Nicolás Campione

Basic R Workshop

March 6, 2014

Uppsala University, Geocentrum

http://nicolascampione.weebly.com/r-worshop.html

What is R? Who created R?

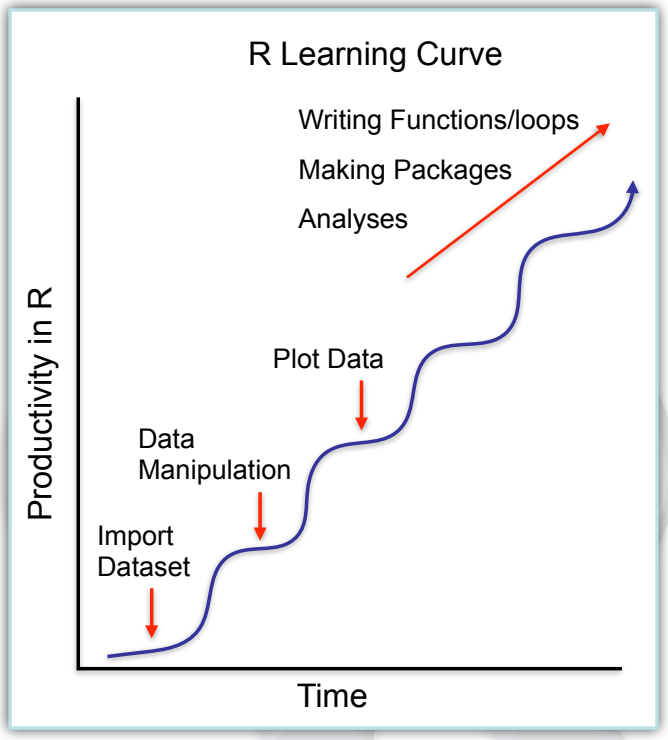
- Computer language for statistics and graphics
- developed in early 90s
  - Ross Ihaka (New Zealand)
  - Robert Gentleman (Canada)
- based on the S programming language
  - John Chambers while at Bell Labs (where the laser was invented)
- R's name comes from creators and association with S
- R is the free version of S (now S-Plus)

What can R do?

- Data Manipulation
  - subsetting, subsampling, and randomization
  - transformation (e.g., logarithm)
  - maths
- Statistical techniques:
  - linear and non-linear modelling
  - standard statistics (t-test, ANOVA, etc.)
  - univariate, bivariate, and multivariate
  - so much more!!!!
- Graphical techniques:
  - 2D and 3D graphics
  - highly customisable

Advantages and uses of R

- IT'S FREE!!!
- Runs across all platforms
- Environment-based system:
  - customisable to the core
  - modify subsequent analyses
  - scripts allow automation
- Latest functions available
- Publication quality figures
- One disadvantage:
  Steep learning curve(s)

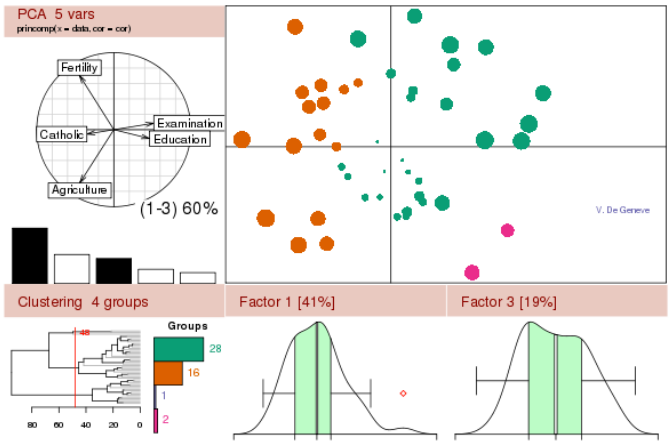


The R Project for Statistica

www.r-project.org

Apps | Google | Entertainments | Finance | Museum | Shopping | Travel | Resources: Personal | Resources: Academic | Resources: Paleo | Other Bookmarks

The R Project for Statistical Computing



About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

Misc
Bioconductor
Related Projects
User Groups
Links

Getting Started:

