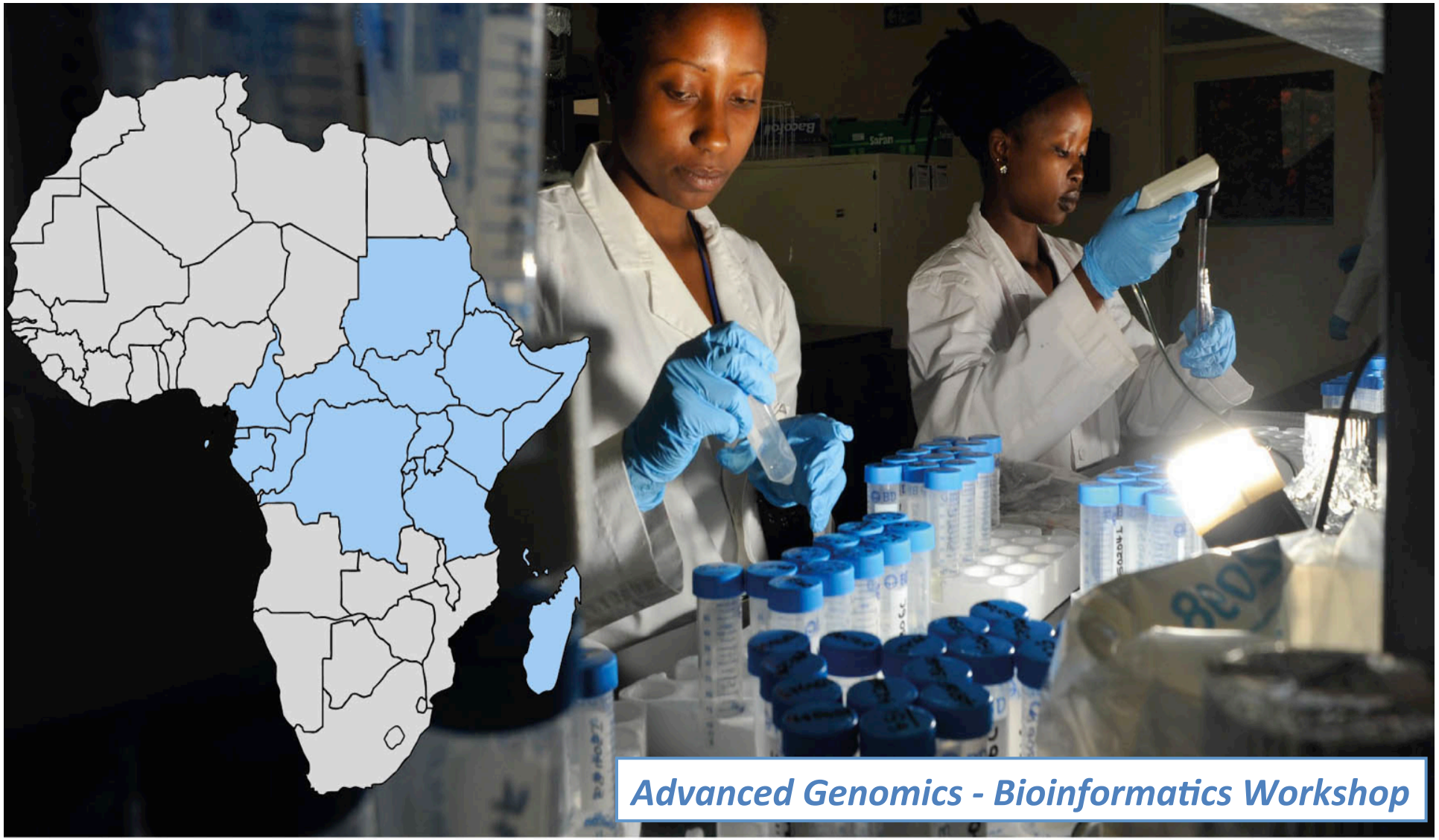


Advanced Genomics - Bioinformatics Workshop

Mark Wamalwa
BecA-ILRI Hub, Nairobi, Kenya
http://hub.africabiosciences.org/
m.wamalwa@cgiar.org

ILRI
INTERNATIONAL LIVESTOCK RESEARCH INSTITUTE

African Union

NEPAD
THE NEW PARTNERSHIP FOR AFRICA'S DEVELOPMENT
A PROGRAMME OF THE AFRICAN UNION

7th – 18th September 2015

biosciences
eastern and central africa

Lecture Overview

- What is R and why use it?
- Setting up R & RStudio for use
- Calculations, functions and variable classes
- File handling, plotting and graphic features
- Statistics
- Packages and writing functions

What is  ?

- “R is a freely available language and environment for statistical computing and graphics”

- Much like  &  , but bette  !

Overview

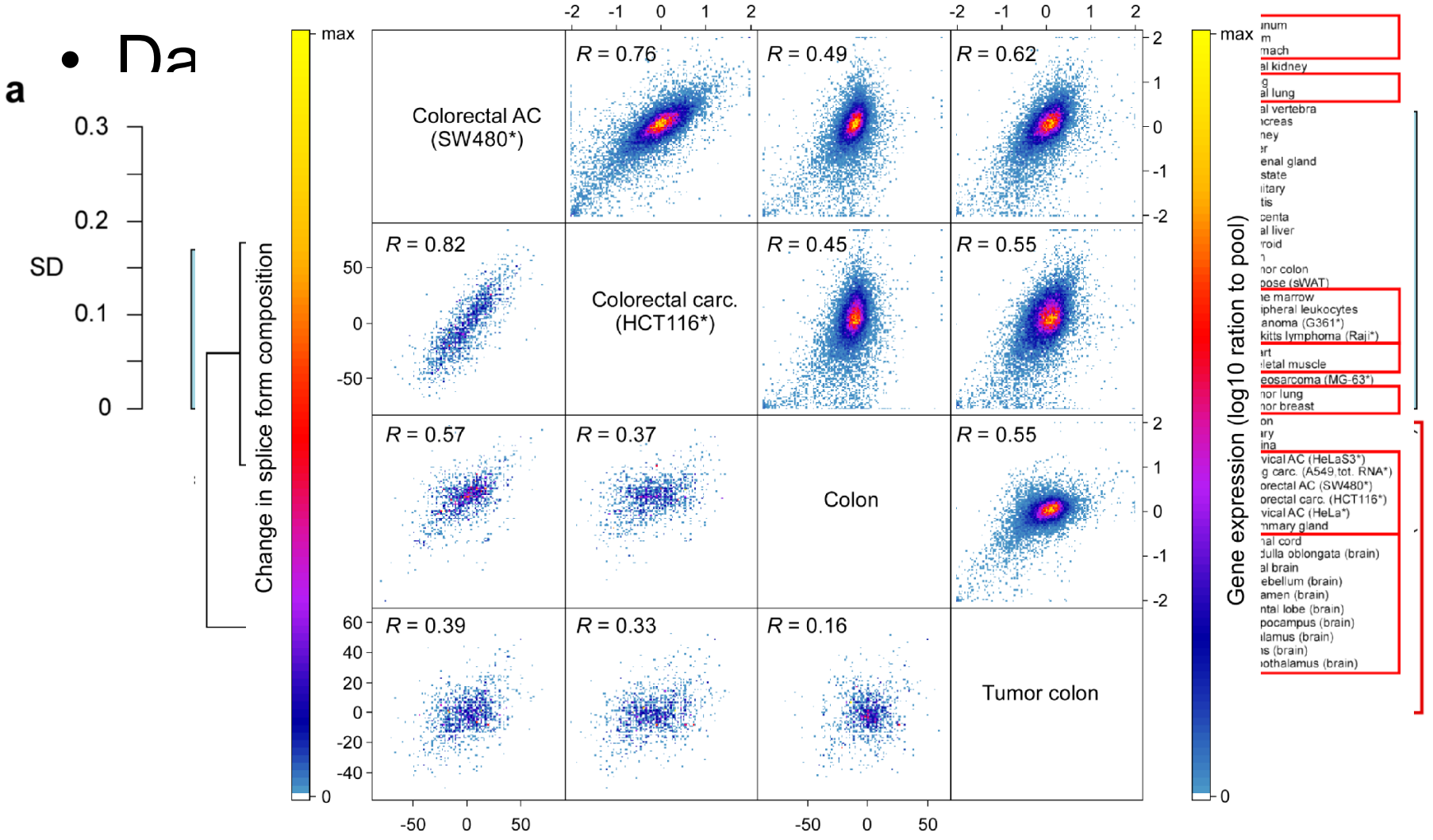
- R is a comprehensive statistical and graphical programming language and is a dialect of the S language:
  - 1988 - S2: RA Becker, JM Chambers, A Wilks
  - 1992 - S3: JM Chambers, TJ Hastie
  - 1998 - S4: JM Chambers
- R: initially written by Ross Ihaka and Robert Gentleman at Dep. of Statistics of U of Auckland, New Zealand during 1990s.
- Since 1997: international “R-core” team of 15 people with access to common CVS archive.

