**R – Approaching a Data Set**

***Find information about:***

* Recoding variables in R
* Regression analysis in R

**How to run** [**R**](http://www.r-project.org/) **in Batch mode**

http://mcs.une.edu.au/~stat356/Rbatch.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Sep 01-2012***

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.

**/\* Getting sample Datasets \*/**

> data ()

> data () #some of internal datasets in R (datasets examples)

> data (DNase) #DNase dataset example

> data (ChickWeight) # retreiving “ChickWeight” dataset

> subset (ChickWeight, weight > 300) #getting records where weight > 300

> nrow (subset (ChickWeight, weight > 300)) # counting a number of rows where weight > 300

# if we would like to find eows where weight = 300 we must right **weight == 300**

> nrow (ChickWeight) # counting number of rows

[1] 578

> ncol (ChickWeight) # counting number of columns

[1] 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**/\* Saving a graph as a .pdf file \*/**

> pdf("dimfile.pdf") # set graphical output file

> hist(rnorm(10000)) # generate 10,000 N(0,1) variates and plot their histogram

> dev.off() # close the graphical output file

> rnorm(100) # generating 100 random variates that normally distributed

Syntax -- rnorm(n, mean = 0, sd = 1)

> rnorm(100,25,2)

**/\* Selecting rows from the dataset \*/**

> subset (ChickWeight, weight > 300) #getting records where weight > 300

**Creating a vector (The c stands for *concatenate*.)**

> x <- c(1,2,4)

> q <- c(x,x,8) # we can use vectors in creating a vector

> q

[1] 1 2 4 1 2 4 8

**Accessing Individual elements of a vector [ ]**

> q [6]

[1] 4

> q [3:7]

[1] 4 1 2 4 8

**Finding the reminder while dividing numbers:**

> 7%%2 # (7/2 = 3; 🡪 7-(2\*3)=1)

[1] 1

**/\* Loops \*/**

for (n in x) {

if (n %% 2 == 1) k <- k+1

}

The same:

for (i in 1:length(x)) {

if (x[i] %% 2 == 1) k <- k+1

}

**Global and Local variables**

Variables inside of the funcyions are local

**1.4.1 Vectors, the R Workhorse**

The elements of a vector must all have the same ***mode***, or data type

*Scalars*, or individual numbers, do not really exist in R.

|  |  |  |
| --- | --- | --- |
| **Numeric Vector**  > x <- c(5,12,13)  > length(x)  [1] 3  > mode (x)  [1] **"numeric"** | **String Vector**  > y <- "abc"  > length(y)  [1] 1  > mode (y)  **[1] "character"** | **Mixed String Vector**  > z <- c("abc","29 88")  > length(z)  [1] 2  > mode (z)  **[1] "character"** |

**Working with Strings**

> u <- paste("abc","de","f") # concatenate the strings

> length (u)

[1] 1

> u

[1] "abc de f"

> v <- strsplit(u," ") ") # split the string according to blanks

> v

[[1]]

[1] "abc" "de" "f"

**1.4.3 Matrices**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

> Ctrl + L -- clean the screen

> rm(list=ls()) # Cleaning all the variables and data

> rm(list=ls(all=TRUE)) -- remove all the variables

> help.start()

> help (read.table) -- if you stuck

> help.search ("mean") **-- if you don’t know that exact name of the function**

> ?mean -- asking help about the “mean” function

> help (mean) -- asking help about the “mean” function

> colors () # getting all available colors

> objects () # checking what object we have in our workspace

> ls () # checking what object we have in our workspace

> rm ("z","sg", "data1", "j") -- # removing objects

> ls ()

[1] "a" "avl" "b" "char"

[5] "check" "d" "datafile" "DNase"

[9] "m" "new" "Orange" "q"

[13] "s" "sel" "settinggroup" "x"

[17] "y"

When you attach() a dataset, it loads the dataset into R's current searchpath. The same occurs with packages when you use library(). This can be seen with:

*> > search()*

> detach(partaq4)

> sample (DayCalls,3) # Taking a random sample from the string. Random 3 numbers from the column DayCalls

> length (x) # lentgh of the vector

> library () -- list all available packages

**Creating a sequence:**

> seq (-40,60,3) # From -40 to 60 with the step of 3

Logical operators:

a <- 1:10 # writing number to the vector a

a

sel <- (a>3)

We can use: &, |, ! signs

& for AND

| for OR

! for NO

***Character vector***

Ch <- (“Hello”, “World”, “!”)

