

Scientific and Technical Computing

Introduction to R

© The University of Texas at Austin, 2014
Please see the final slide for copyright and licensing information



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
 - International project since 1997
 - Built by statisticians for statisticians
- Open source with GPL license
 - Free to anyone
 - In active development *
 - <http://www.r-project.org/>

What R does

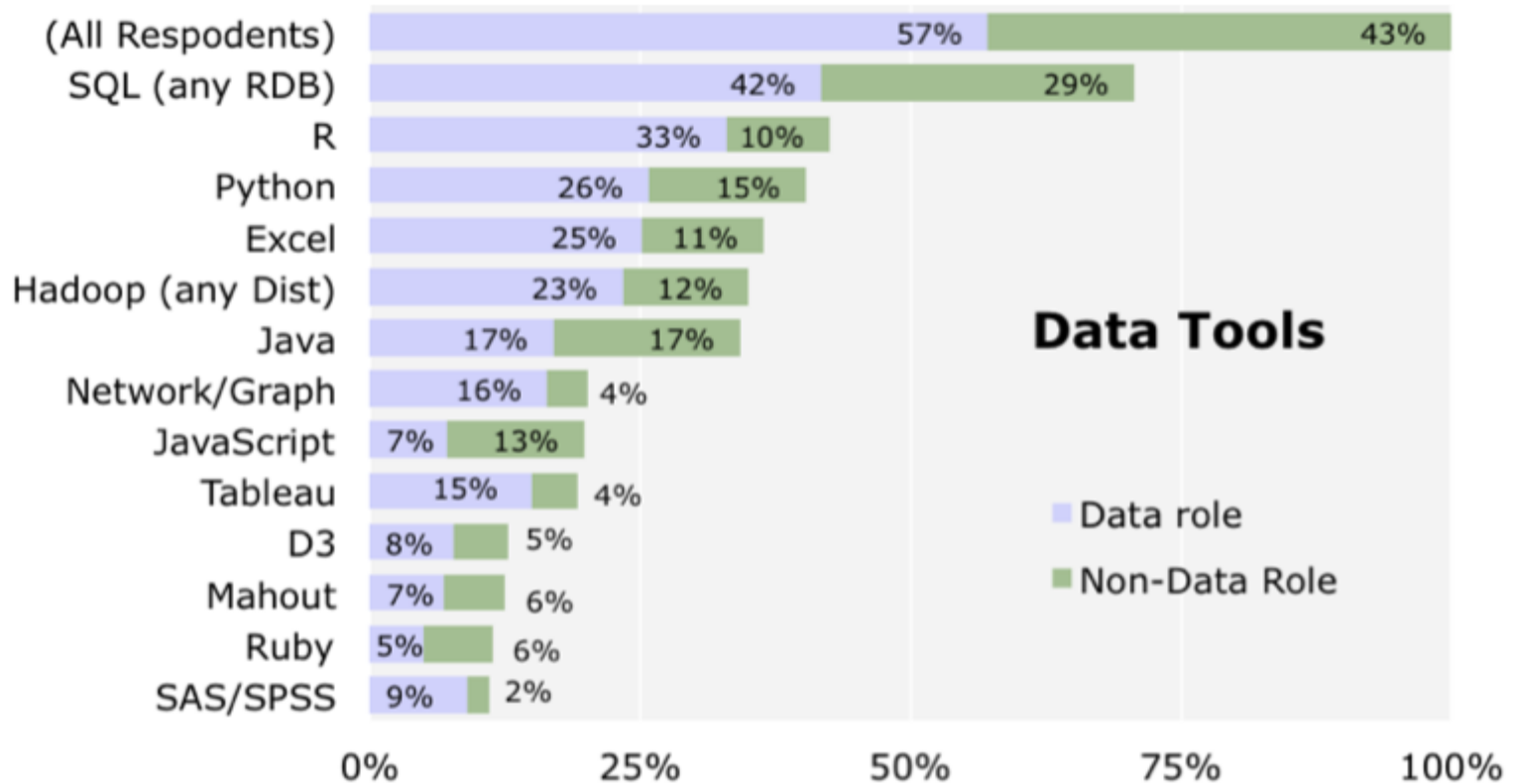
R is a programming environment for statistical and data analysis computations.