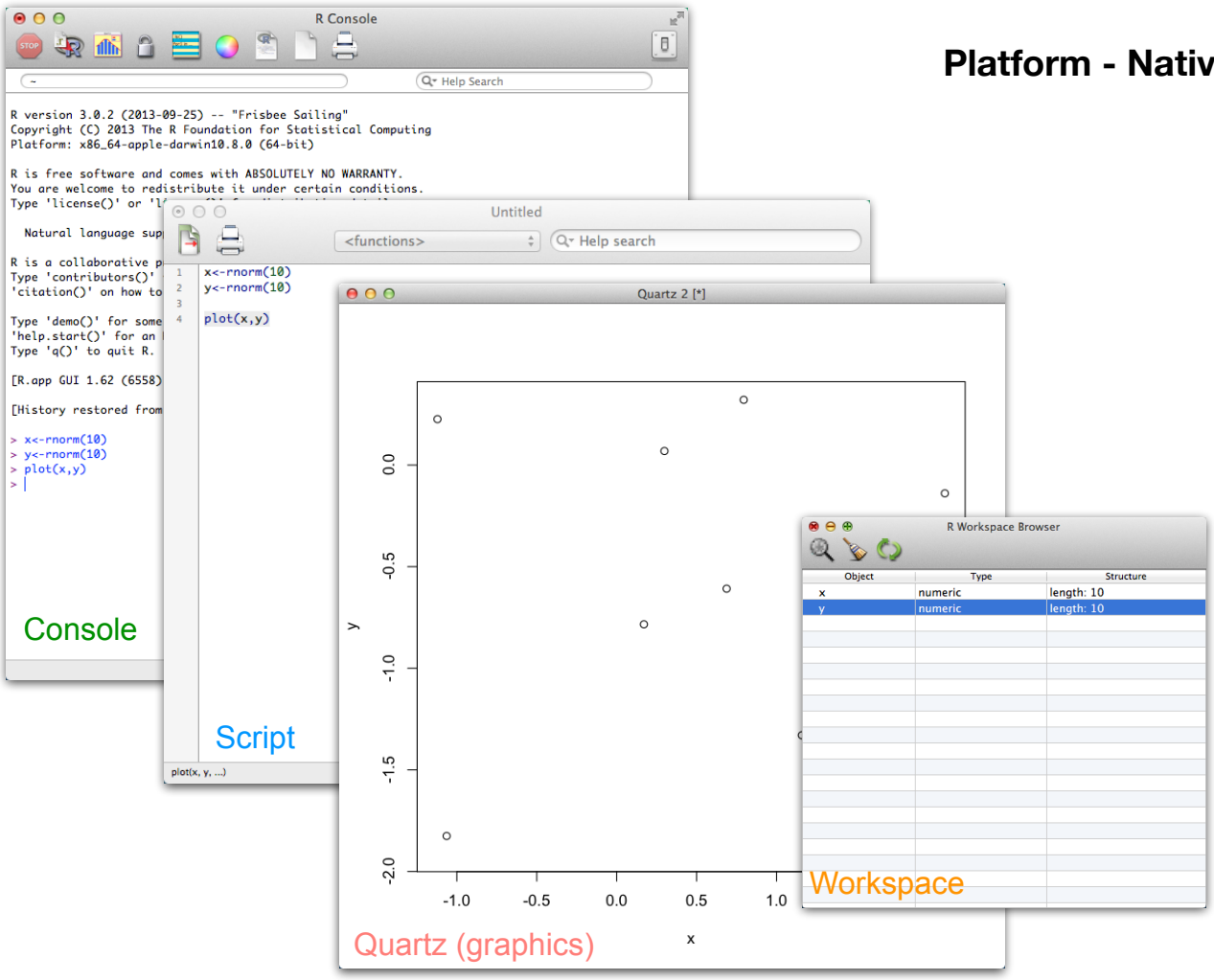
- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