***Index vector***

a<- 1:10

b <- a[a>3]

b

[1] 4 5 6 7 8 9 10

**Giving names to a vector**

> b <- a[1:3]

> b

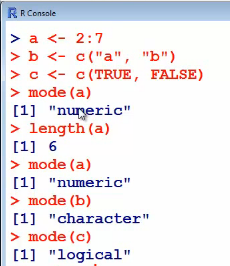
[1] 1 2 3

> names (b) <- c("Salary", "Age", "Income")

> b

Salary Age Income

1 2 3



------------------------------------------------------------

[**1. Get the data into R (read the file)**](#_1._Get_the)

**2. Assigning column names to variables**

> attach (<table name>)

**3. Changing the table or a column (“edit” of “fix”)**

> fix (SAT) -- gives us an opportunity to change data

> edit (<table name>)

> fix (<column name>)

**4. SUMMARY STATISTICS AND DATASET ANALYSIS**

> str (SAT) **-- gives variable in a row format**

> summary (SAT) **-- gives the summary of the table**

> density (x) **-- shows density distribution of a variable**

sum (x) – sum of the numbers

mean (x) -- mean

median (x) -- median

var (x) -- variance

sd (x) – standard deviation

max (x) -- maximum

min (x) -- minimum

quantile (x) -- quantile

> names (x) -- getting headers

> data1$Spending -- shows data for a particular column

> data1 [1:5,3] -- getting values from 1 to 5 from the column 3

> head (new) -- shows several first rows of the data set

> tail (new) -- shows several last rows of the data set

One of the first things to consider when receiving a dataset is to validate your assumption. Is the data clean? Does it make sense? I personally use ***head(ds)*** and ***tail(ds)*** to look at the 1st and last values.

Other functions include sd (standard deviation), var (variance), range (low value and high values), and **IQR** that displays the interquartile range (difference between 1st and 3rd quartiles). The cor() function computes the correlation between variables in the dataset, or, more specifically, the vectors provided as the values of x and y.

> stem (DayCharge)

The decimal point is 1 digit(s) to the right of the |

1 | 49

2 | 77

3 | 128

4 | 14

5 | 7

**II. DATA VISUALIZATION**

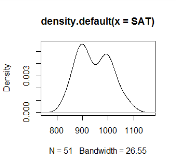
> library (graphics)

> dotchart (DayCharge)

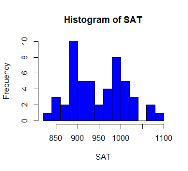
> stripchart (DayCharge)

> plot (sort (DayCharge)) # for low volume data

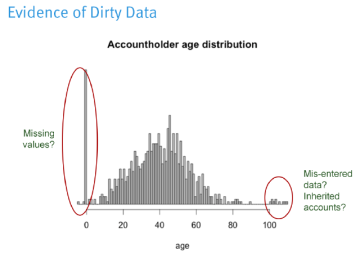
> barplot (DayCharge)



> plot (density (x))

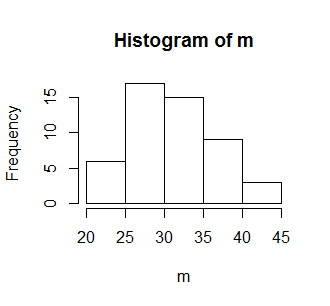


> hist (SAT, breaks = 40, col = "blue",xlab = "SAT")



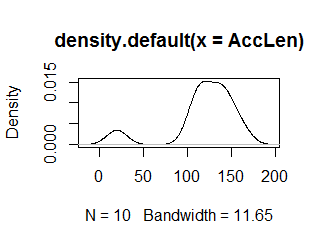
> hist(age, breaks=100, main="Accountholder age distribution", xlab="age", col="gray")

