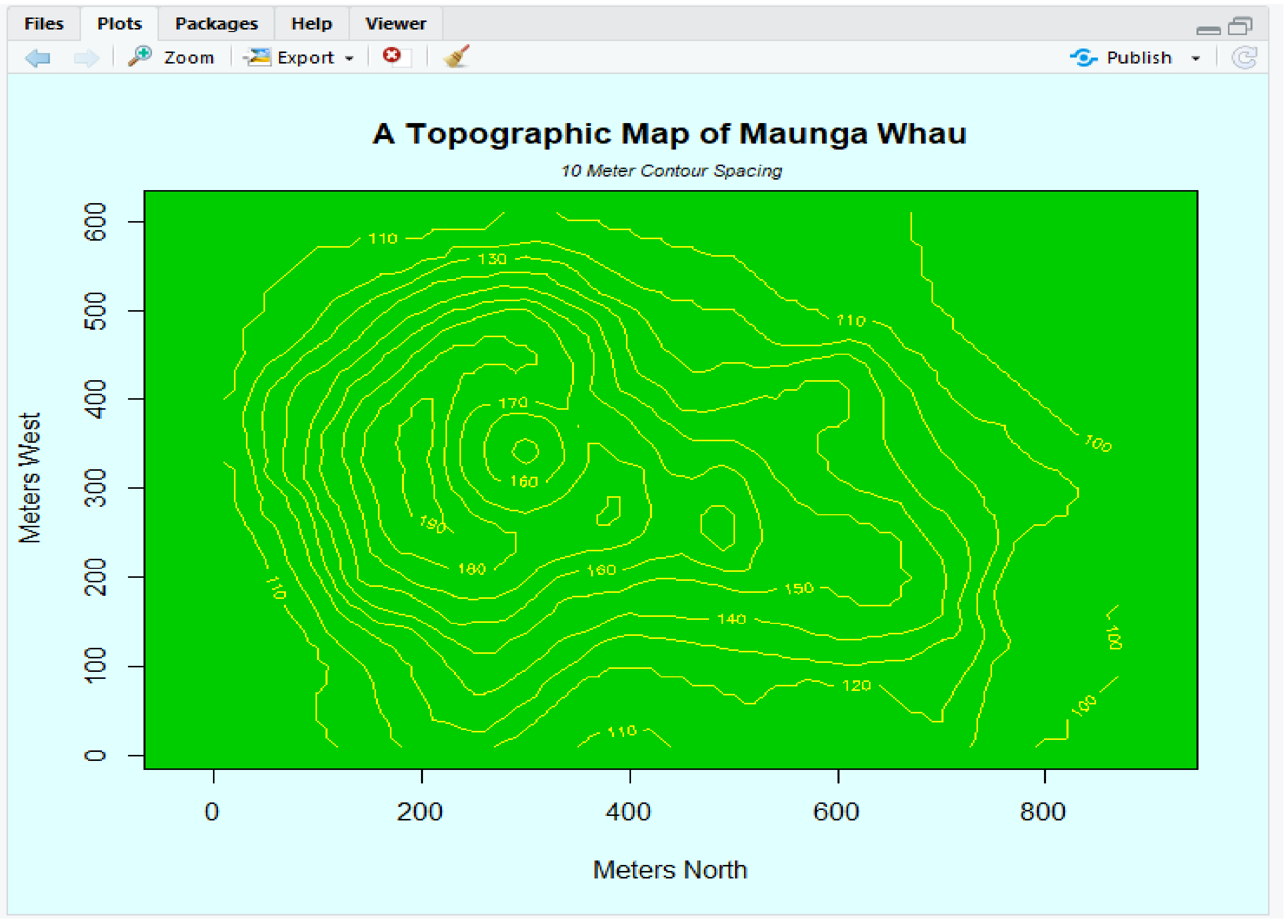
Demonstration of R functions



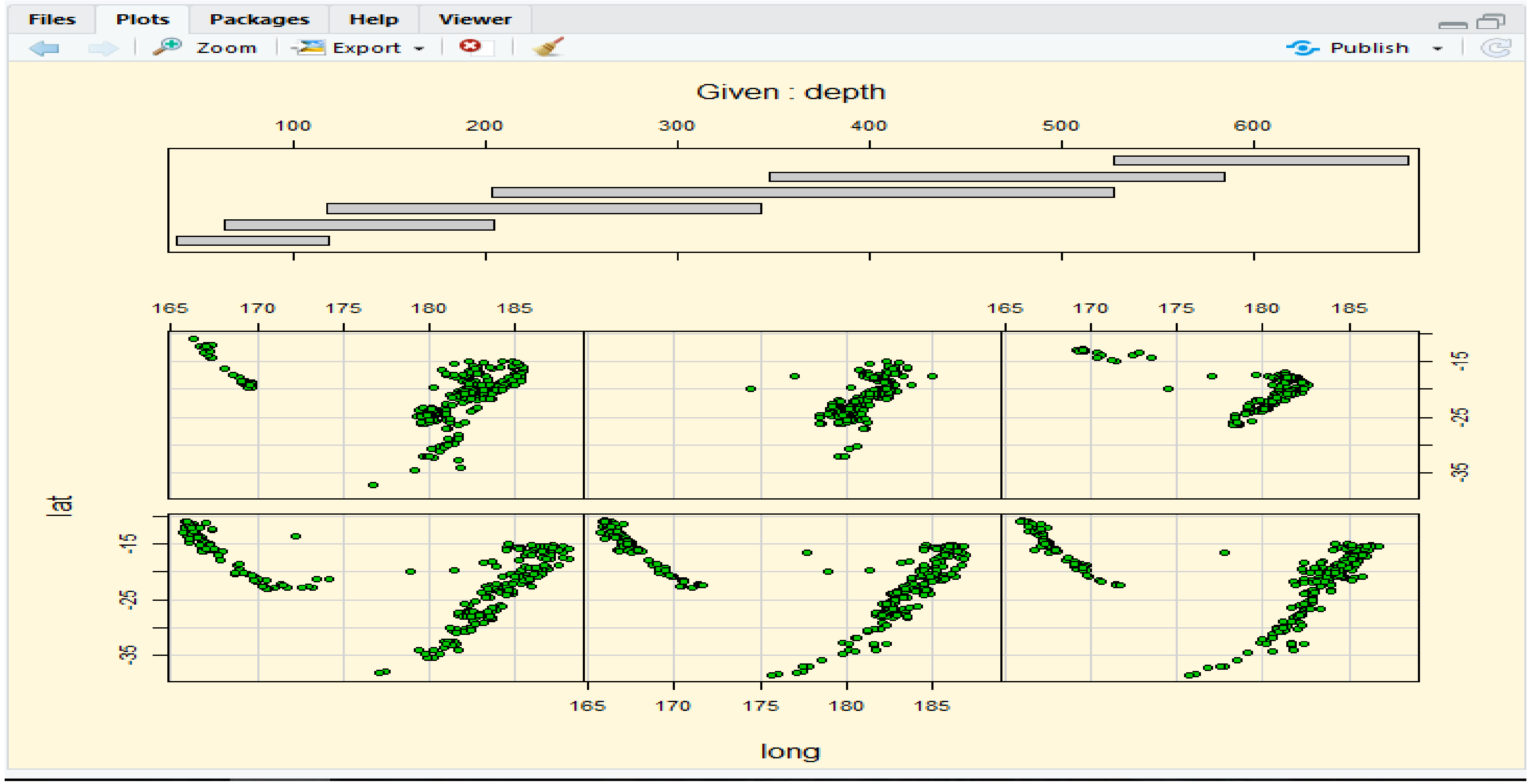


Demonstration of R functions



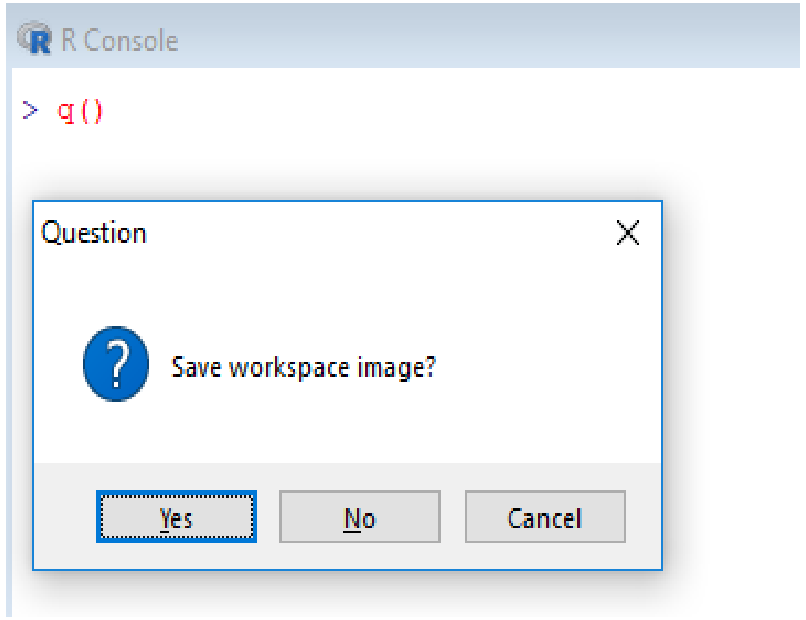


Demonstration of R functions



How to quit in R?