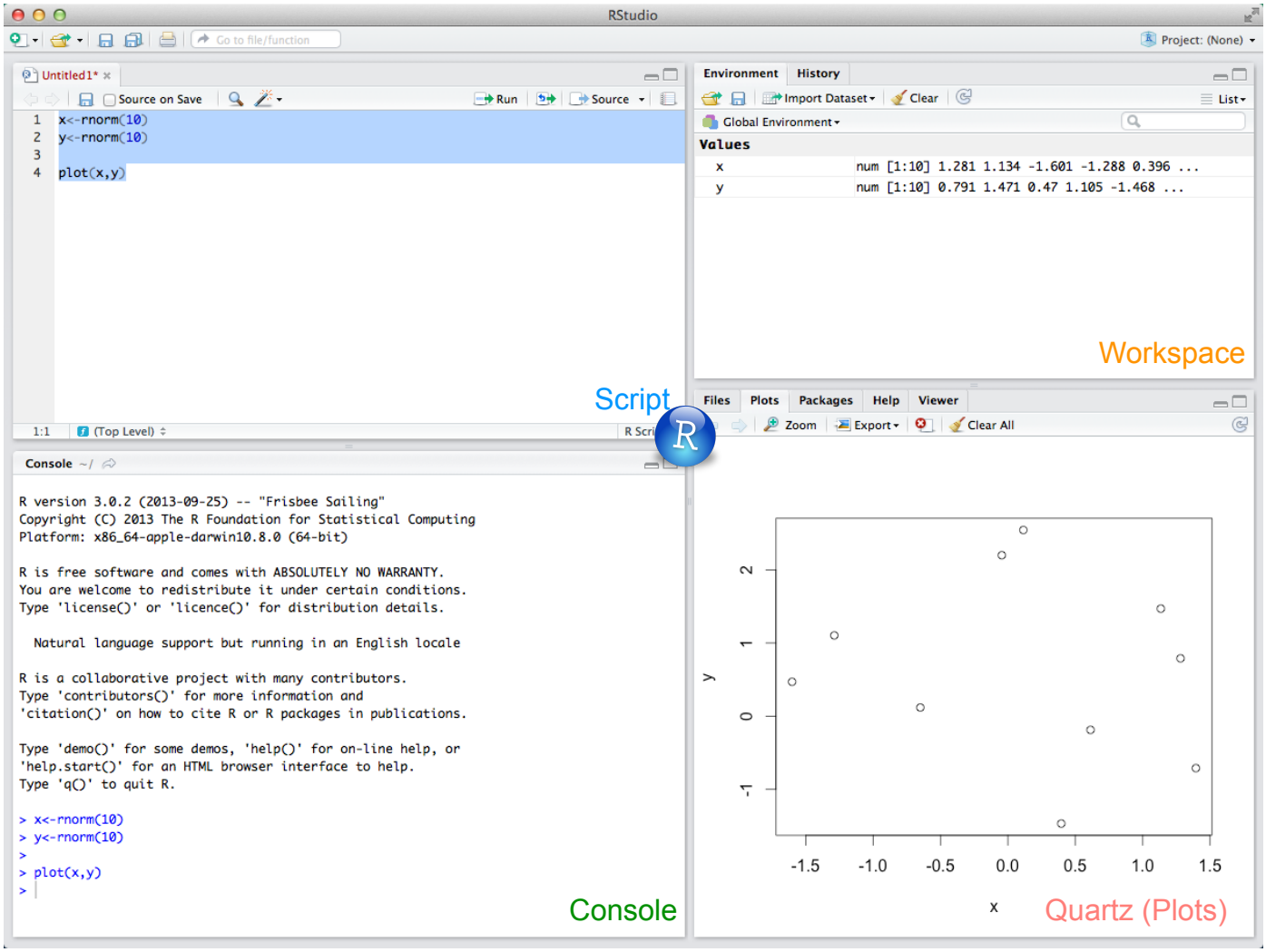
News:

- **R version 3.0.2** (Frisbee Sailing) has been released on 2013-09-25.
- **useR! 2013**, took place at the University of Castilla-La Mancha, Albacete, Spain, July 10-12 2013.
- **The R Journal Vol.5/1** is available.
- **R version 2.15.3** (Security Blanket) has been released on 2013-03-01.