Why use  ?

- SPSS and Excel users are **limited** in their **developed for them by statistical researchers** or **create their own**
- They **don't have to pay** money to use them
- Once experienced enough they are almost unlimited in their ability to change their environment
- The users have to **pay** money to use the software

‘s Strengths

- Da



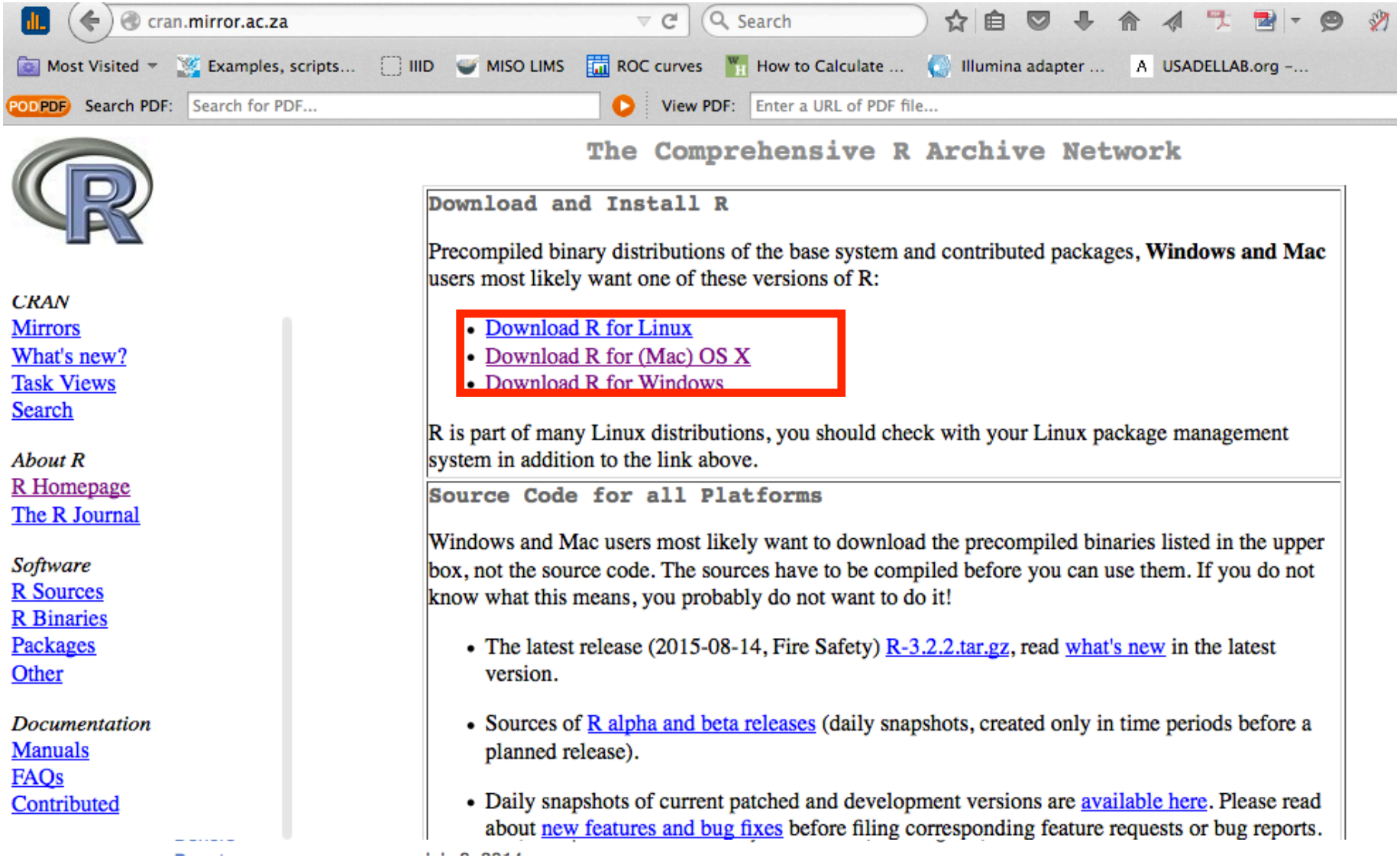
‘s Weaknesses

- Not very user friendly at start
- No commercial support
- Substantially slower than programming languages (e.g. Perl, Java, C++)

Lecture Overview

- What is R and why use it?
- Setting up R & RStudio for use
- Calculations, functions and variable classes
- File handling, plotting and graphic features
- Statistics
- Packages and writing functions

Installing

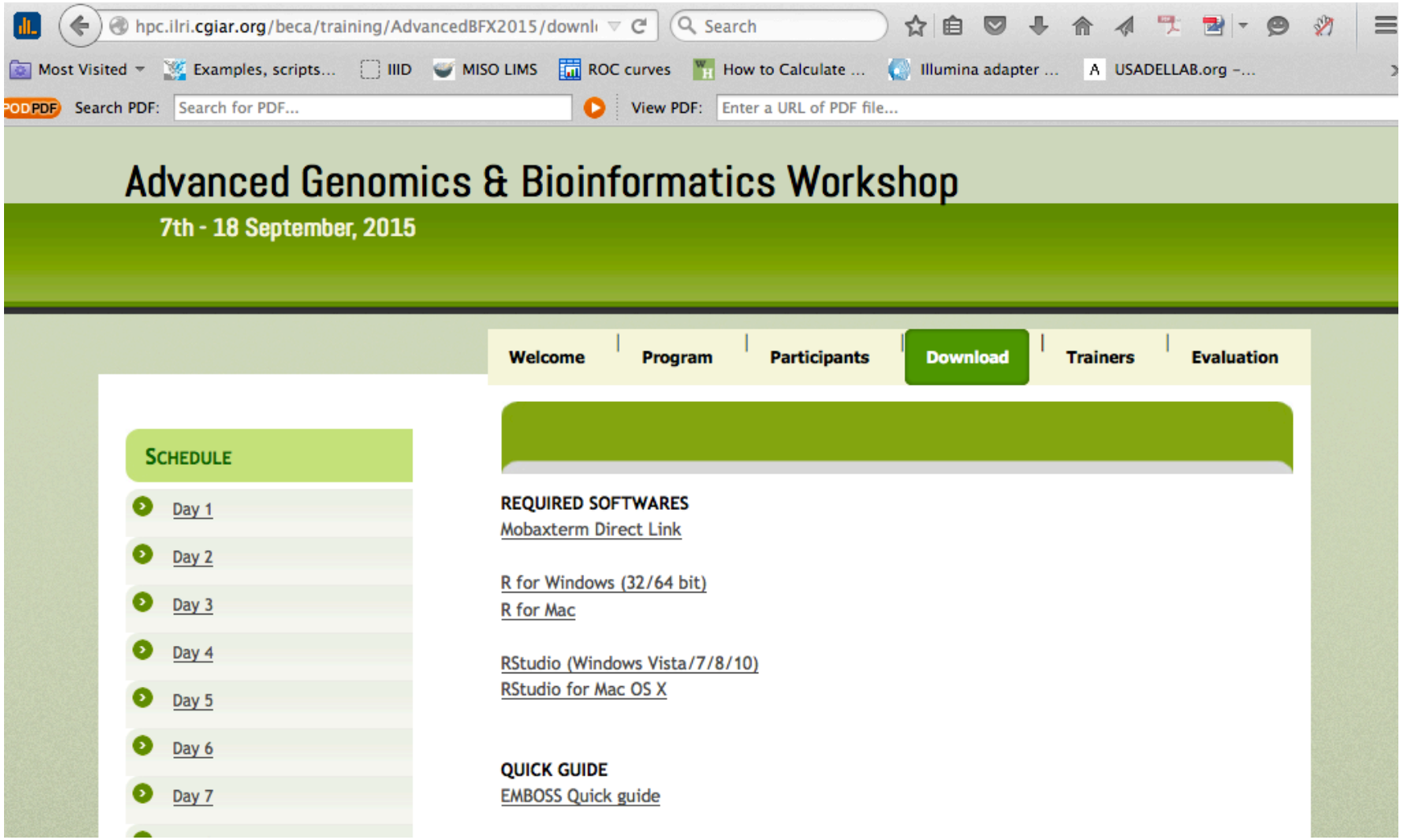

cran.mirror.ac.za
Search
Most Visited
Examples, scripts...
IIID
MISO LIMS
ROC curves
How to Calculate ...
Illumina adapter ...
USADELLAB.org -...
Search PDF:
Search for PDF...
View PDF:
Enter a URL of PDF file...
The Comprehensive R Archive Network
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
Contributed
Download and Install R
Precompiled binary distributions of the base system and contributed packages, Windows and Mac users most likely want one of these versions of R:
Download R for Linux
Download R for (Mac) OS X
Download R for Windows
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.
Source Code for all Platforms
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!
The latest release (2015-08-14, Fire Safety) R-3.2.2.tar.gz, read what's new in the latest version.
Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.

