Introduction to R

David Walling
Data Group
Texas Advanced Computing Center
walling@tacc.utexas.edu

R-project background

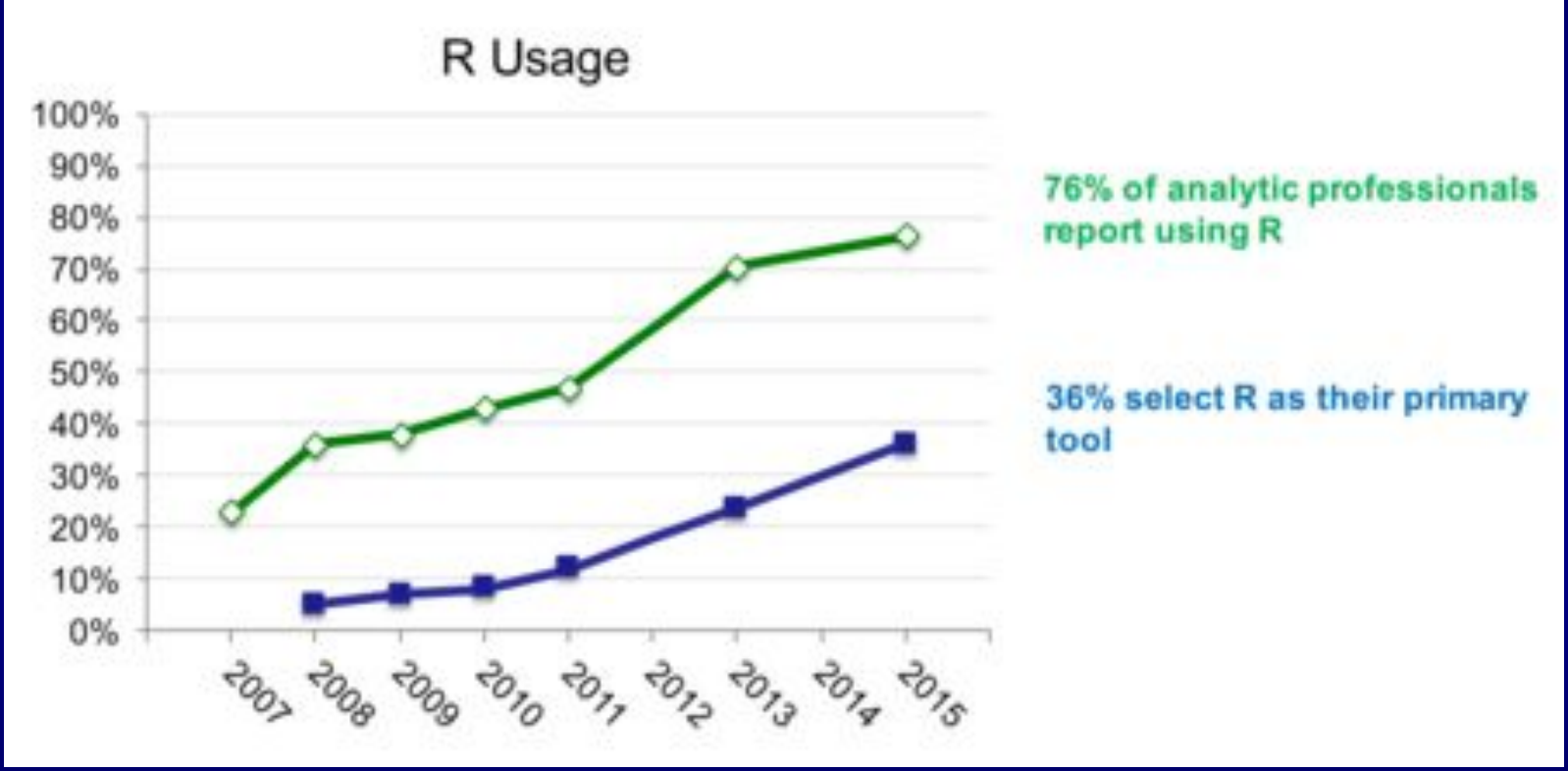
- Origin and History
  - Initially written by Ross Ihaka and Robert Gentleman at Dep. of Statistics of U of Auckland, New Zealand during 1990s.
  - An offspring of S: Bell Labs, interactive Fortran/C
  - International project since 1997/Beta release 200
  - Explosive growth last 10 years
- Open source with GPL license
  - Free as in beer
  - In active development *
  - http://www.r-project.org/

What R does

R is a programming environment for statistical and data analysis computations.

- Core Package
  - Statistical functions
  - Plotting and graphics
  - Data handling and storage
    - predefined data reader
    - textual, regular expressions
    - hashing
  - Data analysis functions
  - Programming support:
    - loops, branching, subroutines
    - Object Oriented
- Extensive community contributed packages.

R's Popularity



http://www.r-bloggers.com/new-surveys-show-continued-popularity-of-r/

R command line interface on cluster

```
login1$ R

R version 2.15.1 (2012-06-22) -- "Roasted Marshmallows"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-unknown-linux-gnu (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.

>
```

RStudio: A better user interface of R

- RStudio is an open source graphical user environment for R users.
  - https://www.rstudio.com/
- RStudio allow users to
  - Interactive code development
  - Run R scripts
  - Exploring local file system
  - Viewing data file
  - Viewing graphical output from R
  - …