This server is hosted by the Institute for Statistics and Mathematics of WU (Wirtschaftsuniversität Wien).

Platform - Native




RStudio
Untitled1*
Source on Save
Run
Source
1 x<-rnorm(10)
2 y<-rnorm(10)
3
4 plot(x,y)
Script
Environment
History
Import Dataset
Clear
Global Environment
Values
x num [1:10] 1.281 1.134 -1.601 -1.288 0.396 ...
y num [1:10] 0.791 1.471 0.47 1.105 -1.468 ...
Workspace
Files
Plots
Packages
Help
Viewer
Zoom
Export
Clear All
Console
R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (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.
> x<-rnorm(10)
> y<-rnorm(10)
> plot(x,y)
Console
y
x
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
Quartz (Plots)

Workshop

- Playing with R
  - objects
  - class and data types
  - manipulating data
  - functions
  - libraries
- Preparing data for R
- Setting up a working directory
- Plotting data
- Customising R
  - making functions
  - making loops

Morning

Afternoon

notes - #Courier

code - Courier

answers - Courier

Console and Objects

```
> 5
[1] 5
> 7+6
[1] 13
> xvar<-1:8
> xvar
[1] 1 2 3 4 5 6 7 8
> xvar*2
[1] 2 4 6 8 10 12 14 16
> lab<-"dinosaur"
> lab
[1] "dinosaur"
> class(xvar)
[1] "integer"
> class(lab)
[1] "character"
```

Exercise 1

1. Assign the value 50 to an object called femtio
2. Assign the value 16 to an object called sexton
3. Add femtio and sexton directly into an object called sum
4. What is sum?
5. What is the class of sum?

Variable Types

Class	Example
integer	1, 2, 3, 4, 5, 6, 7, 8
numeric	-1.086, -0.105, 1.471
logical	TRUE, FALSE
character	“dinosaur”, “sp.1”
factor	“male”, “female”
complex	imaginary numbers

Variable Types

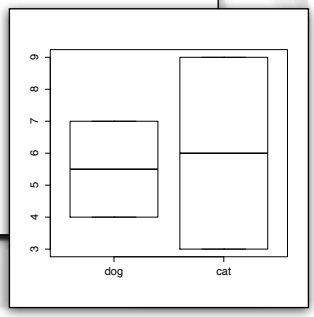
Character

```
> x<-c("dog","cat","cat","dog")
> x
[1] "dog" "cat" "cat" "dog"

> length(x) #number of values
[1] 4
> x=="dog"
[1] TRUE FALSE FALSE TRUE
> as.numeric(x)
ERROR
> plot(x,c(7,9,3,4))
ERROR
```

Factor

```
> x<-as.factor(x)
> x
[1] dog cat cat dog
Levels: dog cat
> length(x)
[1] 4
> x=="dog"
[1] TRUE FALSE FALSE TRUE
> as.numeric(x)
[1] 1 2 2 1
> plot(x,c(7,9,3,4))
```



Data Types

Class
vector
matrix
data.frame
list
array

```
> vt<-c(3,1,7,30)
> length(vt)
[1] 4
#Extracting & Subsetting
> vt[3]
[1] 7
> vt[2:4]
[1] 1 7 30
> vt[c(1,4)]
[1] 3 30
> sum(vt)
[1] 41
```

Data Types

Class

vector

matrix

data.frame

list

array

```
> mt<-matrix(c(x,vt),nrow=4)
> mt
     [,1] [,2]
[1,]    2    3
[2,]    1    1
[3,]    1    7
[4,]    2   30
#Extracting & Subsetting
> mt[1,2]
[1] 3
> mt[,3]
[1] 3 1 7 30
> mt[2,]
[1] 1 1
> nrow(mt) #number of rows
> ncol(mt) #number of columns
> dim(mt) #number of rows
[1] 4 2
```