Installing Rstudio 

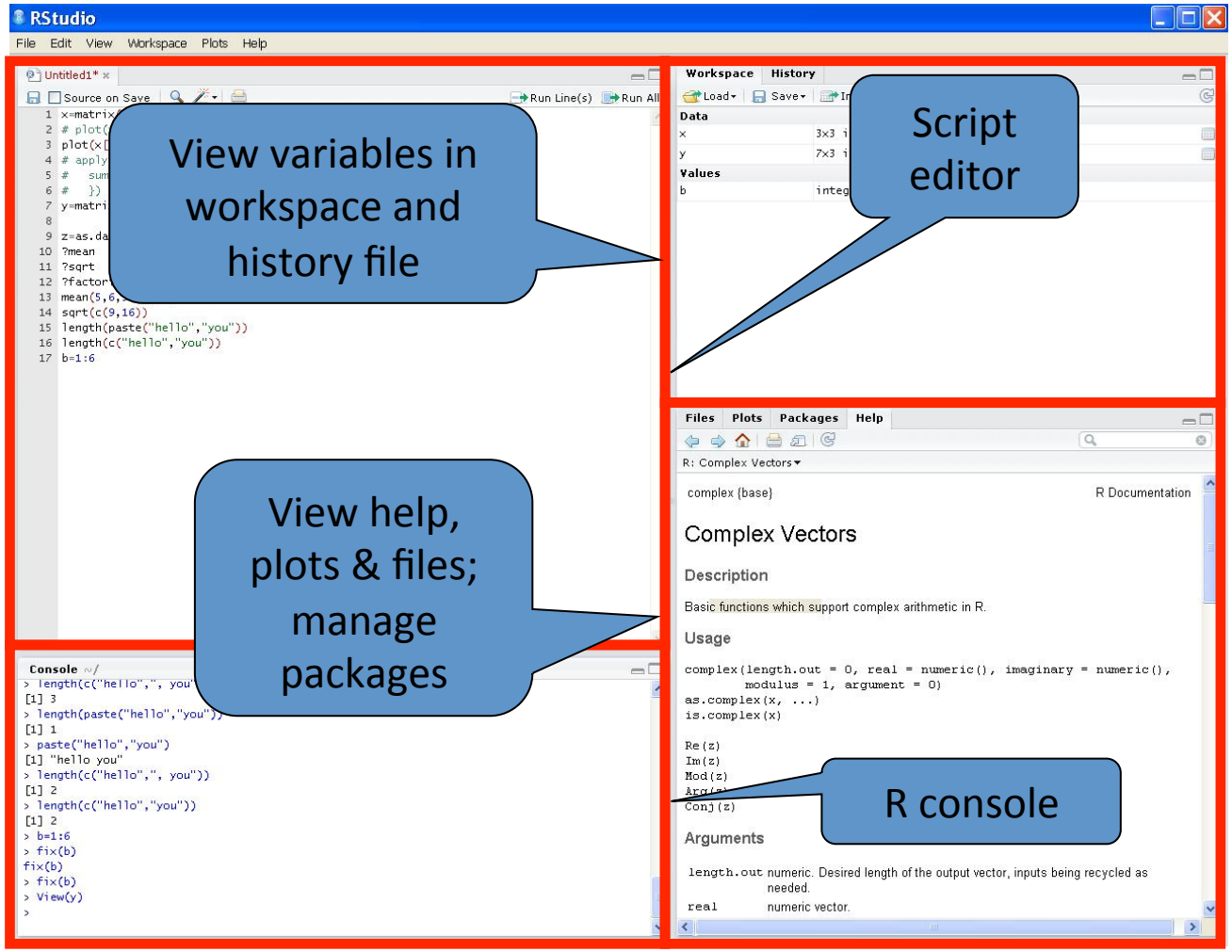
- “RStudio is a… integrated development environment (IDE) for R”

- Install the “desktop edition” from this link: http://www.rstudio.org/download/
- http://hpc.ilri.cgiar.org/beca/training/AdvancedBFX2015/download.html