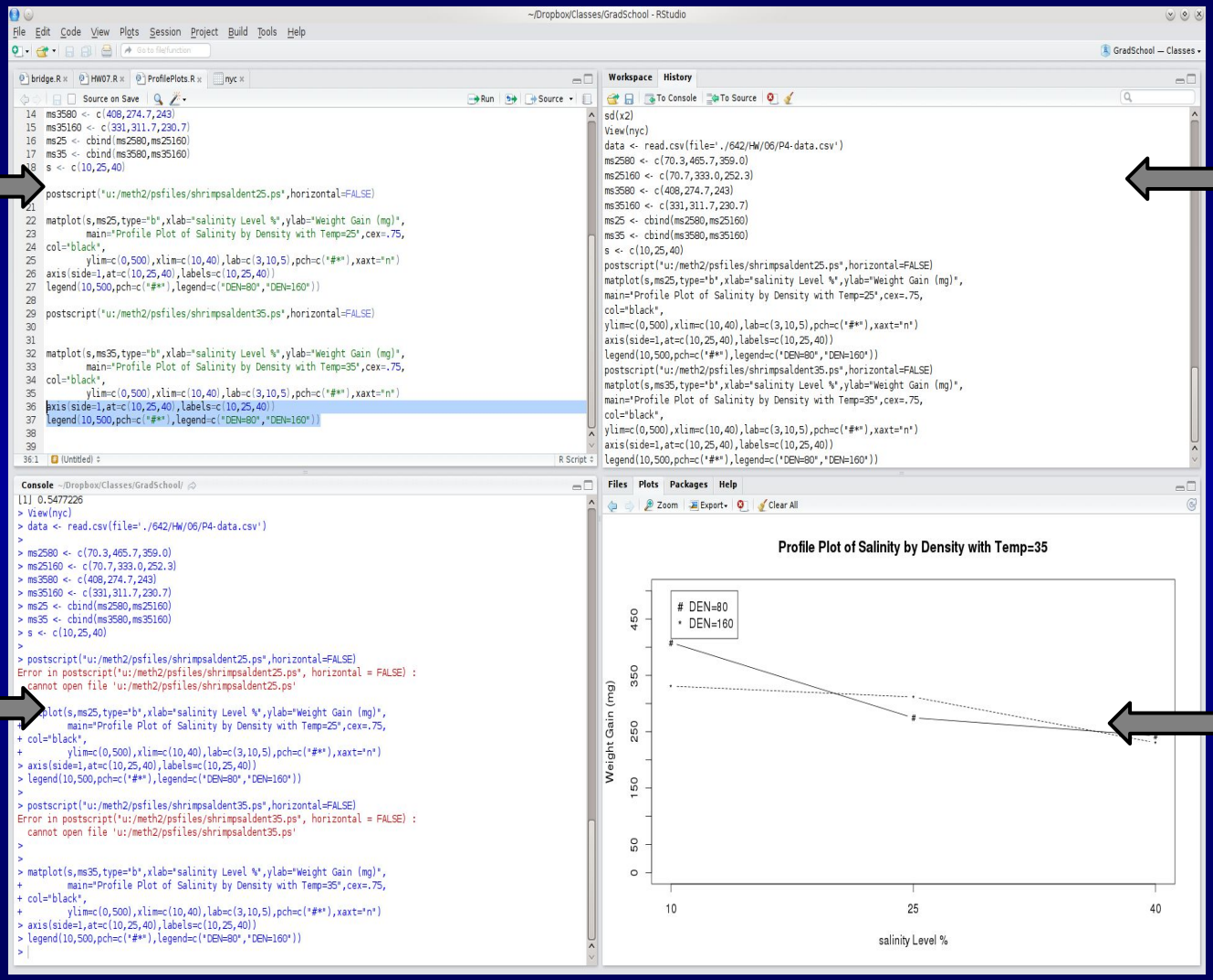
RStudio GUI

Source Editor + Interactive highlight+run

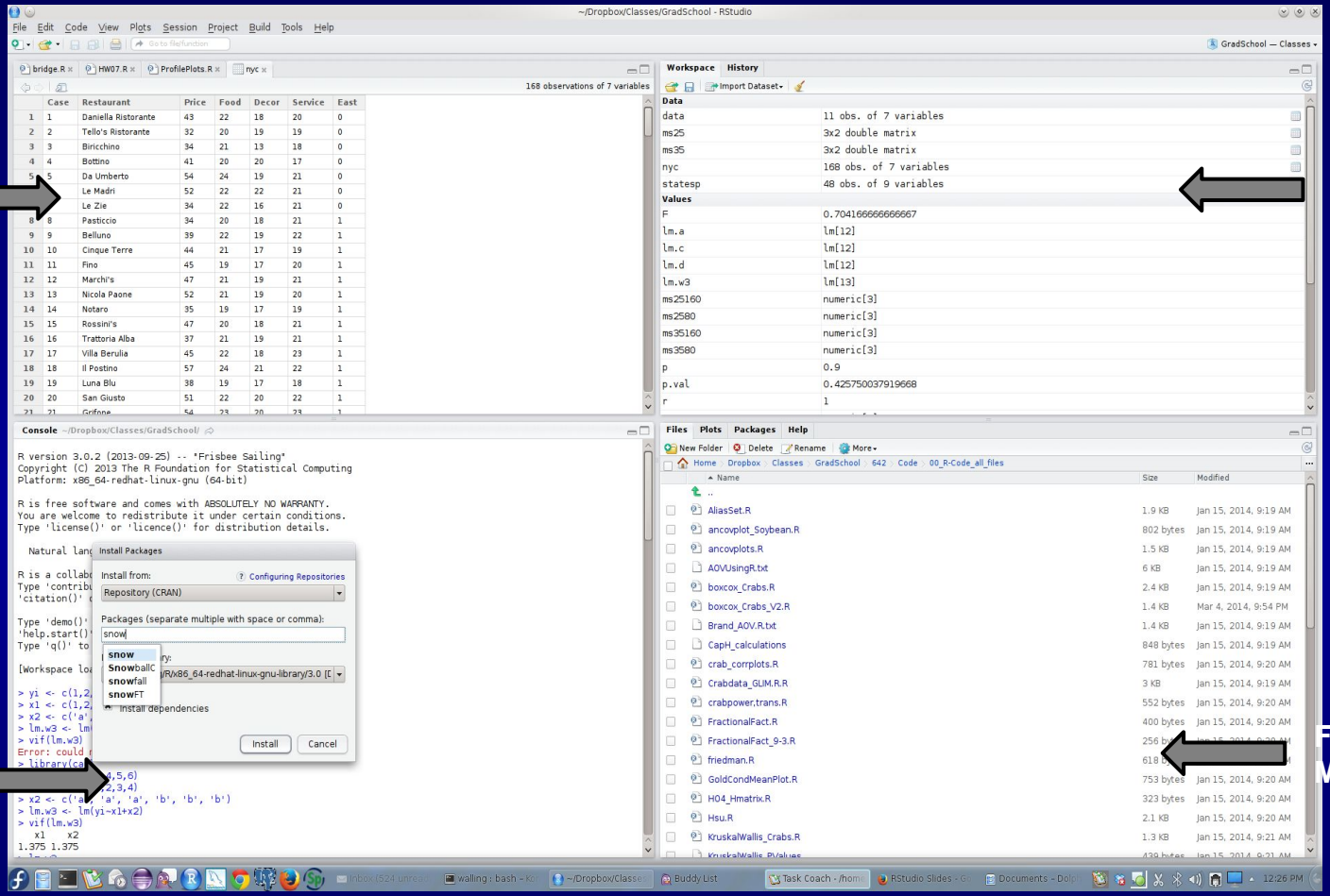
Interactive Console

Session History

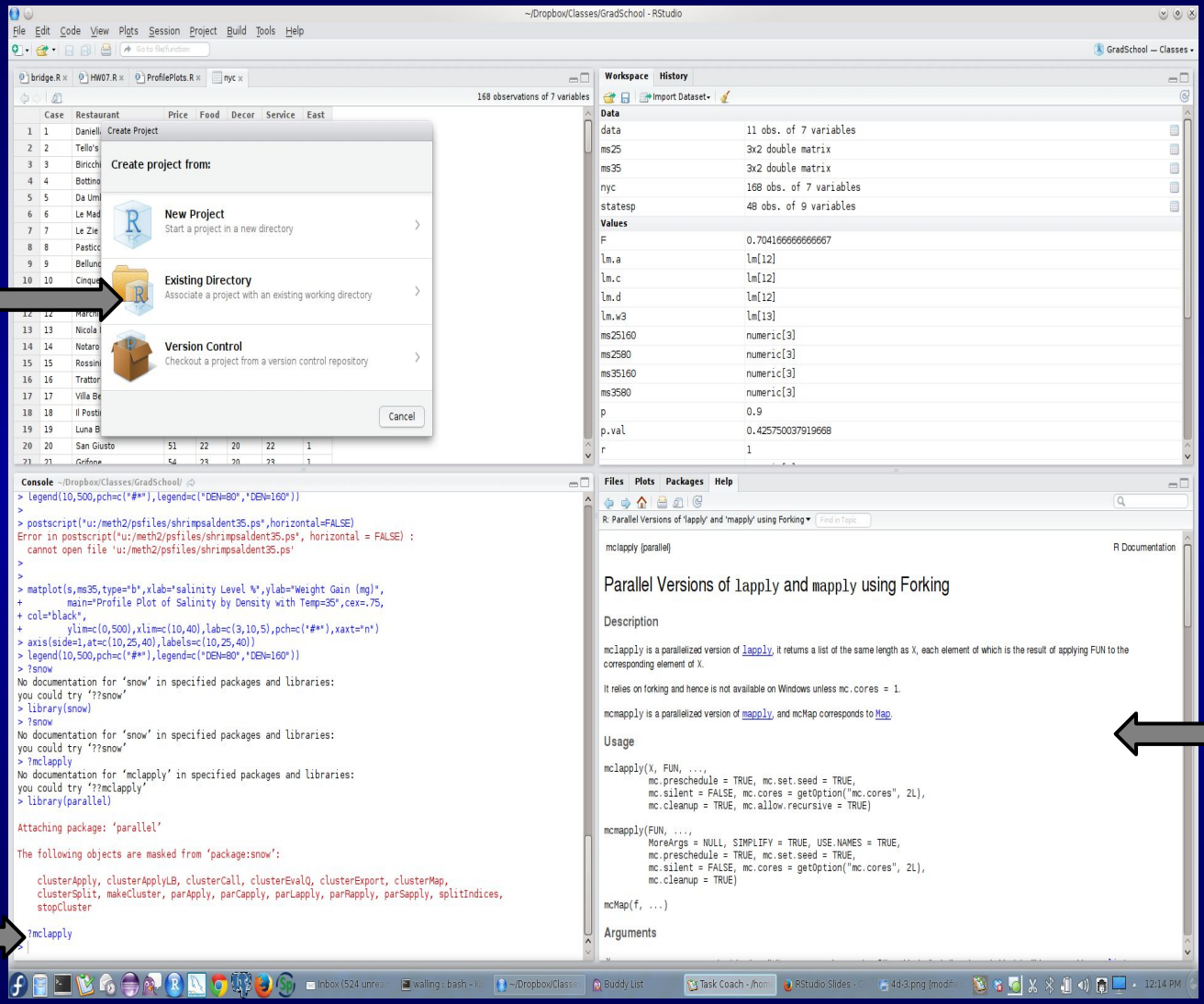
Plots Management



RStudio GUI


Data Viewer
Variable/Environment Viewer
Package Management
File Management

RStudio GUI



Project + Source Control Integration

Interactive Help

Interactive Help

Getting help

- **"?" Or "help"**
  Details about a specific command whose name you know (input arguments, options, algorithm, results):
  e.g.
  >? t.test

  or

  >help(t.test)

```
t.test                  package:ctest                  R Documentation

Student's t-Test

Description:

     Performs one and two sample t-tests on vectors of data.

Usage:

     t.test(x, y = NULL, alternative = c("two.sided", "less", "greater"),
            mu = 0, paired = FALSE, var.equal = FALSE,
            conf.level = 0.95, ...)
     t.test(formula, data, subset, na.action, ...)

Arguments:

       x: a numeric vector of data values.

       y: an optional numeric vector data values.

alternative: a character string specifying the alternative hypothesis,