Data Types

Class

vector

matrix

data.frame

list

array

```
> df<-data.frame(animal=x,var.1=vt)
> df
  animal var.1
1    dog     3
2    cat     1
3    cat     7
4    dog    30
#Extracting & Subsetting
> df[3,]
  animal var.1
3    cat     7
> df$animal
[1] dog cat cat dog
Levels: cat dog
#Access Variables Directly
> attach(df)
> var.1
[1]  3  1  7 30
```

Data Types

Class
vector
matrix
data.frame
list
array

```
> lt<-list(matrix=mt,data.frame=df)
> lt
$matrix
     [,1] [,2]
[1,]    2    3
[2,]    1    1
[3,]    1    7
[4,]    2   30

$data.frame
  animal var.1
1    dog     3
2    cat     1
3    cat     7
4    dog    30
```

Data Types

Class
vector
matrix
data.frame
list
array

```
#Extracting & Subsetting
> lt$matrix[3,2]
[1] 7
> lt$data.frame$var.1
[1]  3  1  7 30
#Access Items Directly
> attach(lt)
> matrix
     [,1] [,2]
[1,]    2    3
[2,]    1    1
[3,]    1    7
[4,]    2   30
```

Data Types

- Extension of ‘matrix’, can have three dimensions.

Class

- vector
- matrix
- data.frame
- list
- array

```
> ar<-array(c(mt,mt*2),dim=c(4,2,2),
            dimnames=list(1:4,
            c("animal","var.1"),
            c("Group 1","Group 2")))
> ar
, , Group 1

  animal var.1
1      2     3
2      1     1
3      1     7
4      2    30

, , Group 2

  animal var.1
1      4     6
2      2     2
3      2    14
4      4    60
```


Data Types

- Extension of ‘matrix’, can have three dimensions.

Class

- vector
- matrix
- data.frame
- list
- array


```
#Extracting & Subsetting
> ar[,2,]
  Group 1 Group 2
1       3       6
2       1       2
3       7      14
4      30      60
> ar[3,,]
        Group 1 Group 2
animal        1       2
var.1         7      14
> ar[3,2,]
Group 1 Group 2
      7      14
> ar[,"var.1",]
```

Data Types

Class
vector
matrix
data.frame
list
array

- Extension of ‘matrix’, can have three dimensions.

```
#Arithmetic
> sum(ar)
[1] 141
> colSums(ar)
       Group 1 Group 2
animal       6      12
var.1       41      82
> colSums(ar,dims=2)
Group 1 Group 2
     47      94
> rowMeans(ar)
    1     2     3     4
 3.75  1.50  6.00 24.00
> rowMeans(ar,dims=2)
  animal var.1
1    3.0   4.5
2    1.5   1.5
3    1.5  10.5
4    3.0  45.0
```

Exercise 2

1. Make a vector called vektor with the values:
   31 6 43 25 18 31 3 28 34 12 15 21 37 40 9
2. Convert vektor into a 5x3 matrix called matris.
3. Extract the first 4 rows and 2 columns of matris into an object called mini.matris.
4. What is the sum of each rows of mini.matris?
5. Which row of mini.matris includes a value of 3?

Subset by Values

Action	Symbol	
equal/not equal to:	==, !=	Numeric Character Factor
greater/less than	>, <	Numeric
greater/less than and equal to	>=, <=	Numeric
and/or	&, |	Combines Actions

Subsetting by Values

```
> df
  animal var.1
1    dog     3
2    cat     1
3    cat     7
4    dog    30
> df==3
     animal var.1
[1,]  FALSE  TRUE
[2,]  FALSE FALSE
[3,]  FALSE FALSE
[4,]  FALSE FALSE

> df$var.1>1&df$var.1<20
[1]  TRUE FALSE  TRUE FALSE

> df$var.1>20|df$var.1<5
[1]  TRUE  TRUE FALSE  TRUE
```

```
> df!="cat"
     animal var.1
[1,]   TRUE  TRUE
[2,]  FALSE  TRUE
[3,]  FALSE  TRUE
[4,]   TRUE  TRUE

> df=="cat"|df=="dog"
     animal var.1
[1,]   TRUE FALSE
[2,]   TRUE FALSE
[3,]   TRUE FALSE
[4,]   TRUE FALSE

> df$animal=="cat"&df$var.1==7
[1] FALSE FALSE  TRUE FALSE
```