Installing Rstudio

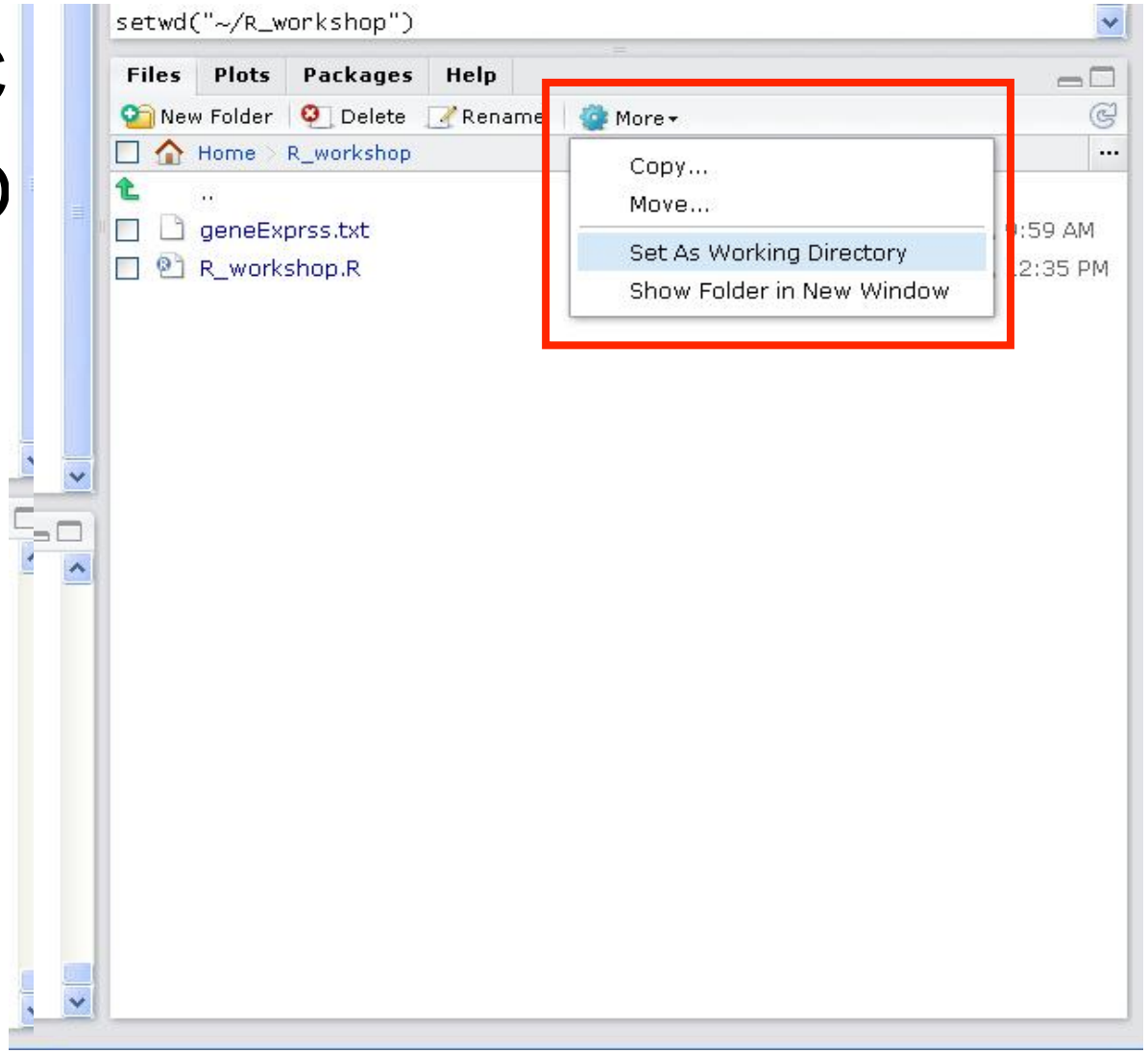


Using RStudio



Set Up Your Workspace

- C
- O

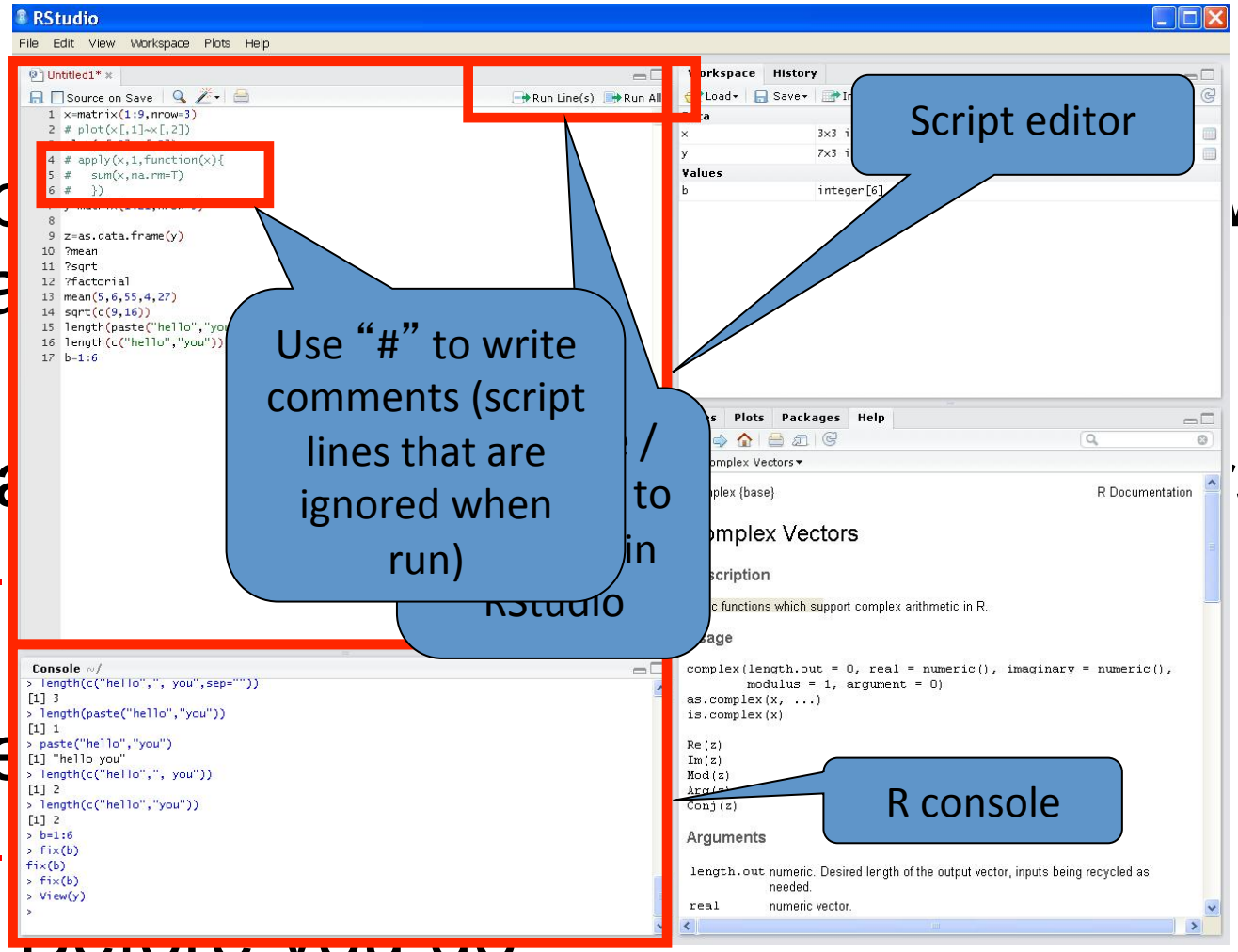


Lecture Overview

- What is R and why use it?
- Setting up R & RStudio for use
- Calculations, functions and variable classes
- File handling, plotting and graphic features
- Statistics
- Packages and writing functions

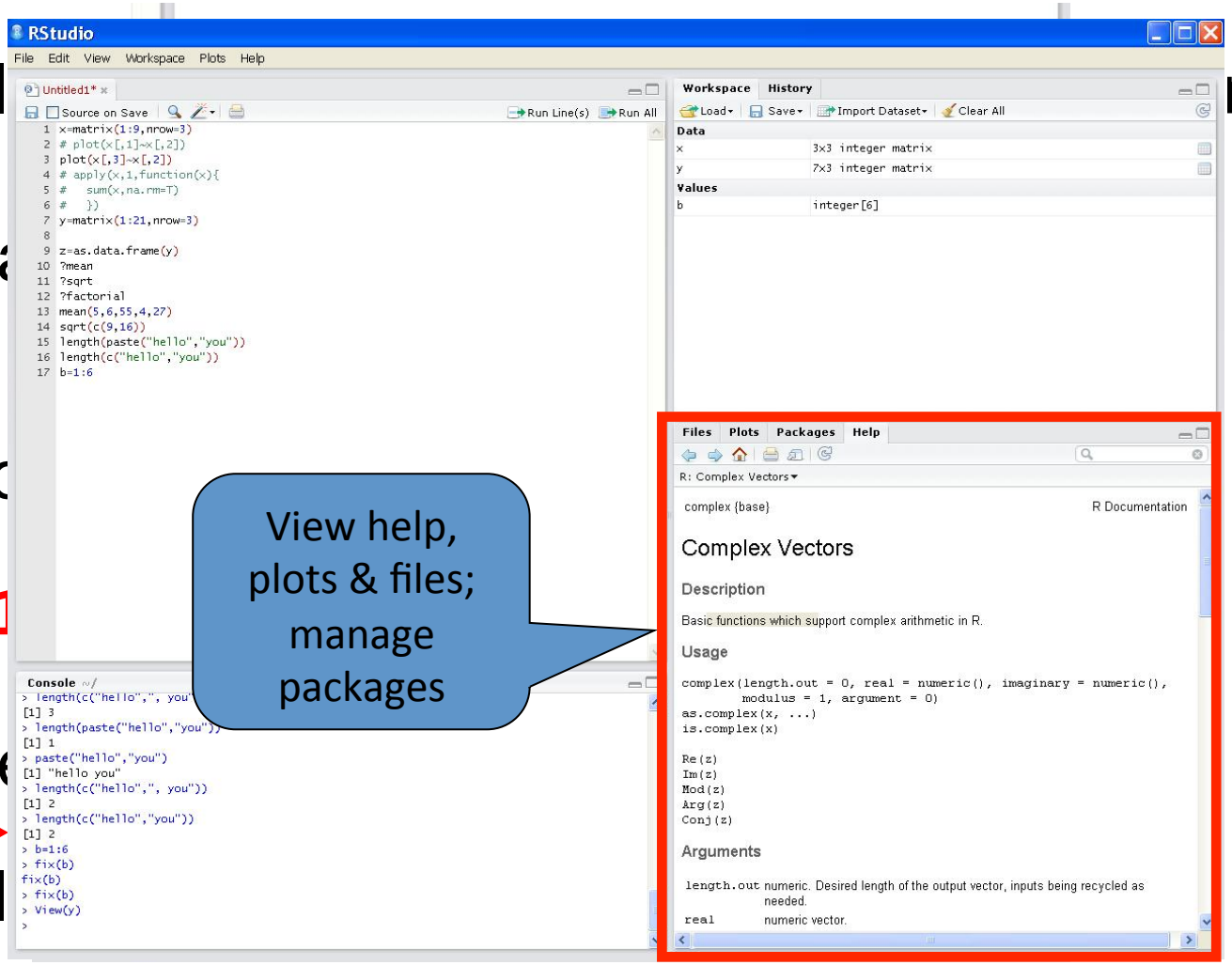
- Basic Calculations

- O
  op
  va
- Ba
  –
- Le
  –



- Basic Functions

- Al ns
- Ca
  >
- Fo
  >
  [1
- Re
  >
  Al



Data Types

R has a wide variety of data types including scalars, vectors (numerical, character, logical), matrices, dataframes, and lists.

Variables

- A variable is a symbolic name given to stored information

- Variables are assigned using either "=" or "<-"

```
> x<-12.6
> x
[1] 12.6
```

Vectors

```
a <- c(1,2,5.3,6,-2,4) # numeric vector
b <- c("one","two","three") # character vector
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)

#logical vector
Refer to elements of a vector using subscripts.
a[c(2,4)] # 2nd and 4th elements of vector
```

Variables - Numeric Vectors

- A vector is the simplest R data structure. A numeric vector is a single entity consisting of a collection of numbers.
- It may be created:
  - Using the `c()` function (**concatenate**) :

```
x=c(3,7.6,9,11.1)
> x
[1]  3  7.6  9 11.1
```

  - Using the `rep(what,how_many_times)` function (**replicate**):

```
x=rep(10.2,3)
```

  - Using the "`:`" operator, signifiying a series of integers

```
x=4:15
```

Variables - Character Vectors

- Character strings are always double quoted
- Vectors made of character strings:

```
> y=c("I","want","to","go","home")
> y
[1] "I" "want" "to" "go" "home"
```

- Using `rep()`:

```
> rep("bye",2)
[1] "bye" "bye"
```

- Notice the difference using `paste()` (1 element):

```
> paste("I","want","to","go","home")
[1] "I want to go home"
```