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

All Encompassing Environment

- Bioconductor
 - Subproject/environment in R
 - Has own ways of:
 - Installing packages
 - Access data from data repos: Genbank, NCBI
 - Data structures: ExpressionSets, Gene Ranges
 - Run BLAST
 - Heavy OO use
- Wrapper to C/Fortran Code
 - JAGS - Just another Gibbs Sampler
 - rjags + R2jags
 - GDAL - translator library for raster and vector geospatial data formats
 - rgdal

R's Popularity



Orielly Strata Conf
Survey 2013

Getting Started

- Download and install locally from
 - <http://www.r-project.org/>
 - <http://www.rstudio.com/>
- TACC
 - `ssh maverick.tacc.utexas.edu`
 - `%> module load Rstats Rstudio`
 - `%> cp /share/doc/slurm/job.Rstudio .`

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.
```

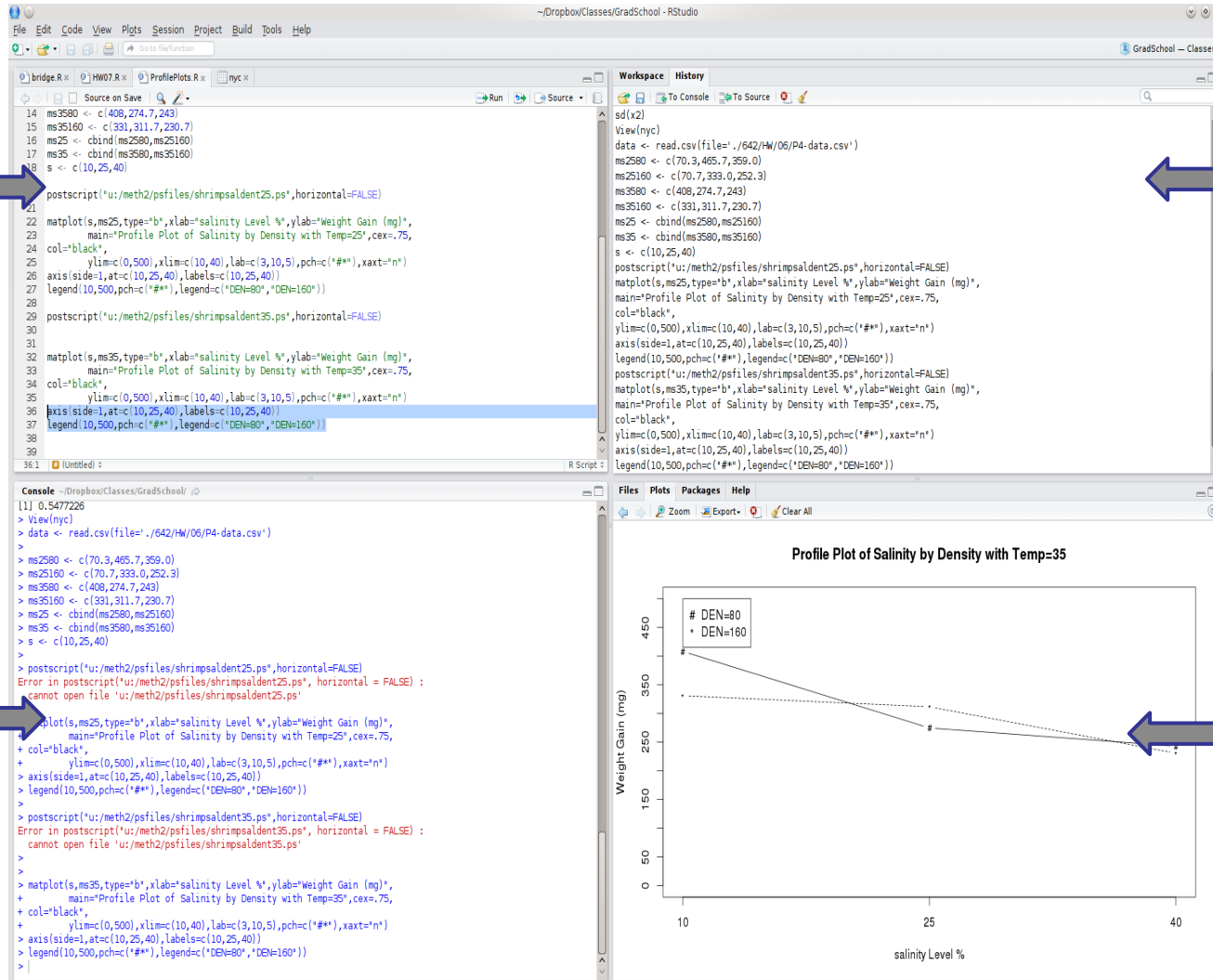
```
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.
```