**/\* Histogram \*/**

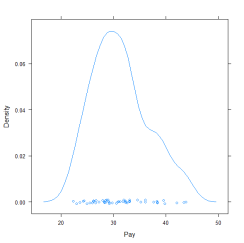
> data1 [1:50,3]-> m -- CREATING A VECTOR FROM THE TABLE

> m

> hist (m)



> plot (density (x)) -- density plot



**Use Lattice Package**

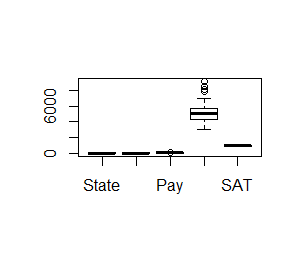
library(lattice)

densityplot (purchase\_size)

> plot (x) – plotting a graph

> plot (x,t)

> plot (<table name>$<column name>)

> plot (data1$Pay)

> Boxplot (data1) -- to see boxplots for each variable in the data set

> boxplot (x)

> boxplot (x,t)

> plot (x, t, type = "b")

> plot (x,t,type = "b", main = "My TITLE")

type

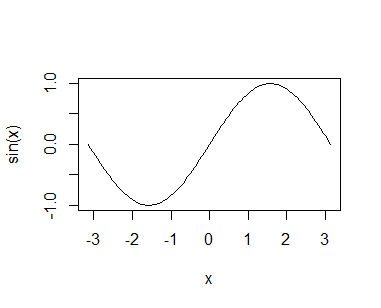
what type of plot should be drawn. Possible types are

* "p" for **p**oints,
* "l" for **l**ines,
* "b" for **b**oth,
* "c" for the lines part alone of "b",
* "o" for both ‘**o**verplotted’,
* "h" for ‘**h**istogram’ like (or ‘high-density’) vertical lines,
* "s" for stair **s**teps,
* "S" for other **s**teps, see ‘Details’ below,
* "n" for no plotting.
* main - an overall title for the plot: see [title](http://127.0.0.1:14511/library/graphics/help/title).
* sub - a sub title for the plot: see [title](http://127.0.0.1:14511/library/graphics/help/title).
* xlab - a title for the x axis: see [title](http://127.0.0.1:14511/library/graphics/help/title).
* ylab - a title for the y axis: see [title](http://127.0.0.1:14511/library/graphics/help/title).
* asp - the *y/x* aspect ratio, see [plot.window](http://127.0.0.1:14511/library/graphics/help/plot.window).

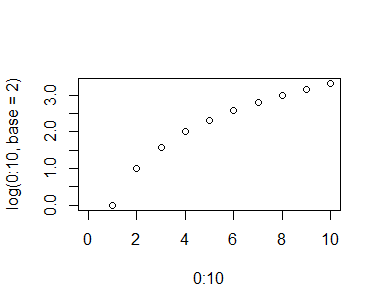
> plot (x=1:6, y=a) # plotting a graph

> points (3,6) # adding additional point to the graph

> plot (sin, -pi,pi) # adding function, starting and ending points



> plot (0:10, log (0:10, base = 2))



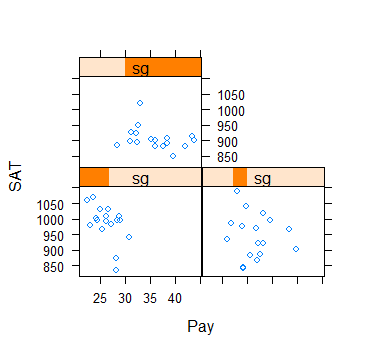
> plot (0:10, log (0:10, base = 2), type ="b")

**/\* uploading and using lattice package \*/**

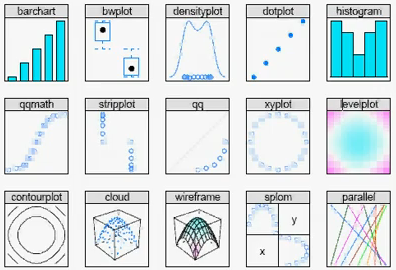
library (lattice) -- running the lattice packige

sg = equal.count (data1$Spending, number = 3, overlap = 0.1) -- splitting the Spending column on 3 sets

xyplot (SAT~Pay|sg, data = data1) -- plotting SAT on Pay scatter plot in regrads to three splittings