Variables - Boolean Vectors

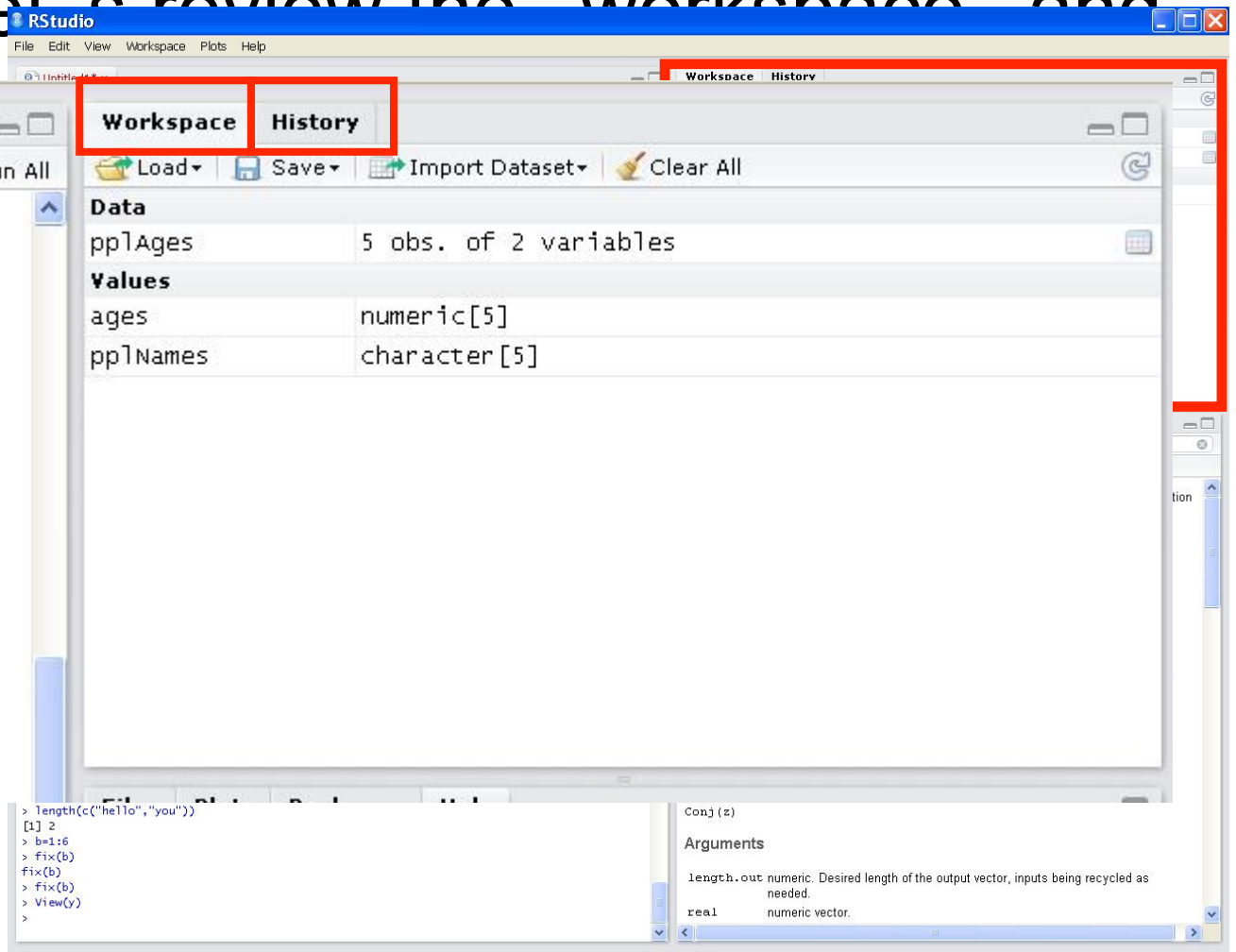
- Logical; either **FALSE** or **TRUE**

- ```
  > 5>3
  [1] TRUE
  ```

- ```
  > x=1:5
  > x
  [1] 1 2 3 4 5
  > x<3
  [1] TRUE TRUE FALSE FALSE FALSE
  z=x<3
  ```

RStudio – Workspace & History

- Let's review the 'workspace' and 'h



Manipulation of  Vectors

- Our vector: `x=c(101,102,103,104)`
- `[]` are used to access elements in `x`
- Extract 2nd element in `x`

```
> x[2]
[1] 102
```

- Extract 3rd and 4th elements in `x`

```
> x[3:4] # or x[c(3,4)]
[1] 103 104
```

Manipulation of  Vectors

-
```
> x
[1] 101 102 103 104
```

- Add 1 to all elements in **x**:

```
> x+1
[1] 102 103 104 105
```

- Multiply all elements in **x** by 2:

```
> x*2
[1] 202 204 206 208
```

More  Operators

- Comparison operators:
  - Equal `==`
  - Not equal `!=`
  - Less / greater than `<` / `>`
  - Less / greater than or equal `<=` / `>=`

- Boolean (either `FALSE` or `TRUE`)
  - And `&`
  - Or `|`
  - Not `!`

Manipulation of  Vectors

- Our vector: `x=100:150`

- Elements of `x` higher than 145

```
> x[x>145]
[1] 146 147 148 149 150
```

- Elements of `x` higher than 135 and lower than 140

```
> x[ x>135 & x<140 ]
[1] 136 137 138 139
```

Manipulation of  Vectors

- Our vector:

```
> x=c("I","want","to","go","home")
```

- Elements of x that do not equal “want”:

```
> x[x != "want"]
[1] "I" "to" "go" "home"
```

Note: use “`==`” for 1 element and “`%in%`” for several elements

- Elements of x that equal “want” **and** “home”:

```
> x[x %in% c("want","home")]
[1] "want" "home"
```

Variables – Matrices

All columns in a matrix must have the same mode(numeric, character, etc.) and the same length.

General format

mymatrix <- **matrix(***vector*, **nrow=***r*, **ncol=***c*, **byrow=***FALSE*,**dimnames=list(***char_vector_rownames*, *char_vector_colnames***))**

byrow=TRUE indicates that the matrix should be filled by rows. **byrow=FALSE** indicates that the matrix should be filled by columns (the default). **dimnames** provides optional labels for the columns and rows.

Variables – Matrices

- A matrix is a table of a different class
  - Accessing elements in matrices:
    - `x[row,column]`
    - The "Height" column:

```
> x[,"Height"] # or:
> x[,2]
```

    - Note: you cannot use "$"

```
> x$Weight
```

Weight	Height
67	174
58	160
80	187
77	173
95	185

- Each **column** must be of t
  (e.g. numeric, character, e
- The number of elements in each row must be identical

Variables – Matrices

```
# generate a 5 x 4 numeric matrix
    y<-matrix(1:20, nrow=5,ncol=4)
    rnames <- c("R1", "R2","R3","R4","R5")
    cnames <- c("C1", "C2","C3","C4")
    y<-matrix(1:20, nrow=5,ncol=4,
    byrow=TRUE,dimnames=list(rnames, cnames))
# another example
    cells <- c(1,26,24,68)
    rnames <- c("R1", "R2")
    cnames <- c("C1", "C2")
    mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE,
    dimnames=list(rnames, cnames))
#Identify rows, columns or elements using subscripts.
y[,4] # 4th column of matrix
    y[3,] # 3rd row of matrix
    y[2:4,1:3] # rows 2,3,4 of columns 1,2,3
```

Variables – Data Frames

- A data frame is simply a table
- Each **column** may be of a different class (e.g., numeric, character, etc.)
- The number of elements in each row must be identical

```
d <- c(1,2,3,4)
e <- c("red", "white", "red", NA)
f <- c(TRUE,TRUE,TRUE,FALSE)
mydata <- data.frame(d,e,f)
names(mydata) <- c("ID","Color","Passed")
  #variable names

There are a variety of ways to identify the elements of a
dataframe .

myframe[3:5] # columns 3,4,5 of dataframe
myframe[c("ID","Age")] # columns ID and Age from dataframe
myframe$X1 # variable x1 in the dataframe
```

Variables – Data Frames

- A data frame is simply a table
  - Accessing elements in data frame:
    - `x[row,column]`
    - The "age" column:

```
> x$age # or:
> x[,"age"] # or:
> x[,1]
```

- Each **column** may be of a different class (e.g. numeric, character, etc.)
- The number of elements in each row must be identical

age	gender	disease
50	M	TRUE
43	M	FALSE
25	F	TRUE
18	M	TRUE
72	F	FALSE
65	M	FALSE
45	F	TRUE

    - All male rows:

```
> x[x$gender=="M",]
```

Lists

An ordered collection of objects (components). A list allows you to gather a variety of (possibly unrelated) objects under one name.

```
# example of a list with 4 components -
    # a string, a numeric vector, a matrix, and a scaler
    w <- list(name="Fred", mynumbers=a, mymatrix=y,
    age=5.3)

    # example of a list containing two lists
    v <- c(list1,list2)
```

Exercise

- Construct the character vector ‘pplNames’ containing 5 names: “Srulik”, “Esti”, ”Shimshon”, “Shifra”, “Ezra”
- Construct the numeric vector ‘ages’ that includes the following numbers: 21, 12 (twice), 35 (twice)
- Use the `data.frame()` function to construct the ‘pplAges’ table out of ‘pplNames’ & ‘ages’
- Access the ‘pplAges’ rows with ‘ages’ values greater than 19

Lecture Overview

- What is R and why use it?
- Setting up R & RStudio for use
- Calculations, functions and variable classes
- File handling, plotting and graphic features
- Statistics
- Packages and writing functions