```
> █
```

RStudio: A better user interface of R

- RStudio is a open source graphical user environment for R users.
 - <http://www.rstudio.com/>
- RStudio allow users to
 - Interactive code development
 - Run R scripts
 - Explore 'local' file system
 - Viewing data
 - Viewing graphical output from R
 - ...

RStudio GUI



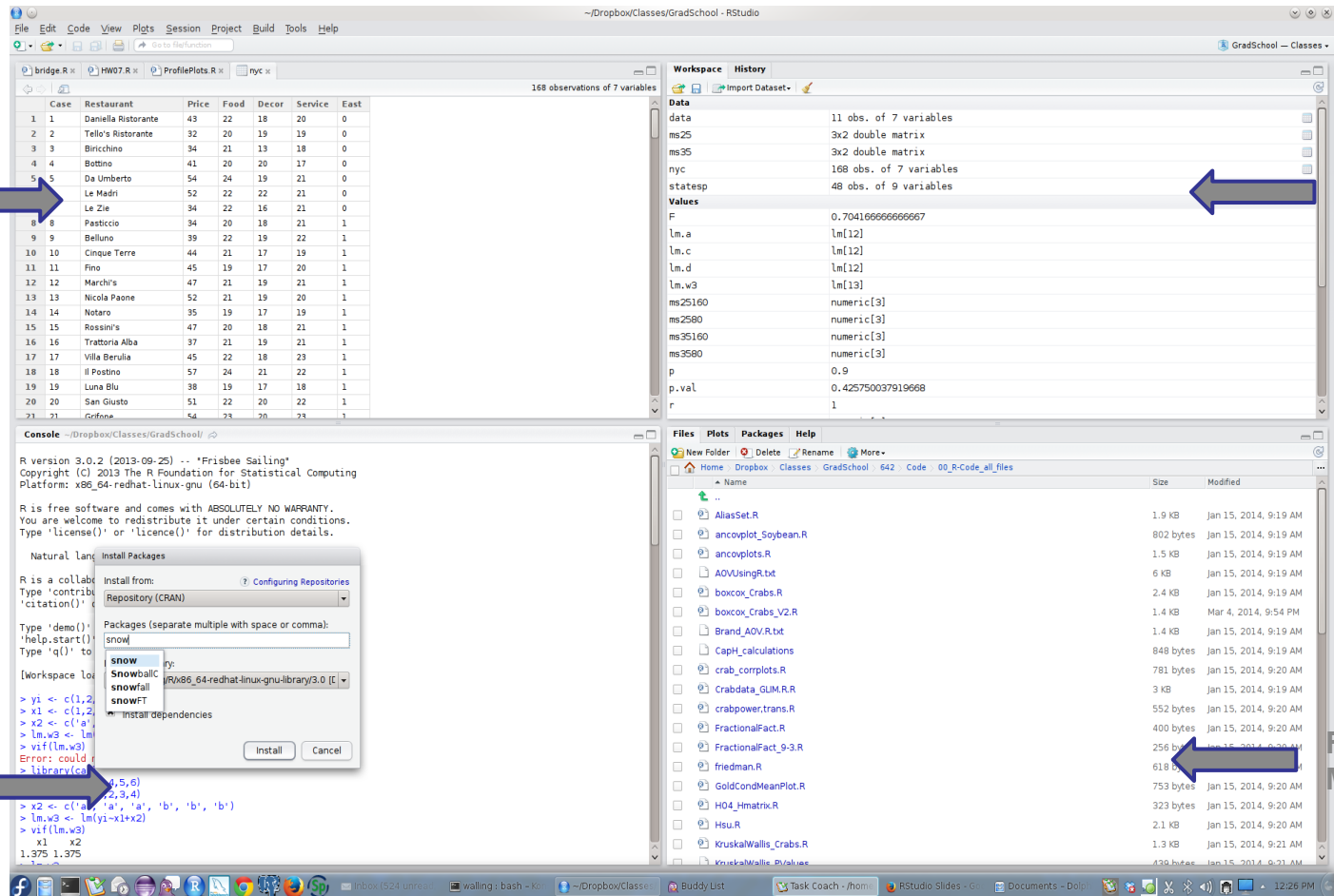
Source Editor +
Interactive
highlight+run

Session
History

Interactive
Console

Plots
Management

RStudio GUI



Data Viewer

Variable/Environment Viewer

Package Management

File Management

RStudio GUI

Project + Source
Control
Integration

The screenshot displays the RStudio interface with the following components:

- Top Panel:** Menu bar (File, Edit, Code, View, Plots, Session, Project, Build, Tools, Help) and a toolbar.
- Left Panel:** A file browser showing a directory structure. A 'Create Project' dialog box is open, offering options: 'New Project' (Start a project in a new directory), 'Existing Directory' (Associate a project with an existing working directory), and 'Version Control' (Checkout a project from a version control repository).