          must be one of `"two.sided"' (default), `"greater"' or
          `"less"'.  You can specify just the initial letter.

      mu: a number indicating the true value of the mean (or difference
          in means if you are performing a two sample test).

  paired: a logical indicating whether you want a paired t-test.

var.equal: a logical variable indicating whether to treat the two
```

Basic Syntax

Math Operations

- R as a calculator
  - +, -, /, *, ^, log, exp, …

```
> (17*0.35)^(1/3)
[1] 1.812059
> log2(128)
[1] 7
> exp(1)
[1] 2.718282
> 3^-1
[1] 0.3333333
```

Variables

- Numeric

```
> a=49
> a
[1] 49
```

- Character String

```
> b="this is a string"
> b
[1] "this is a string"
```

- Logical

```
> c=(1+1==3)
> c
[1] FALSE
```

Assigning Values to Variables

- "<-" or "="

```
> a=4
> a
[1] 4
> a<-40
> a
[1] 40
```

- As[illegible]ple values
  - Combine, c()
  - Stdin, scan()
  - Series, seq()

```
> a=c(1, 2, 4, 7, 9)
> a
[1] 1 2 4 7 9
```

```
> a=scan()
1: 9
2: 7
3: 4
4: 2
5: 1
6:
Read 5 items
> a
[1] 9 7 4 2 1
```

```
> a=(1:6)
> a
[1] 1 2 3 4 5 6
```

```
> a=seq(1,6,0.5)
> a
 [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
```