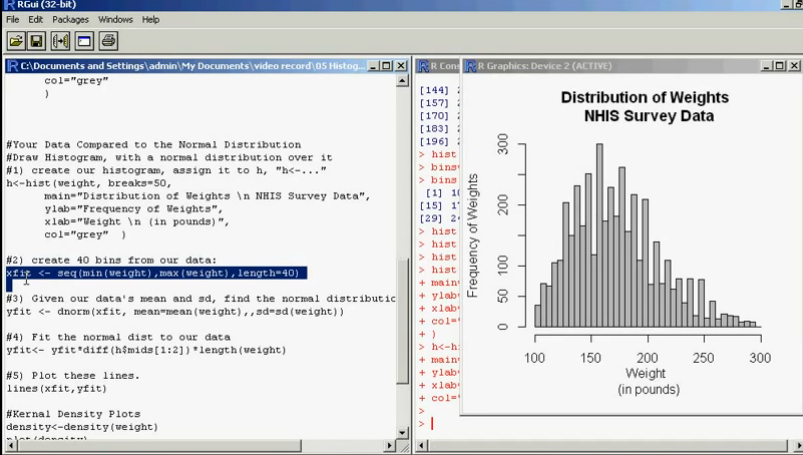


**Different Commands**



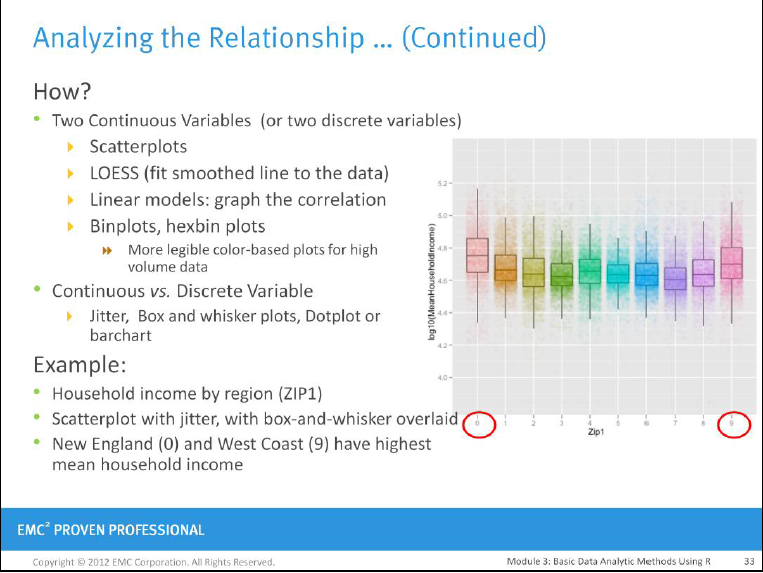
# Creating a Histogram in R Software (the hist() function)

<http://www.youtube.com/watch?v=wGW9M93YswY>



**5. Analyzing the relationship between two variables**

(pp. 135 of Amit’s doc)



**> pairs (new)** # analyzing all pairs at once

------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 1. Get the data into R (read the file)

\* Drug and Drop the file into the R to get the path

\* nrow = 10 -- will bring first 10 rows to R

**> SAT <- read.table ("C:\\RProjects\\mydata\\Telecom.txt", header = T, sep = ",", nrows = 10)**

**/\* Comma Separated File CSV\*/**

> **MY** <- read.table ("C:\\RProjects\\DataTest\\GiveMe.csv",header = T, sep = ",")

Where:

Sep = “,” – defining delimiter

Header = T – defining the first row as a header

Defining SAT as a table name and upload data there

> **SAT** <-read.table ("C:\\RProjects\\DataTest\\SAT.csv", sep = ",", header = T)

**Another Way**

> **T** <- read.csv ("C:\\RProjects\\DataTest\\SAT.csv", header = T) -- we do not need to use sep = “,” here