Subsetting by Values

```
> which(df$animal=="cat"&df$var.1==7)
[1] 3 #row number corresponding to these values

> ex1<-which(df$animal=="cat"&df$var.1==7)
> df[ex1,]
  animal var.1
3    cat     7

> ex2<-which(df$animal=="dog")
> df[ex2,]
  animal var.1
1    dog     3
4    dog    30
```

Functions

- Compilation of R code meant to carry out a specific task
- Specifics (arguments) are defined within ( )
- Most are built-in to R, but you can make your own

```
#we've seen some already
c() #puts values together
class() #returns class type
length() #number of values
as.numeric() #change class type
as.factor() #change class type
as.matrix() #change class type
as.data.frame() #change class type
matrix() #make matrix
data.frame() #make data frame
list() #make list
array() #make array
attach() #attach names
```

```
plot() #plot data
sum() #sums values
nrow() #number of rows
ncol() #number of columns
dim() #returns dimensions
colSums() #sums column values
rowSums() #sums row values
colMeans() #average of columns
rowMeans() #average of rows
which() #returns placement of value
#assign names to rows or columns
rownames()<-names.vector
colnames()<-names.vector
```

Exercise 3

1. Make a character vector called dino with the names:
   Tyrannosaurus
   Triceratops
   Edmontosaurus
   Troodon
2. Convert mini.matris into a data.frame called data.
3. Combine dino with data into an object called dataset.
4. Assign the columns names:
   Dino
   skull.length
   jaw.length
5. Which dino(s) has(ve) a skull length greater than 20 with a jaw length less than or equal to 31?

Functions

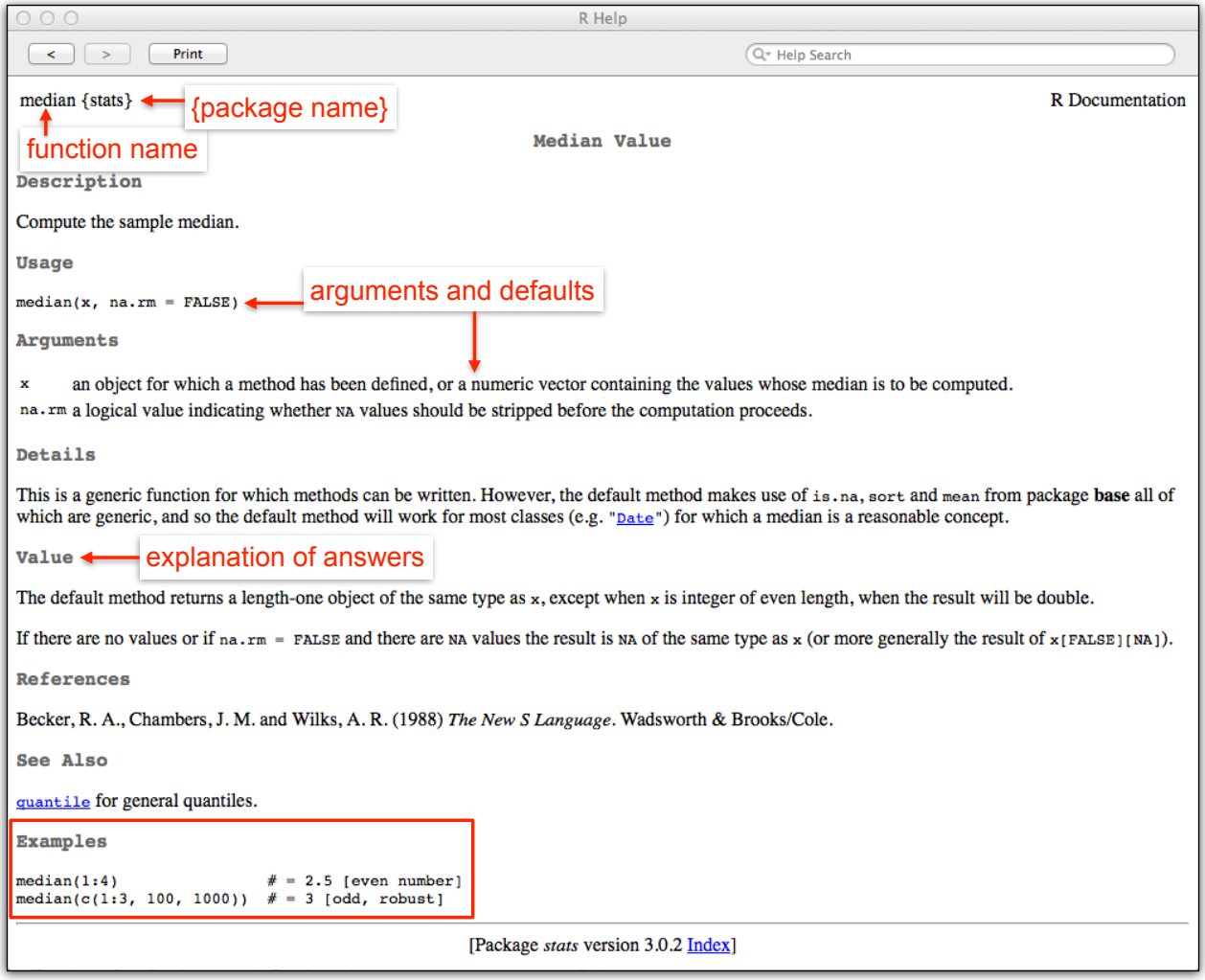
- Compilation of R code meant to carry out a specific task
- Specifics (arguments) are defined within ( )
- Most are built-in to R, but you can make your own