- Center Panel:** The R console showing the execution of R code. The code includes a legend, a postscript plot, and a main plot. The output shows an error: 'cannot open file 'u:/meth2/psfiles/shrimpsalident35.ps''. Below the error, the console suggests trying 'library(snow)' and 'library(parallel)'. The output also shows 'Attaching package: 'parallel'' and 'The following objects are masked from 'package:snow': clusterApply, clusterApplyLB, clusterCall, clusterEvalQ, clusterExport, clusterMap, clusterSplit, makeCluster, parApply, parCapply, parLapply, parLapply, parSapply, splitIndices, stopCluster'.
- Right Panel:** The 'Workspace' and 'History' tabs. The 'Data' tab shows a list of objects: 'data' (11 obs. of 7 variables), 'ms25' (3x2 double matrix), 'ms35' (3x2 double matrix), 'nyc' (168 obs. of 7 variables), 'statesp' (48 obs. of 9 variables), and 'F' (0.704166666666667). The 'Values' tab shows the values of the objects. The 'Files' tab shows the file structure. The 'Plots' tab shows the plot. The 'Packages' tab shows the installed packages. The 'Help' tab shows the 'Parallel Versions of lapply and mapply using Forking' documentation page.

Interactive Help

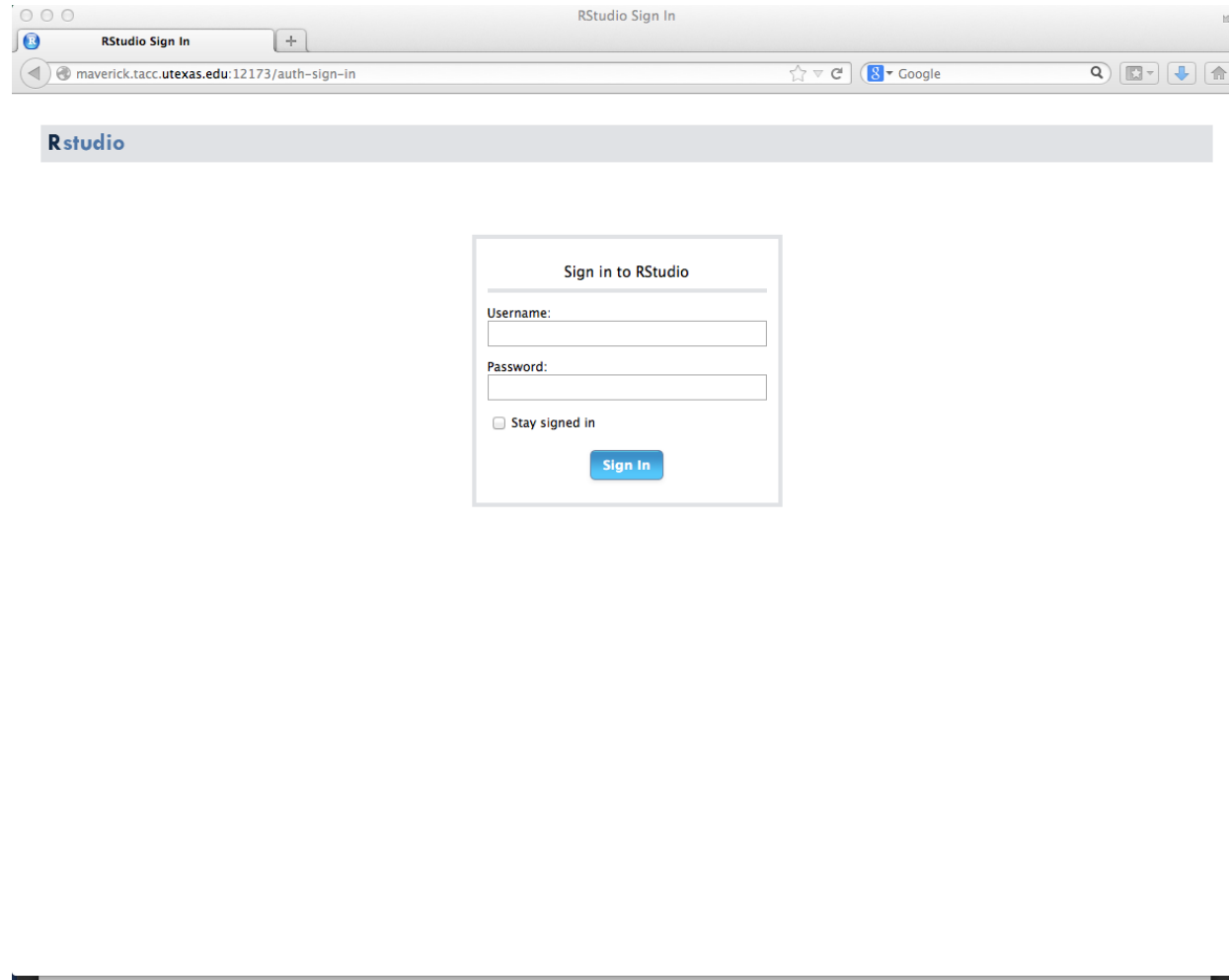