**/\* If a file is a Space Separated File \*/**

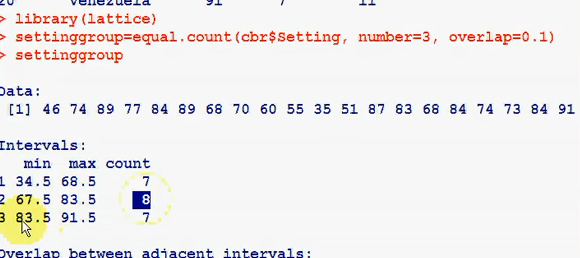
> **MY** <- read.table ("C:\\RProjects\\DataTest\\GiveMe.csv",header = T, sep = "") -- we do not use space in between “”

**/\* TAB Separated File\*/**

> **MY** <- read.table ("C:\\RProjects\\DataTest\\GiveMe.csv",header = T, sep = "\t")

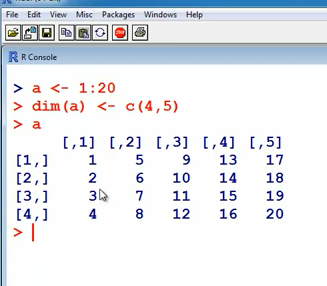
* Data pre-processing
  + Data cleaning
    - Identifying and correcting data types
    - Change variable types (recoding data)
    - Resolve inconsistencies
    - Identify and/or remove outliers
    - Deal with missing values
  + Data reduction
    - Combine variables to reduce data volume (too many variables)
    - Use sample data (too many rows)
  + Data transformation
    - Normalization and aggregation
* 4. Data visualization
  + Tables (summary, pivot)
  + One dimensional Graphs
    - Histograms, bar charts
  + Two dimensional Graphs
    - Scatter plots, mosaic charts, overlay plots
  + Higher dimensions
    - 3-D Scatter plots
    - Tree maps
    - Bubble plots
    - Principal components
* 5. Pattern Discovery
  + Clustering
  + Sequence analysis
* 6. Predictive Modeling
* 7. Model Comparison

--------------------------------------------------





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



a <- c (1:20) creates a vector of 10 elements numbered 1 through 10

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Merging Vectors**

> x <- 1:5

> y = 6:10

> x

[1] 1 2 3 4 5

> y

[1] 6 7 8 9 10

> z = c(x,y)

> z

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

> z = c (y,x)

> z

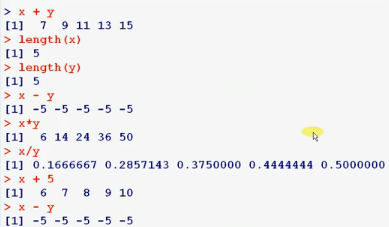
[1] 6 7 8 9 10 1 2 3 4 5 -- notice the difference when we change x and y

>

> length (x)

[1] 5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



> y

[1] 6 7 8 9 10

> y [2] -- shows the second element in the vector

[1] 7

> y [c(2,3)] – shows the second and third elements

[1] 7 8

> y [-c(2,3)] – shows all elements except for the second and third

[1] 6 9 10

>

**/\* Creating a Matrix \*/**

> a <- 1:20

> dim (a) <- c(4,5)

> a

[,1] [,2] [,3] [,4] [,5]

[1,] 1 5 9 13 17

[2,] 2 6 10 14 18

[3,] 3 7 11 15 19

[4,] 4 8 12 16 20

>

***Another way to create an array***

> a <- array (1:20, dim = c(4,5))

> a

[,1] [,2] [,3] [,4] [,5]

[1,] 1 5 9 13 17

[2,] 2 6 10 14 18

[3,] 3 7 11 15 19

[4,] 4 8 12 16 20

> a <- array (1, dim = c(4,5))

> a

[,1] [,2] [,3] [,4] [,5]

[1,] 1 1 1 1 1

[2,] 1 1 1 1 1

[3,] 1 1 1 1 1

[4,] 1 1 1 1 1

>

***Transpose an array to another array***

> b <- t (a)