NA: Missing Value

- **Variables of each data type (numeric, character, logical) can also take the value NA: not available.**
  - **NA is not the same as 0**
  - **NA is not the same as “”**
  - **NA is not the same as FALSE**

- **Any operations (calculations, comparisons) that involve NA may or may not produce NA:**

```
> NA
[1] NA
> 1+NA
[1] NA
> log(NA)
[1] NA
```

```
> NA | TRUE
[1] TRUE
>  NA | FALSE
[1] NA
>  NA & TRUE
[1] NA
> NA & FALSE
[1] FALSE
```

```
>  max(c(1,2,3, NA))
[1] NA
>  max(c(1,2,3,NA), na.rm=T)
[1] 3
```

Basic Data Structure

- Vector
  - an ordered collection of data of the same type
  - a single number is the special case of a vector with 1 element.
  - Usually accessed by index

```
> # Vectors
> a = c(1,2,3)
> a
[1] 1 2 3
> a[2]
[1] 2
> a*2
[1] 2 4 6
> max(a)
[1] 3
> sum(a)
[1] 6
```

Basic Data Structure

- Matrix
  - Rows, Columns
  - Single data type
  - Linear algebra computations

```
> A = matrix(c(1,2,3,4,5,6,7,8,9), nrow=3)
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> A[2,]
[1] 2 5 8
> x = c(1,2,3)
> A * x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    4   10   16
[3,]    9   18   27
> A %*% x
     [,1]
[1,]   30
[2,]   36
[3,]   42
```

Exercise

1. Create a vector of integers from 1 to 2016 and sum the result.

Exercise

1. Create a vector of integers from 1 to 2016 and sum the result.

```
> # 1
> a = (1:2016)
> sum(a)
[1] 2033136
```

Exercise

1. Create a vector of integers from 1 to 2016 and sum the result.

```
> # 1
> a = (1:2016)
> sum(a)
[1] 2033136
```

2*. Create a 3x3 matrix of i^3 for i = 1 to 9, by row. Sum the 3rd row.

Exercise

1. Create a vector of integers from 1 to 2016 and sum the result.

```
> # 1
> a = (1:2016)
> sum(a)
[1] 2033136
```

2*. Create a 3x3 matrix of i^3 for i = 1 to 9. Sum the 3rd row.

```
> i = (1:9)
> A = matrix(i^3, nrow=3, byrow=TRUE)
> sum(A[3,])
[1] 1584
```

Basic Data Structure

- List
  - an ordered collection of data of arbitrary types.
  - key-value pair
  - Accessible by key

```
> doe = list(name="john", age=28, married=F)
> doe$name
[1] "john"
> doe$age
[1] 28
> doe$married
[1] FALSE
> doe[1]
$name
[1] "john"
```

Dataframes

- R handles data in objects known as dataframes;
  - rows: data items;
  - columns: values of the different attributes
    - Values in each column should be from the same type.

	Area	Slope	Vegetation	Soil.pH	Damp	Worm.density
Silwood.Bottom	5.1	2	Arable	5.2	FALSE	7
Gunness.Thicket	3.8	0	Scrub	4.2	FALSE	6
Oak.Mead	3.1	2	Grassland	3.9	FALSE	2
North.Gravel	3.3	1	Grassland	4.1	FALSE	1
South.Gravel	3.7	2	Grassland	4.0	FALSE	2
Pond.Field	4.1	0	Meadow	5.0	TRUE	6
Water.Meadow	3.9	0	Meadow	4.9	TRUE	8
Pound.Hill	4.4	2	Arable	4.5	FALSE	5

Built In Data Sets

- R provides many pre-installed data sets
- data()
- data(mtcars)

mtcars {datasets} R Documentation

Motor Trend Car Road Tests

Description

The data was extracted from the 1974 *Motor Trend* US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

Usage

```
mtcars
```

Format

A data frame with 32 observations on 11 variables.

[, 1] mpg	Miles/(US) gallon
[, 2] cyl	Number of cylinders
[, 3] disp	Displacement (cu.in.)
[, 4] hp	Gross horsepower
[, 5] drat	Rear axle ratio
[, 6] wt	Weight (1000 lbs)
[, 7] qsec	1/4 mile time
[, 8] vs	V/S
[, 9] am	Transmission (0 = automatic, 1 = manual)
[,10] gear	Number of forward gears
[,11] carb	Number of carburetors

Read Dataframes From File

- read.table()

the first column contains data label

```
> worms<-read.table("worms.txt",header=T,row.names=1)
```

path: in double quotes

the first row contains the variables names

  - Read tab-delimited file directly.
  - Variable name in header row cannot have space.

- To see the content of the dataframes (object) just type is name:

```
> worms
```

Selecting Data from Dataframes

- Subscripts within square brackets
  - `[,` means “all the rows” and
  - `,]` means “all the columns”

- To select the first three column of the dataframe

```
> worms[,1:3]
                 Area Slope Vegetation
Silwood.Bottom    5.1     2     Arable
Gunness.Thicket   3.8     0      Scrub
Oak.Mead          3.1     2  Grassland
North.Gravel      3.3     1  Grassland
South.Gravel      3.7     2  Grassland
Pond.Field        4.1     0     Meadow
Water.Meadow      3.9     0     Meadow
Pound.Hill        4.4     2     Arable
```

Selecting Data from Dataframes

- names()
  - Get a list of variables attached to the input name

```
> names(worms)
[1] "Area"          "Slope"          "Vegetation"
[4] "Soil.pH"       "Damp"           "Worm.density"
```

- attach()
  - Make the variables accessible by name:

```
> attach(worms)
```

Selecting Data from Dataframes

- Using logic expression while selecting:

	Area	Slope	Vegetation	Soil.pH	Damp	Worm.density
Silwood.Bottom	5.1	2	Arable	5.2	FALSE	7
Gunness.Thicket	3.8	0	Scrub	4.2	FALSE	6
Oak.Mead	3.1	2	Grassland	3.9	FALSE	2
North.Gravel	3.3	1	Grassland	4.1	FALSE	1
South.Gravel	3.7	2	Grassland	4.0	FALSE	2
Pond.Field	4.1	0	Meadow	5.0	TRUE	6
Water.Meadow	3.9	0	Meadow	4.9	TRUE	8
Pound.Hill	4.4	2	Arable	4.5	FALSE	5

```
> worms[Area>4&Slope<1,]
           Area Slope Vegetation Soil.pH Damp Worm.density
Pond.Field  4.1     0     Meadow       5 TRUE            6
```

Selecting Data From a Dataframe

More examples:

```
> worms[Damp,]
            Area Slope Vegetation Soil.pH Damp Worm.density
Pond.Field   4.1     0     Meadow     5.0 TRUE            6
Water.Meadow 3.9     0     Meadow     4.9 TRUE            8
> worms$Vegetation
[1] Arable    Scrub     Grassland Grassland Grassland Meadow    Meadow
[8] Arable
Levels: Arable Grassland Meadow Scrub
> worms$Vegetation=="Grassland"
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
> worms[ worms$Vegetation=="Grassland",]
            Area Slope Vegetation Soil.pH  Damp Worm.density
Oak.Mead     3.1     2  Grassland     3.9 FALSE            2
North.Gravel 3.3     1  Grassland     4.1 FALSE            1
South.Gravel 3.7     2  Grassland     4.0 FALSE            2
```

subset rows by a logical vector

subset a column

comparison resulting in logical vector

subset the selected rows

Sorting Data in Data frames

- order()

State the Area for sorting order

State columns to be sorted

```
> worms[order(worms[,1]),1:6]
                Area Slope Vegetation Soil.pH  Damp Worm.density
Oak.Mead         3.1     2  Grassland     3.9 FALSE            2
North.Gravel     3.3     1  Grassland     4.1 FALSE            1
South.Gravel     3.7     2  Grassland     4.0 FALSE            2
Gunness.Thicket  3.8     0      Scrub     4.2 FALSE            6
Water.Meadow     3.9     0     Meadow     4.9  TRUE            8
Pond.Field       4.1     0     Meadow     5.0  TRUE            6
Pound.Hill       4.4     2     Arable     4.5 FALSE            5
Silwood.Bottom   5.1     2     Arable     5.2 FALSE            7
```

Sorting Data in Dataframes

- More on sorting selected

sorted in descending order

```
> worms[rev(order(worms[,4])),c(4,6)]
                 Soil.pH Worm.density
Silwood.Bottom       5.2            7
Pond.Field           5.0            6
Water.Meadow         4.9            8
Pound.Hill           4.5            5
Gunness.Thicket      4.2            6
North.Gravel         4.1            1
South.Gravel         4.0            2
Oak.Mead             3.9            2
```

str()

- str() provides details on a particular data structure.

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
>
```

Exercise

3. Load ‘iris’ data set included with R. Find max sepal length.

Exercise

3. Load ‘iris’ data set. Find max sepal length.

```
> # 3
> data(iris)
> max(iris$Sepal.Length)
[1] 7.9
```

Exercise

3. Load ‘iris’ data set. Find max sepal length.

```
> # 3
> data(iris)
> max(iris$Sepal.Length)
[1] 7.9
```

4. Count number of ‘virginica’ species entries.

Exercise

3. Load ‘iris’ data set. Find max sepal length.

```
> # 3
> data(iris)
> max(iris$Sepal.Length)
[1] 7.9
```

4*. How many ‘virginica’ species entries?

```
> # 4
> dim(iris[iris$Species=='virginica', ])
[1] 50  5
```

Flow Control

- If … else

```
if (logical expression) {
  statements
} else {
  alternative statements
}
```

*** else branch is optional**

- loops

```
for(i in 1:10) {
   print(i*i)
}
```

```
i=1
while(i<=10) {
   print(i*i)
   i=i+sqrt(i)
}
```

Flow Control

- apply `(arr, margin, fct )`
  - Applies the function `fct` along some dimensions of the vector/matrix `arr`, according to `margin`, and returns a vector or array of the appropriate size.

```
> m
                Soil.pH Worm.density
Silwood.Bottom      5.2             7
Pond.Field          5.0             6
Water.Meadow        4.9             8
Pound.Hill          4.5             5
Gunness.Thicket     4.2             6
North.Gravel        4.1             1
South.Gravel        4.0             2
Oak.Mead            3.9             2
> apply(m, 1, sum)
 Silwood.Bottom      Pond.Field    Water.Meadow      Pound.Hill Gunness.Thicket
           12.2            11.0            12.9             9.5            10.2
   North.Gravel    South.Gravel       Oak.Mead
            5.1             6.0             5.9
> apply(m, 2, sum)
     Soil.pH Worm.density
        35.8         37.0
```

Flow Control

- lapply (li, fct) and sapply (li, fct)
  - To each element of the list `li`, the function `fct` is applied. The result is a list whose elements are the individual `fct` results.
  - Sapply, converting results into a vector or array of appropriate size if possible

```
> fct = function(x) { return(c(x, x*x, x*x*x)) }
> sapply(1:5, fct)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    1    4    9   16   25
[3,]    1    8   27   64  125
```

```
> lapply(1:5, fct)
[[1]]
[1] 1 1 1

[[2]]
[1] 2 4 8

[[3]]
[1]  3  9 27

[[4]]
[1]  4 16 64

[[5]]
[1]   5  25 125
```

Exercise

5*. Create vector: idx <- c(18, 27, 9, 19, 27, 1, 23, 5, 19, 15, 13, 5). Use ‘sapply’ to return the letter at each index in the included ‘letters’ data structure.

Exercise

5*. Create vector: idx <- c(18, 27, 9, 19, 27, 1, 23, 5, 19, 15, 13, 5). Use ‘sapply’ to return the letter at each index in the included ‘letters’ data structure.