Interactive Help

Web interface of Rstudio on Maverick

- Users can run an interactive web session with RStudio using maverick.tacc.utexas.edu
- The Job script template is in /share/doc/slurm/job.RStudio
- On Stampede,
 - Submitting the job with
 - %> sbatch job.Rstudio
 - Check job status
 - %> squeue -u walling

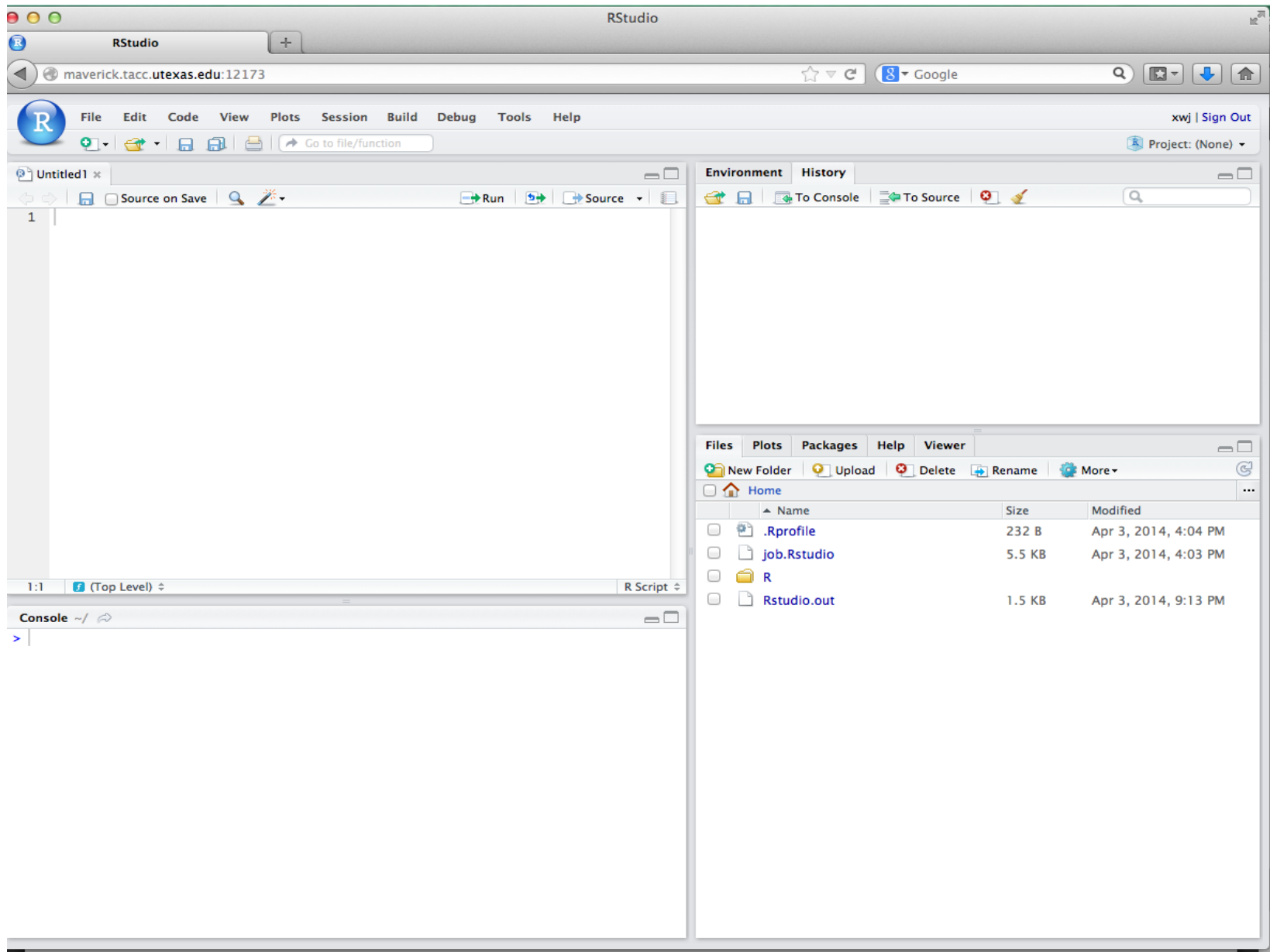
JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
20566	vis	Rstudio	walling	R	0:10	1	c221-701
 - Get port info
 - %> cat Rstudio.out
 - After the job is running, a URL will be available in for connection e.g.
 - <http://maverick.tacc.utexas.edu:12173>

- Then visit the URL and log in with your account credential



The screenshot shows a web browser window titled "RStudio Sign In". The address bar displays the URL "maverick.tacc.utexas.edu:12173/auth-sign-in". The page features the "Rstudio" logo in the top left corner. In the center, there is a "Sign in to RStudio" form with the following fields and options:

- Username:** A text input field.
- Password:** A text input field.
- ☐ Stay signed in
- Sign In** button



Basic 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
```