Working With a File

- For e
- Work
  - Sa
  - Re
  - Ar
- 305 g
  comp
  - Va
  - Va
- File includes 306 rows X 49 columns

gene	Adipose (sWAT)	Adrenal gland	Peripheral leukocytes	
A2BP1	-0.43449	-0.35804	-1.32619	-1.3987
APBB1	-0.28725	-0.22703	-0.17179	-0.244
API5	0.03281	-0.01801	-0.12612	0.1081
APP	-0.27273	-0.1069	-0.97834	-0.35481
AQR	0.09021	0.06473	0.25003	0.28211
AR	0.25143	-0.22576	-0.07426	-0.36435
ARID4A	0.11067	-0.01458	0.56982	0.26355
ARL6IP4	0.05169	0.22849	0.08706	0.05618
ARMET	-0.08804	0.09259	-0.39333	-0.01747
BAT1	-0.04255	-0.0177	-0.07406	-0.0371
BCAS2	0.04659	0.08772	0.02258	-0.16976
BIN1	-0.07957	0.02061	-0.26275	-0.7973
BMPR1A	0.07213	-0.12251	-0.64482	-0.69233
BRUNOL4	-0.2691	-0.17955	-0.15756	-0.18693
BRUNOL5	-0.17979	-0.25154	0.08302	-0.01108
BRUNOL6	0.02433	0.12681	0.29392	0.27288
C16orf33	-0.16461	-0.06196	-0.46222	-0.08127
C19orf29	0.15025	0.19744	0.27608	0.21858
C1QBP	-0.11002	0.20685	-0.41541	-0.12527
C20orf4	0.19923	0.24458	0.32765	0.03384
C6orf151	0.10679	-9e-04	0.05188	0.17772
C6orf206	-0.23908	-0.16266	0.02546	-0.10176
CCNL1	-0.07119	-0.11519	0.40454	0.14201
CCNL2	-0.0019	0.16522	-0.0649	-0.05127
CCNT1	-0.13369	-0.01735	0.54416	0.10649
CCNT2	-0.11765	-0.06113	0.14705	-0.00204

File Handling - Read File

- Read file to R
  - Use the `read.table()` function
  - Note: each function receives **input** (‘**arguments**’) and produces **output** (‘**return value**’)
  - The function returns a data frame
  - Run:

```
> geneExprss = read.table(file = "geneExprss.txt",
sep = "\t",header = T)
```

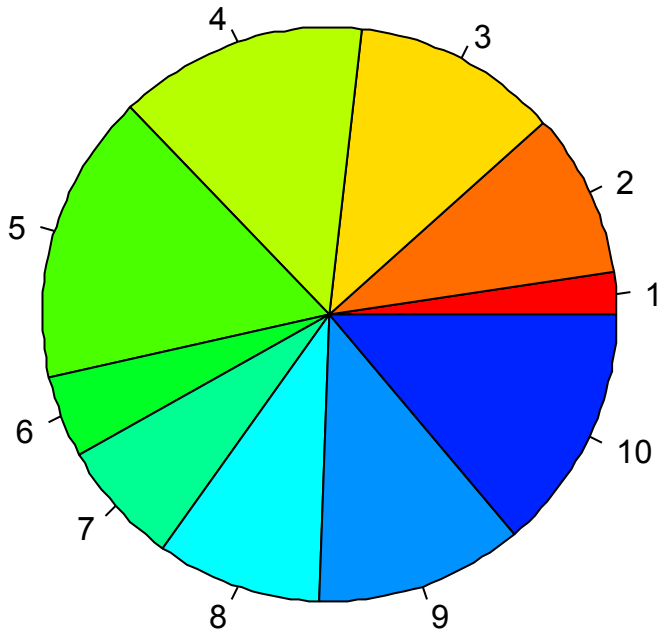
  - Check table:

```
> dim(geneExprss) # table dimentions
> geneExprss[1,] # 1st line
> class(geneExprss) # check variable class
```

  - Or double click on variable name in workspace tab

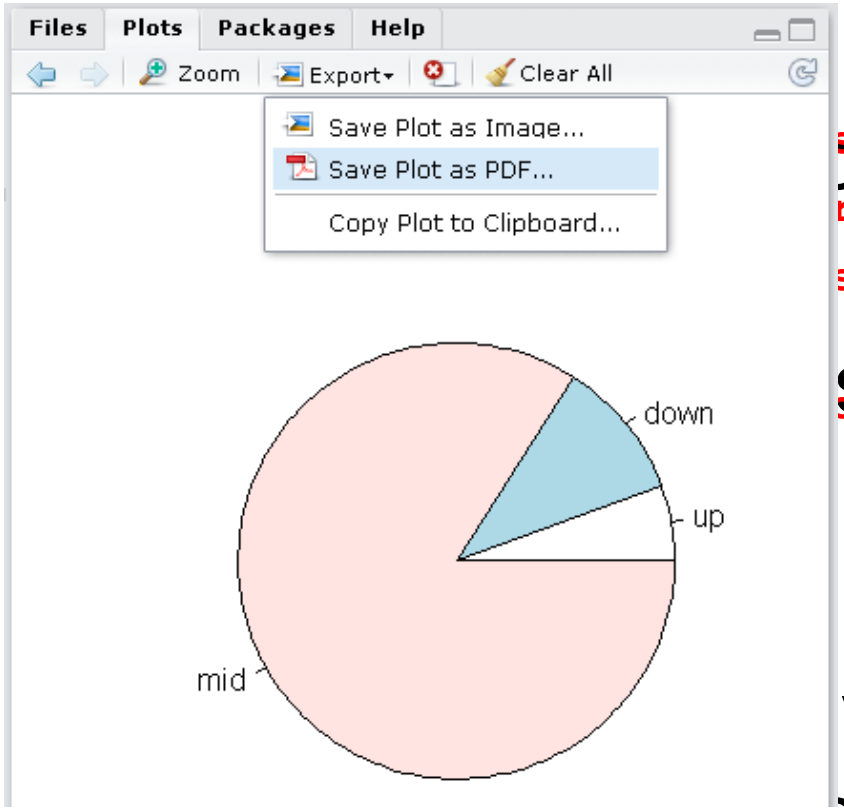
Plotting - Pie Chart

- What fraction of lung genes are over-expressed?
- What about the under-expressed genes?
- A pie chart can illustrate our findings



Using the `pie()` Function

```
> up = leng                                    ss$Lung>0.2])
> down = le                                    rss$Lung<(-0.2)])
> mid = len                                    ss$Lung<=0.2 &
  geneExprs
> pie(c(up,                                    wn","mid"))
```



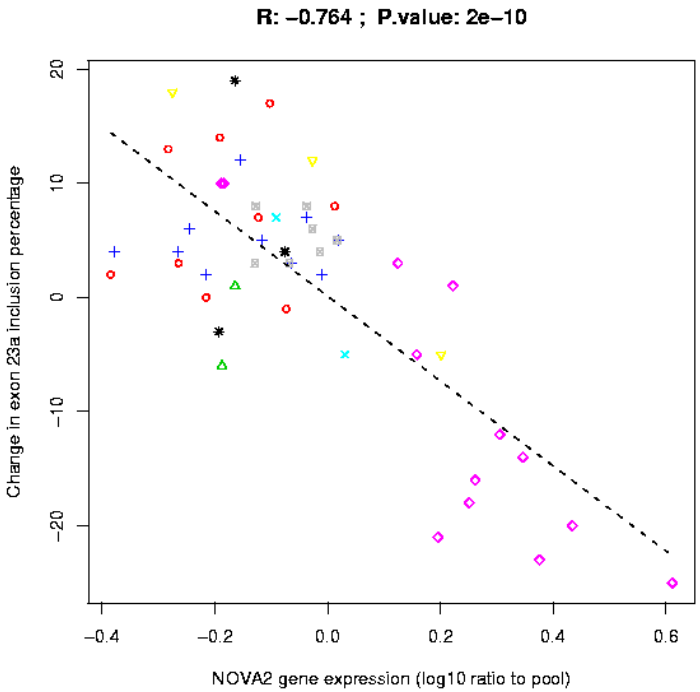
- Let's r … over-expres…
- Let's r … s under-expres…
- Let's u … ves the numbe … or

  - More on saving plots to files in a few slides…

Plotting - Scatter Plot

- How similar is the gene expression profile of the Hippocampus (brain) to that of that of the Thalamus (brain)?