Type q() to quit in R



```
> q()
```

Question

Save workspace image?

Libraries in R

R provides many functions and one can also write own. Functions and datasets are organised into libraries

To use a library, simply type the library function with the name of the library in brackets.

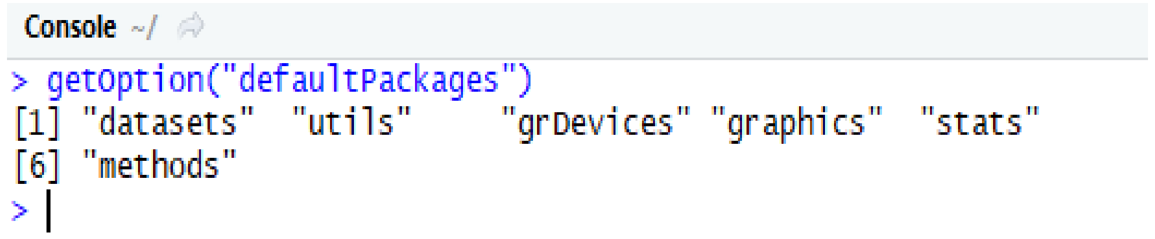
```
library(.)
```

For example, to load the spatial library type:

```
library(spatial)
```

Libraries in R

Examples of libraries that come as a part of base package in R.