- Assign multiple values

- Concatenate, c()
- From 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
- Matrix
 - A rectangular table of data of the same type

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

Basic Data Structure

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

```
> 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

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 its 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

subset rows by a
logical vector

```
> worms$Vegetation
```

```
[1] Arable   Scrub    Grassland Grassland Grassland Meadow  Meadow
[8] Arable
Levels: Arable Grassland Meadow Scrub
```

subset a column

```
> worms$Vegetation=="Grassland"
```

```
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

comparison resulting
in logical vector

```
> 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 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

Flow Control

- If ... else

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

- loops

* else branch is optional

```
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` (list, fct) and `sapply` (list, 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

```
> 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
```

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)
```

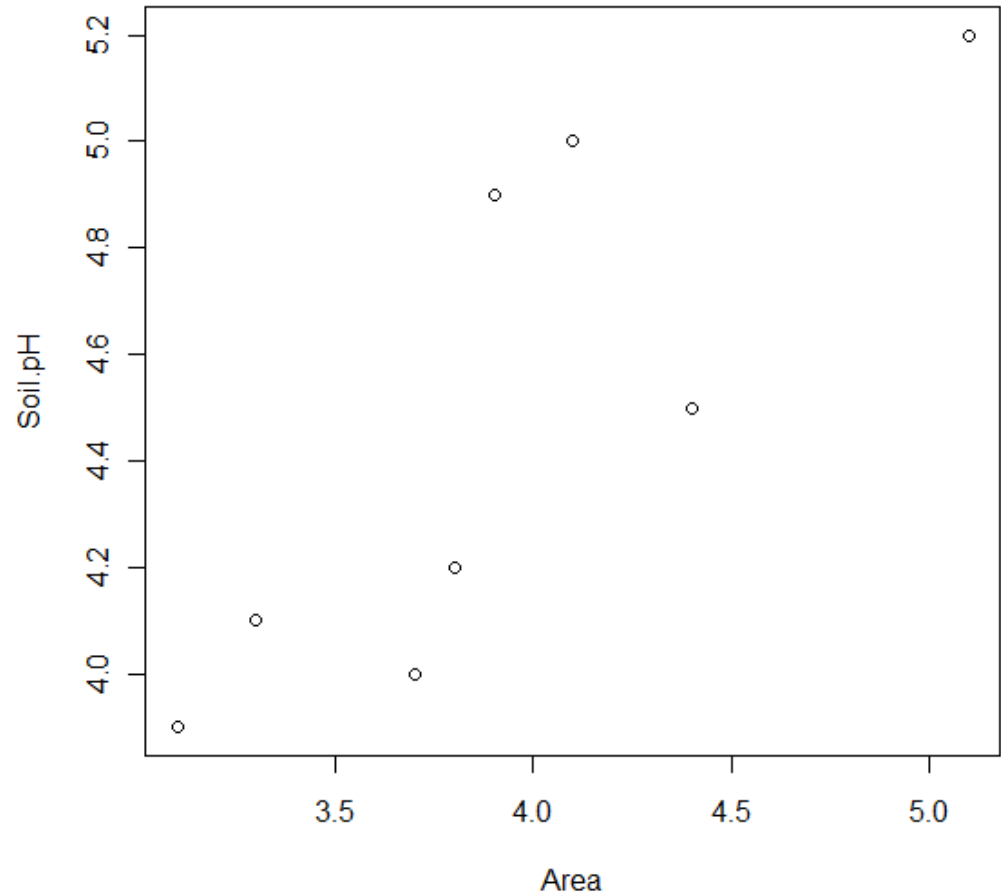
Area	Slope	Vegetation	Soil.pH	Damp	Worm.density
Min. :3.100	Min. :0.000	Arable :2	Min. :3.900	Mode :logical	Min. :1.000
1st Qu.:3.600	1st Qu.:0.000	Grassland:3	1st Qu.:4.075	FALSE:6	1st Qu.:2.000
Median :3.850	Median :1.500	Meadow :2	Median :4.350	TRUE :2	Median :5.500
Mean :3.925	Mean :1.125	Scrub :1	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

Create Plots

- `plot(...)`
 - Create scatter plot.

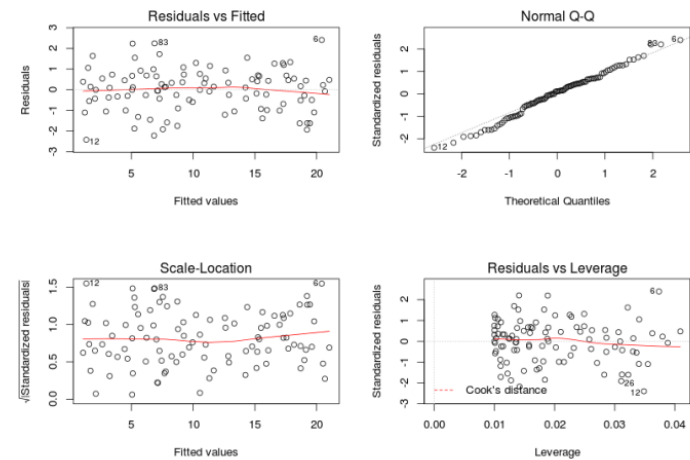
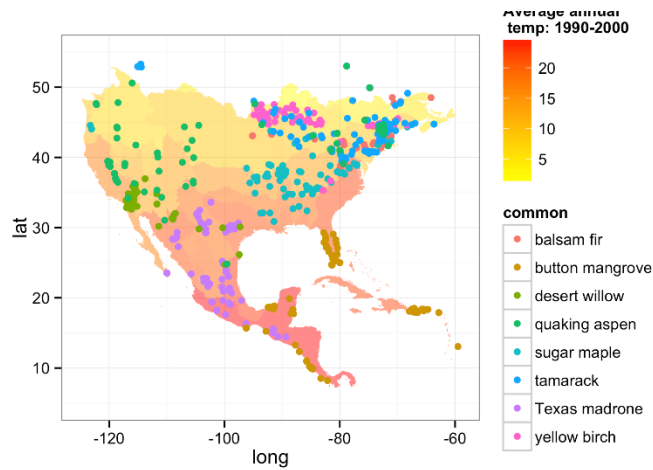
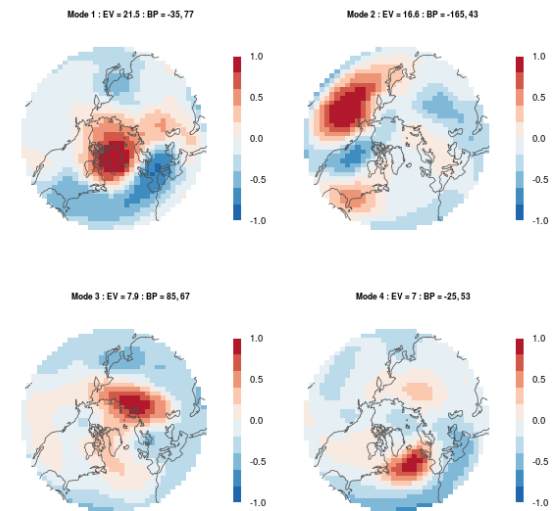
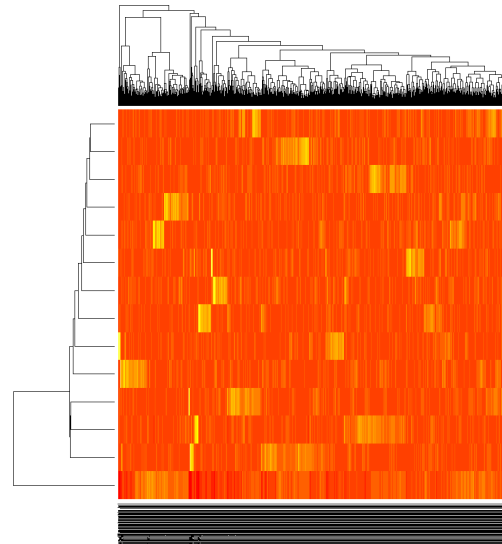
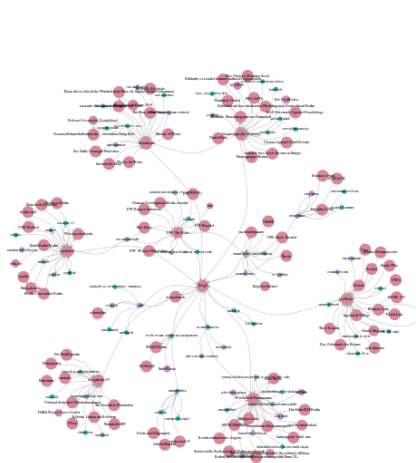
➤ **`plot(Area, Soil.pH)`**

Automatically create
a postscript file with
default name



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



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.

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")
```