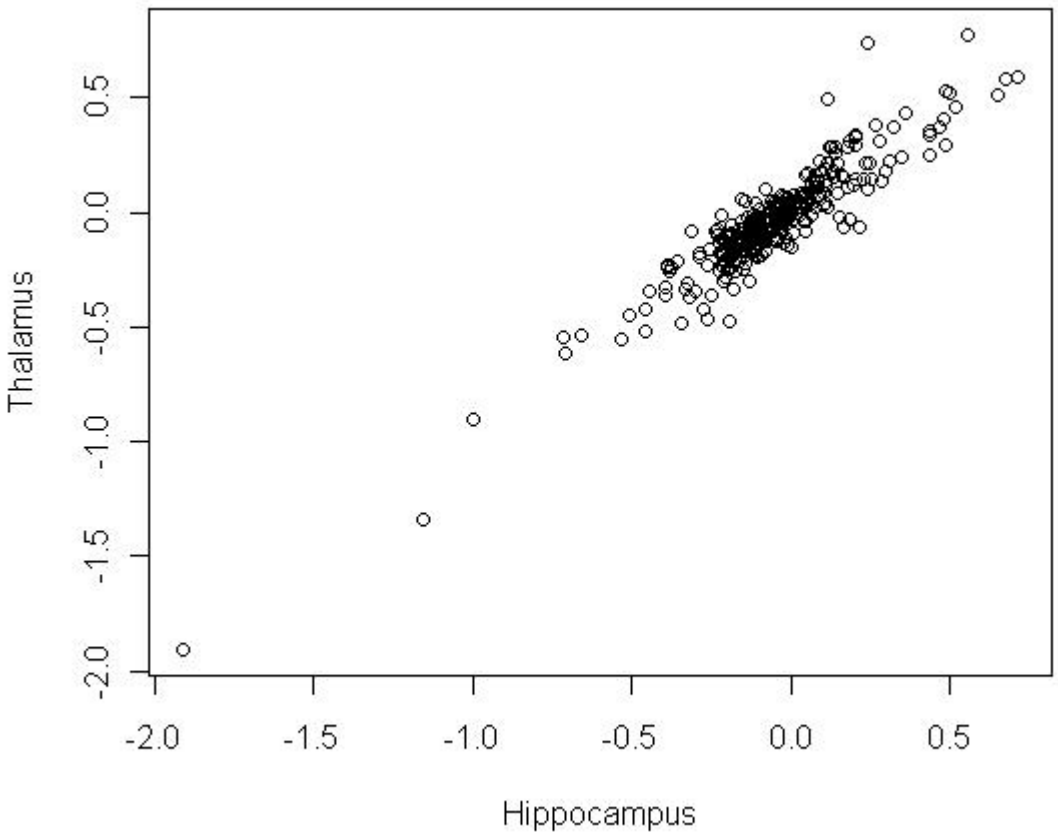
- A scatter plot is ideal for the visualization of the correlation between two variables



Using the `plot()` Function

- Plot t
  Hippo
  Thala

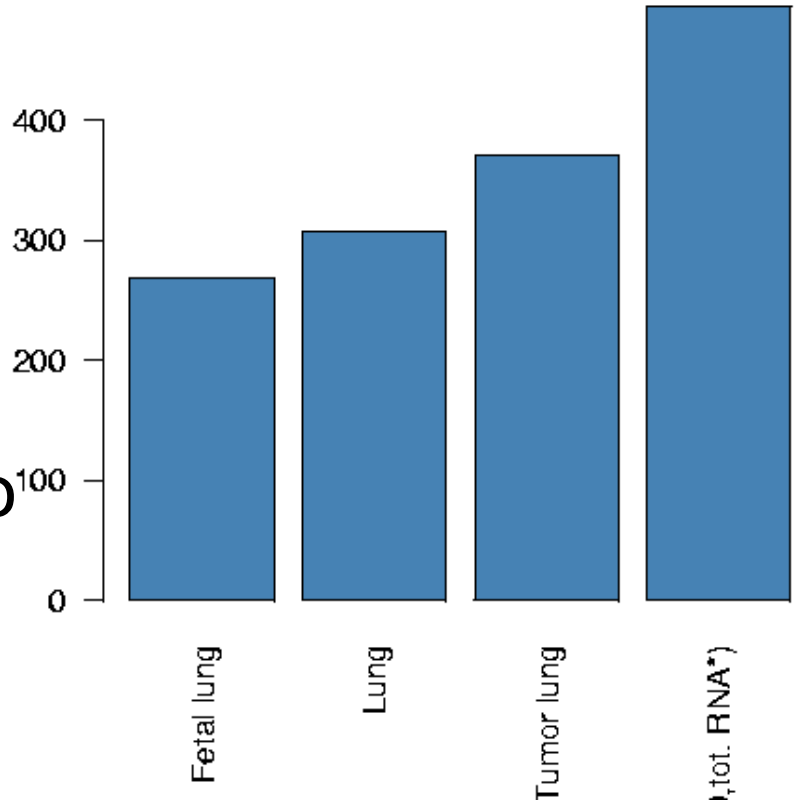
- `> pl`
  `$Hip`
  `$Tha`
  `ylab`



`s`
`ampus",`

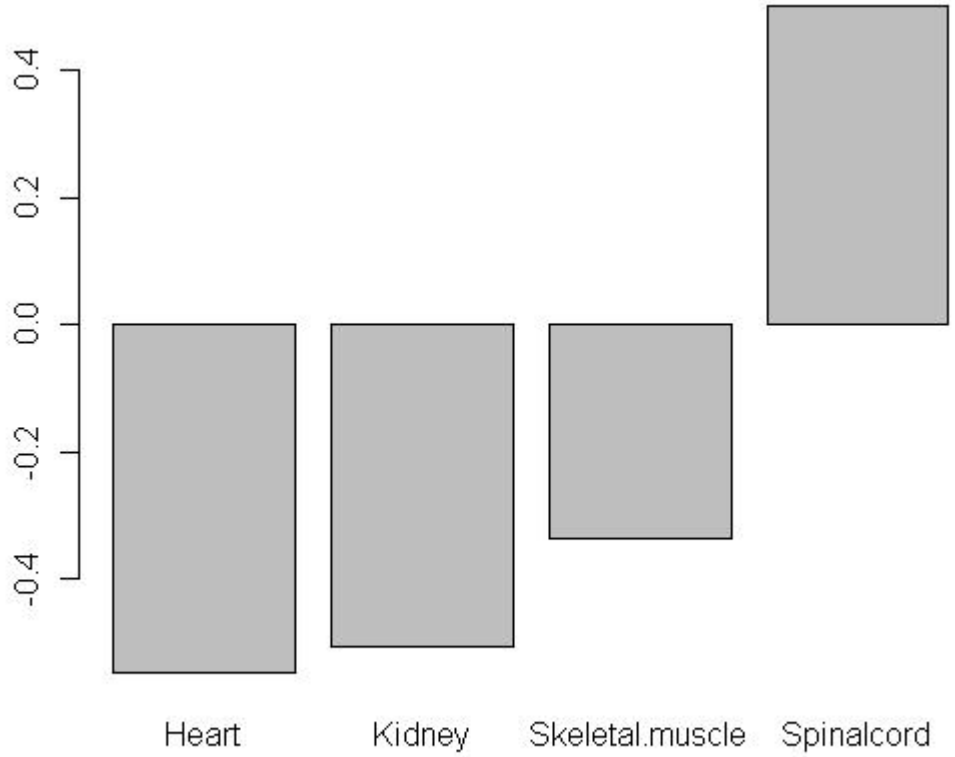
Plotting – Bar Plot

- How does the expression profile of “NOVA1” differ across several tissues?
- A bar plot can be used to compare two or more categories



Using the `barplot()` Function

- Compa… d, Kidney, Heart … plot
- Sort th… `()` function
- `barpl…` class
- `> tis…` `.dney",` `"Skel…` `>barp…` `3,tissues]` `))",t…`