```
> idx <- c(18, 27, 9, 19, 27, 1, 23, 5, 19, 15, 13, 5)
> sapply(idx, function(i) { letters[i]})
 [1] "r" NA  "i" "s" NA  "a" "w" "e" "s" "o" "m" "e"
```

Create Statistical Summary

- Descriptive summary for numerical variables:
  - arithmetic mean;
  - maximum, minimum, median, 25 and 75 percentiles (first and third quartile);
- Levels of categorical variables are counted

```
> summary(worms)
      Area             Slope          Vegetation    Soil.pH          Damp          Worm.density
 Min.   :3.100   Min.   :0.000   Arable   :2   Min.   :3.900   Mode :logical   Min.   :1.000
 1st Qu.:3.600   1st Qu.:0.000   Grassland:3   1st Qu.:4.075   FALSE:6         1st Qu.:2.000
 Median :3.850   Median :1.500   Meadow   :2   Median :4.350   TRUE :2         Median :5.500
 Mean   :3.925   Mean   :1.125   Scrub    :1   Mean   :4.475   NA's :0         Mean   :4.625
 3rd Qu.:4.175   3rd Qu.:2.000                 3rd Qu.:4.925                   3rd Qu.:6.250
 Max.   :5.100   Max.   :2.000                 Max.   :5.200                   Max.   :8.000
```

Exercise

6. What is the 1st quantile for sepal width in the ‘iris’ data set.

Exercise

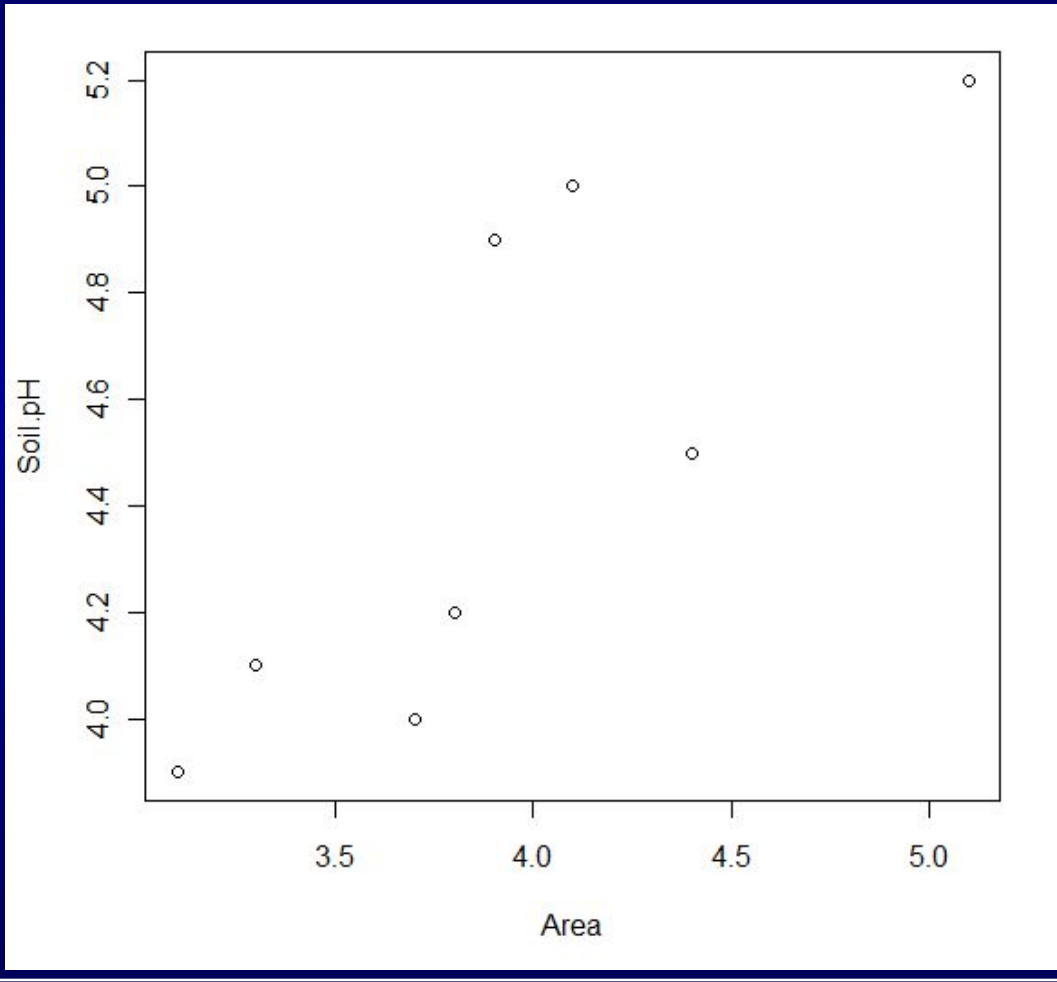
6. What is the 1st quantile for sepal width in the ‘iris’ data set.