> diag (a) # get a diagonal values from the matrix

> eigen (b) # eigen values. Works only for square matrixes

$values

[1] 3.620937e+01 -2.209373e+00 1.599839e-15 7.166935e-16

$vectors

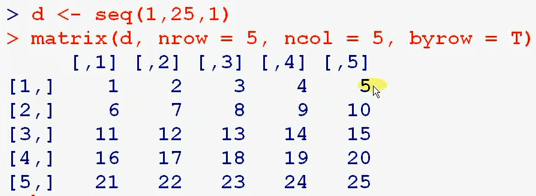
[,1] [,2] [,3] [,4]

[1,] 0.4140028 0.82289268 -0.5477226 0.1125155

[2,] 0.4688206 0.42193991 0.7302967 0.2495210

[3,] 0.5236384 0.02098714 0.1825742 -0.8365883

[4,] 0.5784562 -0.37996563 -0.3651484 0.4745519



> d <- seq (1,25,1)

> matrix (d, nrow = 5, ncol = 5, byrow = T)

[,1] [,2] [,3] [,4] [,5]

[1,] 1 2 3 4 5

[2,] 6 7 8 9 10

[3,] 11 12 13 14 15

[4,] 16 17 18 19 20

[5,] 21 22 23 24 25

> matrix (d, nrow = 5, ncol = 5, byrow = F)

[,1] [,2] [,3] [,4] [,5]

[1,] 1 6 11 16 21

[2,] 2 7 12 17 22

[3,] 3 8 13 18 23

[4,] 4 9 14 19 24

[5,] 5 10 15 20 25

>

> matrix (d, nrow = 5, ncol = 5, byrow = F)**->m**

> m

[,1] [,2] [,3] [,4] [,5]

[1,] 1 6 11 16 21

[2,] 2 7 12 17 22

[3,] 3 8 13 18 23

[4,] 4 9 14 19 24

[5,] 5 10 15 20 25

>

> m [1,]

[1] 1 6 11 16 21

> m [2,2]

[1] 7

> m [2,1:3]

[1] 2 7 12

> m [4,c(1,3,5)]

[1] 4 14 24

> m [4,-c(1,3,5)]

[1] 9 19

***Creating random values:***

> x <- runif (10, 1,9)

> x

[1] 3.928440 3.979434 7.056013 4.538397 3.680303 1.253241 5.942669

[8] 1.919275 4.397525 4.657055

***Rounding values:***

> round (x,0)

[1] 4 4 7 5 4 1 6 2 4 5

> x <- round (x,0)

> x

[1] 4 4 7 5 4 1 6 2 4 5

> dim (x) <- c (2,5)

> x

[,1] [,2] [,3] [,4] [,5]

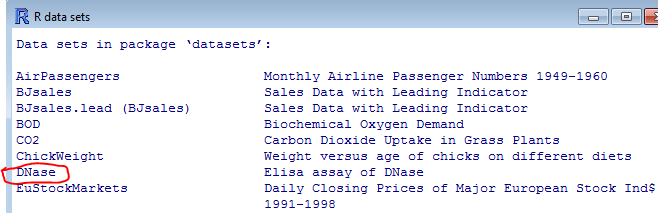
[1,] 4 7 4 6 4

[2,] 4 5 1 2 5

>

> data () -- some of internal datasets in R

> data (DNase)



**/\* Saving data file \*/**

> datafile = DNase

> sink ("c:\\111\\out1.txt")

> datafile

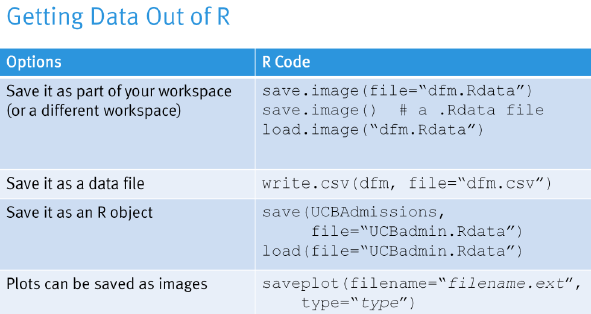
> sink ()

**/\* write txt file \*/**

> write.table (datafile, file = "c:\\111\\out2.txt")

**/\* write csv file table \*/**

> write.csv (datafile, file = "c:\\111\\out3.csv")



**> write.csv (SAT, file = "mysas1.csv")**