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
```

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 5,000+ packages at <http://cran.r-project.org/web/packages/>
 - E.g. Rweka, interface to Weka.

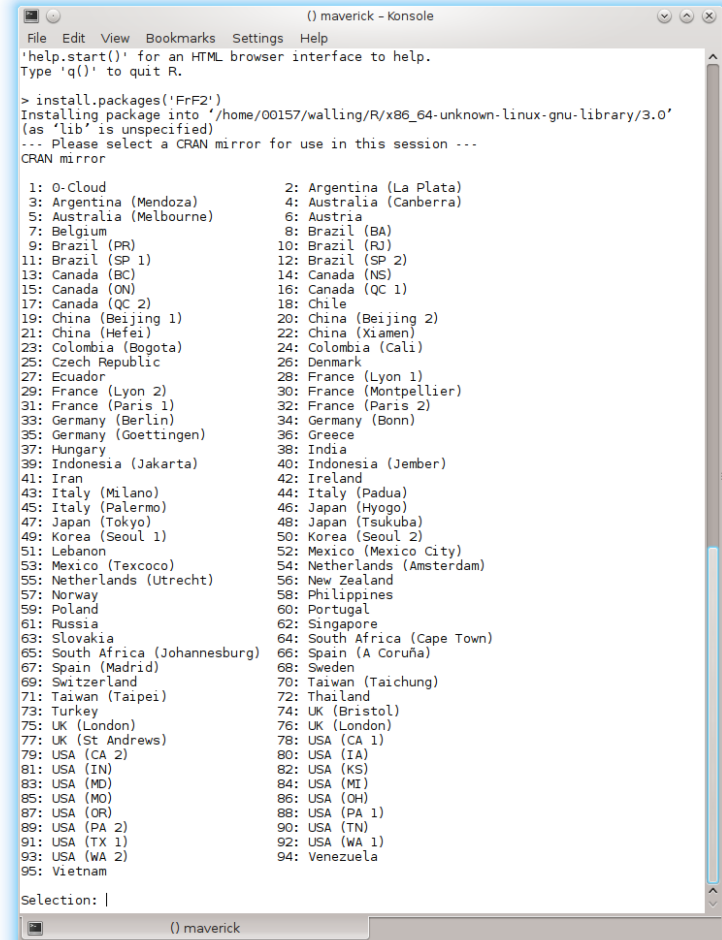
Installing Packages on TACC Systems

- 5465 packages available on CRAN: Comprehensive R Archive Network
- R handles package dependencies for you.
- Many packages compile C/Fortran.
- 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/library"' 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/intel64 -lmkl_rt -shared -fPIC -openmp -mkl=parallel
-O3 -xHost -L/opt/apps/intel/13/composer_xe_2013_sp1.
1.106/mkl/lib/intel64 -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
```



```
() maverick - Konsole
File Edit View Bookmarks Settings Help
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages("FrF2")
Installing package into '/home/00157/walling/R/x86_64-unknown-linux-gnu-library/3.0'
(as 'lib' is unspecified)
... Please select a CRAN mirror for use in this session ...
CRAN mirror

1: O-Cloud
2: Argentina (La Plata)
3: Argentina (Mendoza)
4: Australia (Canberra)
5: Australia (Melbourne)
6: Austria
7: Belgium
8: Brazil (BA)
9: Brazil (PR)
10: Brazil (RJ)
11: Brazil (SP 1)
12: Brazil (SP 2)
13: Canada (BC)
14: Canada (NS)
15: Canada (ON)
16: Canada (QC 1)
17: Canada (QC 2)
18: Chile
19: China (Beijing 1)
20: China (Beijing 2)
21: China (Hefei)
22: China (Xiamen)
23: Colombia (Bogota)
24: Colombia (Cali)
25: Czech Republic
26: Denmark
27: Ecuador
28: France (Lyon 1)
29: France (Lyon 2)
30: France (Montpellier)
31: France (Paris 1)
32: France (Paris 2)
33: Germany (Berlin)
34: Germany (Bonn)
35: Germany (Goettingen)
36: Greece
37: Hungary
38: India
39: Indonesia (Jakarta)
40: Indonesia (Jember)
41: Iran
42: Ireland
43: Italy (Milano)
44: Italy (Padua)
45: Italy (Palermo)
46: Japan (Hyogo)
47: Japan (Tokyo)
48: Japan (Tsukuba)
49: Korea (Seoul 1)
50: Korea (Seoul 2)
51: Lebanon
52: Mexico (Mexico City)
53: Mexico (Texcoco)
54: Netherlands (Amsterdam)
55: Netherlands (Utrecht)
56: New Zealand
57: Norway
58: Philippines
59: Poland
60: Portugal
61: Russia
62: Singapore
63: Slovakia
64: South Africa (Cape Town)
65: South Africa (Johannesburg)
66: Spain (A Coruña)
67: Spain (Madrid)
68: Sweden
69: Switzerland
70: Taiwan (Taichung)
71: Taiwan (Taipei)
72: Thailand
73: Turkey
74: UK (Bristol)
75: UK (London)
76: UK (London)
77: UK (St Andrews)
78: USA (CA 1)
79: USA (CA 2)
80: USA (IA)
81: USA (IN)
82: USA (KS)
83: USA (MD)
84: USA (MI)
85: USA (MO)
86: USA (OH)
87: USA (OR)
88: USA (PA 1)
89: USA (PA 2)
90: USA (TN)
91: USA (TX 1)
92: USA (WA 1)
93: USA (WA 2)
94: Venezuela
95: Vietnam

Selection: |

() maverick
```