```
> # 5
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> quantile(iris$Sepal.Width)
  0%  25%  50%  75% 100%
 2.0  2.8  3.0  3.3  4.4
```

Create Plots

- plot(…)
  - Create scatter plot.

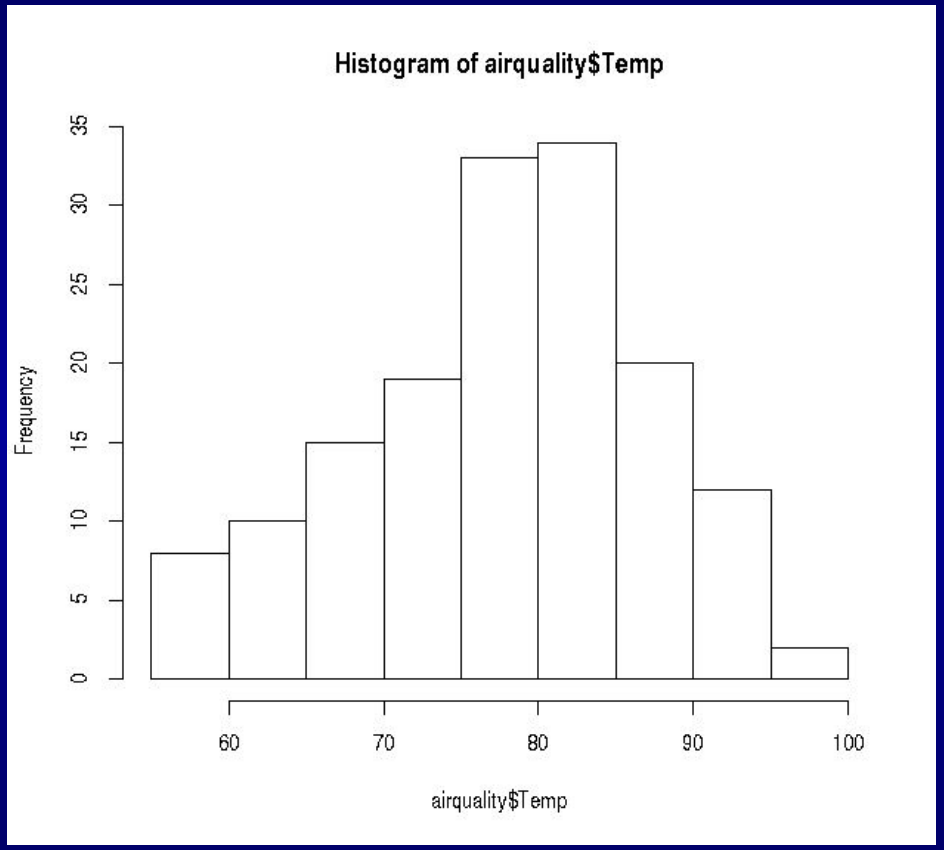
> plot(Area, Soil.pH)



Create Plots

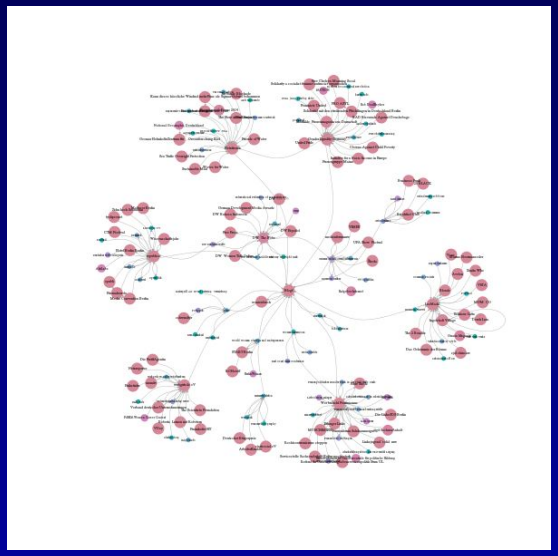
- Histograms: visual frequency distributions

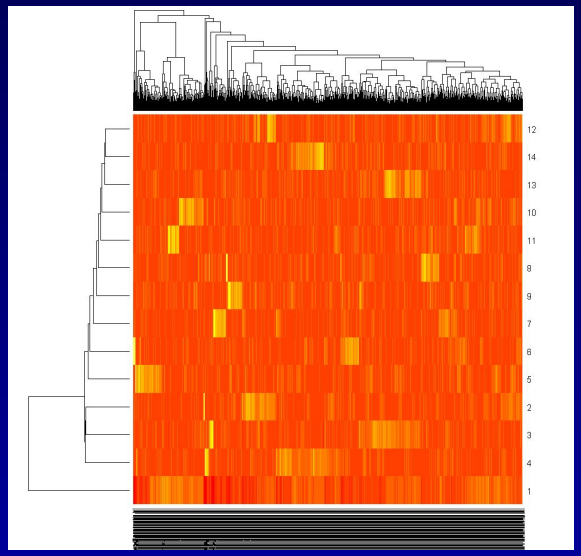
```
> hist(airquality$Temp)
```

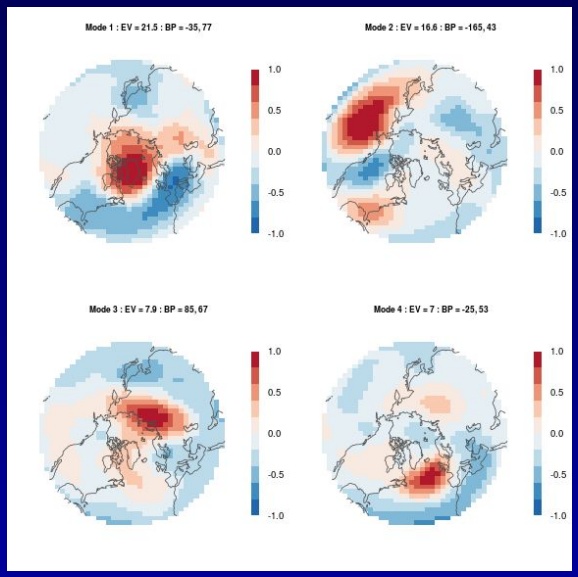


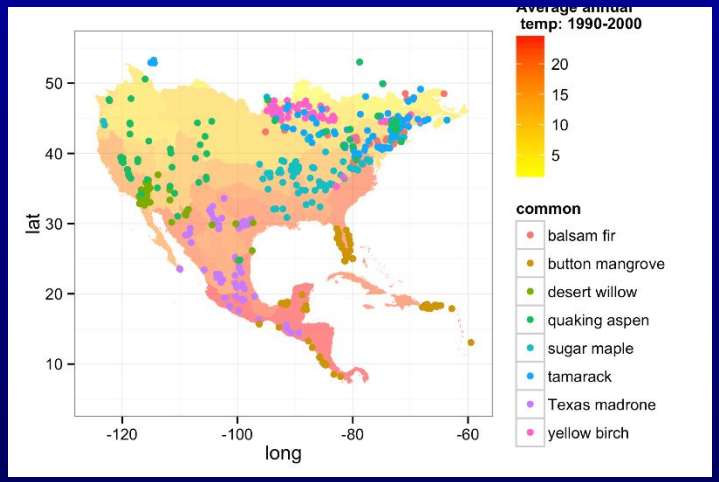
Other Common Plots

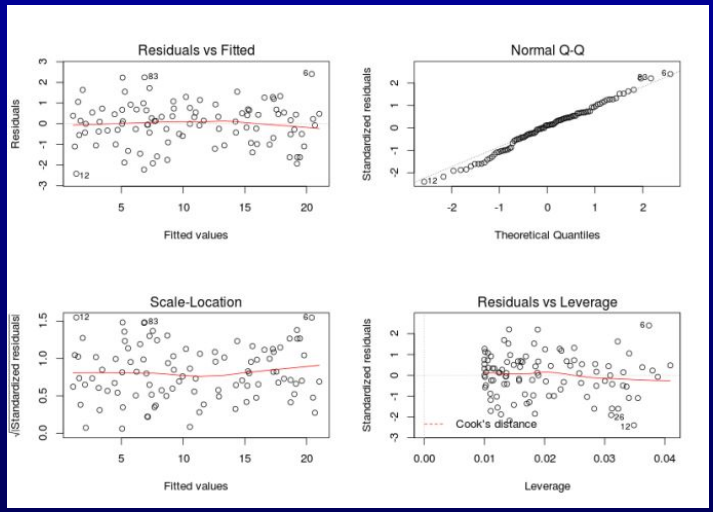
- Univariate:
  - histograms,
  - density curves,
  - Boxplots, quantile-quantile plots
- Bivariate:
  - scatter plots with trend lines,
  - side-by-side boxplots
- Several variables:
  - scatter plot matrices, lattice
  - 3-dimensional plots,
  - heatmap











TACC

XSEDE
Extreme Science and Engineering Discovery Environment

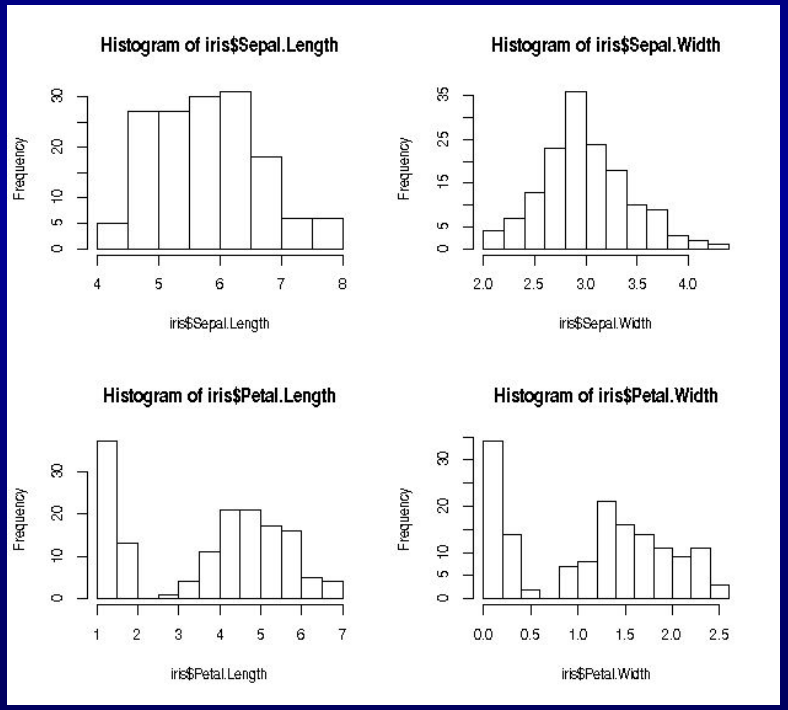
Exercise

7*. Using ‘hist’ plot function, which of the iris numeric variables (Sepal/Petal Width/Length) has the ‘most’ normal-like distribution (i.e. bell curve)?

Exercise

7*. Using ‘hist’ plot function, which of the iris numeric variables (Sepal/Petal Width/Length) has the ‘most’ normal-like distribution (i.e. bell curve)?