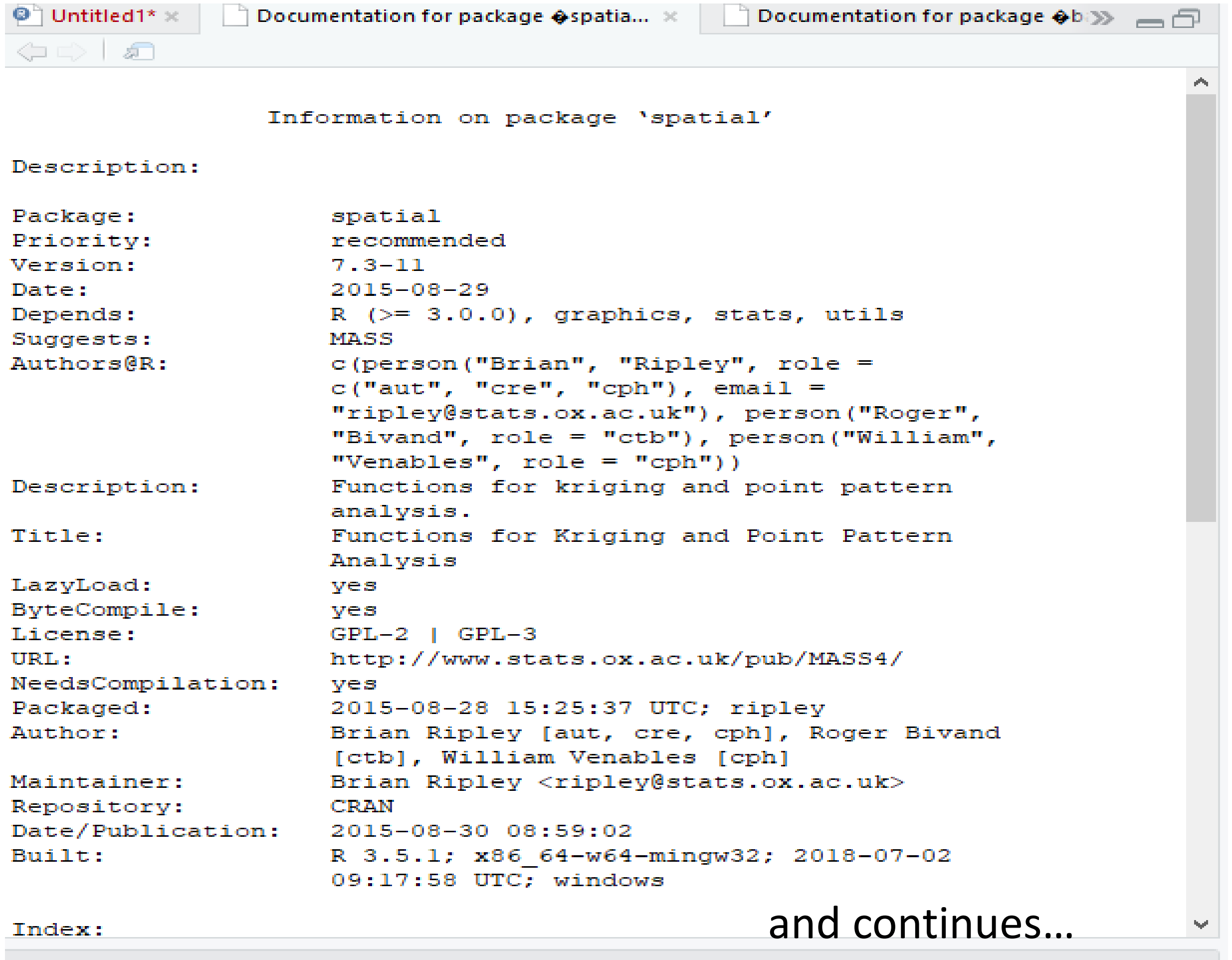


```
> getOption("defaultPackages")
[1] "datasets"  "utils"     "grDevices" "graphics"  "stats"
[6] "methods"
> |
```

Contents in Libraries

It is easy to use the help function to discover the contents of library packages. Here is how we find out about the contents of the spatial library:

library(help=spatial) returns →


Untitled1*
Documentation for package �spatia...
Documentation for package �b

```
                Information on package 'spatial'

Description:

Package:              spatial
Priority:             recommended
Version:              7.3-11
Date:                 2015-08-29
Depends:              R (>= 3.0.0), graphics, stats, utils
Suggests:             MASS
Authors@R:            c(person("Brian", "Ripley", role =
                      c("aut", "cre", "cph"), email =
                      "ripley@stats.ox.ac.uk"), person("Roger",
                      "Bivand", role = "ctb"), person("William",
                      "Venables", role = "cph"))
Description:          Functions for kriging and point pattern
                      analysis.
Title:                Functions for Kriging and Point Pattern
                      Analysis
LazyLoad:             yes
ByteCompile:          yes
License:              GPL-2 | GPL-3
URL:                  http://www.stats.ox.ac.uk/pub/MASS4/
NeedsCompilation:     yes
Packaged:             2015-08-28 15:25:37 UTC; ripley
Author:               Brian Ripley [aut, cre, cph], Roger Bivand
                      [ctb], William Venables [cph]
Maintainer:           Brian Ripley <ripley@stats.ox.ac.uk>
Repository:           CRAN
Date/Publication:     2015-08-30 08:59:02
Built:                R 3.5.1; x86_64-w64-mingw32; 2018-07-02
                      09:17:58 UTC; windows

Index:
```

and continues…

Installing Packages and Libraries

The base R package contains programs for basic operations.
It does not contain some of the libraries necessary for advanced statistical work.
Specific requirements are met by special packages.
They are downloaded and their downloading is very simple.
To install any package,

- run the R program,
- then on the command line, use the install.packages

function to download the libraries we want.

Installing Packages and Libraries

Examples :

ggplot2 is a system for declaratively creating graphics, based on **The Grammar of graphics**. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

dplyr is a powerful R-package to transform and summarize tabular data with rows and columns

The packages ggplot2 or dplyr can be installed by

```
install.packages(" ggplot2 ")

install.packages(" dplyr ")
```