R @ TACC

- Memory
 - R is a memory hog. 32GB - 1TB RAM nodes
 - Bigmemory packages: use file system to stream matrices
- Multi Core
 - Embarrassingly Parallel jobs can easily utilize multicore/nodes
 - library(parallel)
 - `> results <- mclapply(1:16, function(i) mySolver(params[1,]))`
- MIC/GPU Offloading
 - Much of data analytics uses matrix operations.
 - Auto-offload to MIC/GPU resources without changing code.
- Compiler Optimizations
 - Intel compiler
 - MKL BLAS/LAPACK routines
- Package installation support

Further references

- R
 - M. Crawley, Statistics An Introduction using R, Wiley
 - J. Verzani, SimpleR Using R for Introductory Statistics
<http://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf>
 - Programming manual:
 - <http://cran.r-project.org/manuals.html>
 - <http://r-bloggers.com>

We gratefully acknowledge the sponsorship of Chevron Corporation, whose generous support of TACC has made possible this Scientific Computing Curriculum and other student-focused initiatives.

License

© The University of Texas at Austin, 2014

This work is licensed under the Creative Commons Attribution Non-Commercial 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/>

When attributing this work, please use the following text:

"Science and Technical Computing course materials by The Texas Advanced Computing Center, 2014. Available under a Creative Commons Attribution Non-Commercial 3.0 Unported License"