```
> par(mfrow=c(2,2))
> hist(iris$Sepal.Length)
> hist(iris$Sepal.Width)
> hist(iris$Petal.Length)
> hist(iris$Petal.Width)
```



Importing and exporting data

There are many ways to get data into R and out of R.

Most programs (e.g. Excel), as well as humans, know how to deal with rectangular tables in the form of tab-delimited text files.

> x = read.delim("filename.txt")
also: read.table, read.csv

> write.table(x, file="x.txt", sep="\t")

Saving your work

- `history(Inf)`
  - To review the command lines entered during the sessions
- `savehistory("history.txt")`
  - Save the history of command lines to a text file
- `loadhistory("history.txt")`
  - read it back into R
- `save(list=ls(),file="all.Rdata")`
  - The session as a whole can be saved as a binary file.
- `load("c:\\temp\\ all.Rdata")`
  - Read back saved sessions.

Additional Libraries and Packages

- Libraries
  - Comes with Package installation (Core or others)
  - library() shows a list of current installed
  - library must be loaded before use e.g.
    - library(rpart)
- Packages
  - Developed code/libraries outside the core packages
  - Can be downloaded and installed separately
    - install.packages("name")
  - There are currently 10k+ packages at http://cran.r-project.org/web/packages/
    - E.g. Rweka, interface to Weka.
    - ggplot2: very popular for 'building up' plots

Installing Packages on TACC Systems

- **R handles package dependencies for you.**
- **Many packages compile C/Fortan.**
- **In some cases, additional libraries required.**
  - *libXXX.so not found*.
  - **Submit a consulting ticket.**

Installing Packages on TACC Systems

```
> install.packages('FrF2')
Warning in install.packages("FrF2") :
  'lib =
"/opt/apps/intel14/mvapich2_2_0/Rstats/3.0.3/lib64/R/li
brary"' is not writable
Would you like to use a personal library instead?
(y/n) y
Would you like to create a personal library
~/R/x86_64-unknown-linux-gnu-library/3.0
to install packages into?  (y/n) y
```

```
mpicc -std=gnu99 -fPIC -openmp -mkl=parallel -O3 -xHost
-L/opt/apps/intel/13/composer_xe_2013_sp1.1.106/mkl/lib/in
tel64 -lmkl_rt -shared -fPIC -openmp -mkl=parallel -O3
-xHost
-L/opt/apps/intel/13/composer_xe_2013_sp1.1.106/mkl/lib/in
tel64 -lmkl_rt -o BsMD.so bsmd.o -lmkl_intel_lp64
-lmkl_intel_thread -lmkl_core -liomp5 -lmkl_rt -lifport
-lifcoremt -limf -lsvml -lm -lipgo -lirc -lpthread -lirc_s -ldl
-L/opt/apps/intel14/mvapich2_2_0/Rstats/3.0.3/lib64/R/lib -lR
```

TACC

XSEDE
Extreme Science and Engineering Discovery Environment

Further references

- R manual:
  - http://cran.r-project.org/manuals.html
- Topic Views
  - https://cran.r-project.org/web/views/
- Community Blogs
  - **http://r-bloggers.com**

David Walling

walling@tacc.utexas.edu