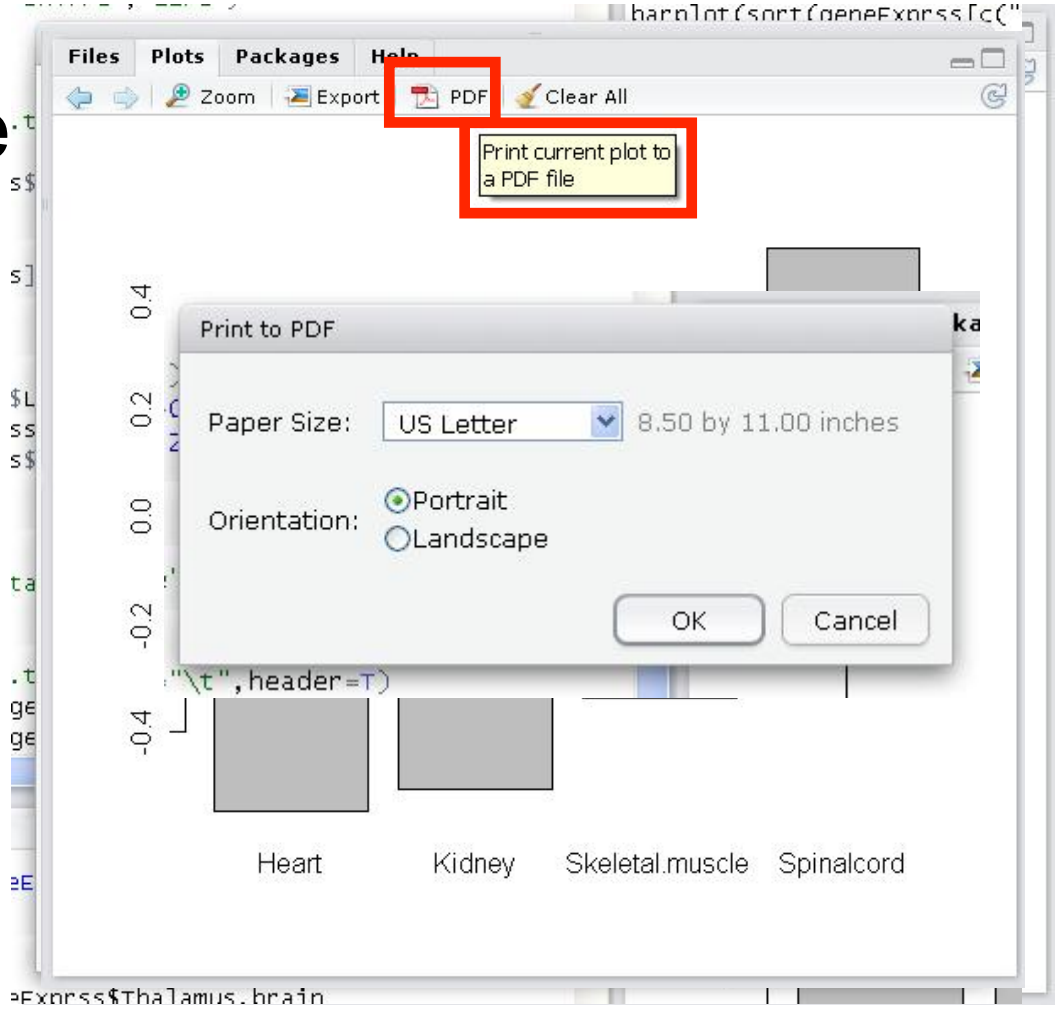
More Graphic Functions to Keep in Mind

- `hist()`
- `boxplot()`
- `plotmeans()`
- `scatterplot()`

Save Plot to File - RStudi

- Cre

  - Create



Save Plot to File in 

For example:

- Before running the visualizing function, redirect all plots to a file of a certain type

```
> geneExprss = read.table(file = "geneExprss.txt", sep =
"\t",header = T)
> tissues = c ("Spinalcord", "Kidney",
"Skeletal.muscle", "Heart")
> pdf("Nova1BarPlot.PDF")
> barplot(as.matrix(geneExprss[143,tissues]))
> graphics.off()
```

– jpeg(filename)

– png(filename)

– pdf(filename)

– postscript(filename)

- After running the visualization function, close graphic device using **dev.off()** or **graphcis.off()**

Lecture Overview

- What is R and why use it?
- Setting up R & RStudio for use
- Calculations, functions and variable classes
- File handling, plotting and graphic features
- Statistics
- Packages and writing functions

Statistics – `cor.test()`

- A few slides back we compared the expression profiles of the Hippocampus.brain and the Thalamus.brain

```
> geneExprss = read.table (file = "geneExprss.txt", sep = "\t", header = T)
```

- But is that correlation statistically significant?

```
> cor.test ( geneExprss $Hippocampus.brain, geneExprss $Thalamus.brain, method = "pearson")
```

- R can help with this sort of question as well

```
> cor.test ( geneExprss $Hippocampus.brain, geneExprss $Thalamus.brain, method = "spearman")
```

- To answer that specific question we'll use the `cor.test()` function

Statistics – More Testing

- `t.test()` # Student t test
- `wilcox.test()` # Mann-Whitney test
- `kruskal.test()` # Kruskal-Wallis rank sum test
- `chisq.test()` # chi squared test
- `cor.test()` # pearson / spearman correlations

- `lm(), glm()` # linear and generalized linear models
- `p.adjust()` # adjustment of P-values for multiple testing (multiple testing correction) using FDR, bonferroni, etc.

Statistics – Examine the Distribution of Your Data

- Use the `summary()` function

- `> geneExprss = read.table (file = "geneExprss.txt", sep = "\t", header = T)`

- `> summary(geneExprss$Liver)`
- `Min. -1.84400`
  `1st Qu. -0.17290`
  `Median -0.05145`
  `Mean -0.08091`
  `3rd Qu. 0.05299`
  `Max. 0.63950`

Statistics – More Distribution Functions

- `mean()`
- `median()`
- `var()`
- `min()`
- `max()`
- When using most of these functions remember to use argument `na.rm=T`

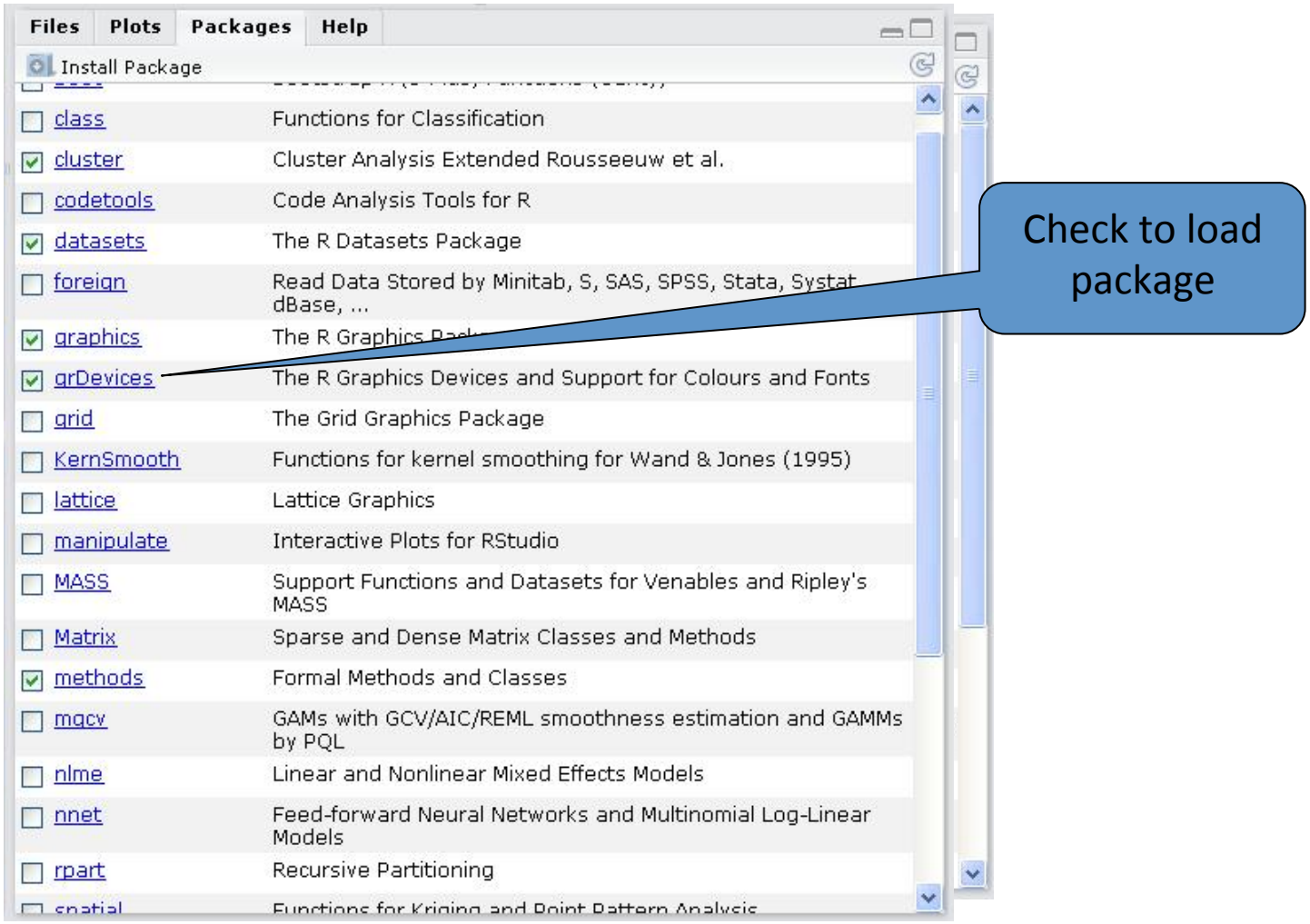
Lecture Overview

- What is R and why use it?
- Setting up R & RStudio for use
- Calculations, functions and variable classes
- File handling, plotting and graphic features
- Statistics
- Packages and writing functions

Functions & Packages

- All operations are performed by functions
- All R functions are stored in packages
- Base packages are installed along with R
- Packages including additional functions can by downloaded by user
- Functions can also be written by user

Install & Load Packages - RStudio



Install & Load  Packages

- Use the functions:
  - `Install.packages("package_name”)`
  - `update.packages(“package_name”)`
  - `library(package_name) # Load a package`

Final  Tips

- Reading the functions' help file (`> ?function_name`)
  - Run the help file examples
- Use http://www.rseek.org/
- Google what you're looking for
- Post on the R forum webpage
- And most importantly – play with it, get the hang of it, and do NOT despair ☺

Acknowledgement

Dror Hollander
Gil Ast Lab

Jeremy Baxter, Rhodes University