- Each function has:
  1. required arguments
- May have:
  2. default arguments
  3. optional arguments

Specified in the **help files** associated with each function

```
#only open packages
> ?sum
> help("sum")
```

```
#all downloaded packages
> ??sum
> help.search("sum")
```



UPPSALA UNIVERSITET

Help Files

- must know the name of the function in order to search
- if not, use `??function` or `help.search("function")`
- this only searches open or downloaded packages
- in order to search for other functions, use internet:
    - google or CRAN

Packages

- Collection of functions, generally created by a single research group
- Functions may be related
- Some packages are defaults for R and open automatically when R is opened:
  - e.g. datasets, graphics, stats, etc.
- Others have to be installed (done once) and then loaded (done every time R is opened)

```
#install package called geiger (Analysis of evolutionary diversification)
> install.packages("geiger")
#access functions in package
> library(geiger)
#for list of function within the package
> library(help="geiger") #loading not necessary, but must be installed
```

Sourcing

- access to functions not uploaded to CRAN
  - personal functions
  - other internet functions

http://nicolascampione.weebly.com/quantitative-methods.html

```
#source from the web
source("http://nicolascampione.weebly.com/uploads/1/9/4/1/19411255/mch.r")
#source from your computer
source("~/Dropbox/Software/RScripts/MCH.R")
```

Preparing Data of R

- spreadsheet editor
  - Excel
  - OpenOffice
- data should be formatted as a rectangular matrix (n x q)
- exported as plain text

  .txt ✓ .csv ✓ .xls ✗ .xslx ✗ .ods ✗
- **N.B.** make first row a header
- first column can also be row names (must be unique)
- **N.B.** avoid spaces
- saving as .csv may be different depending on regional preferences

Importing Data into R

- set a working directory
- houses the necessary data, scripts, functions, etc.
  - windows: File >> Change dir
  - mac: Misc >> Change Working Directory
  - R-studio: Session >> Set Working Directory >> Choose Directory
- **N.B.** I recommend R-Studio, use project option
  - Project: (None)>>New Project>>Existing Directory>>Browse

```
getwd() #leave brackets empty, returns current working directory
setwd("folder extension") #steps above do this for you
```

Importing Data into R

```
#load data to R

> data<-read.table("data.txt") #for space delineated files *

> data<-read.csv("data.csv") #for comma delineated files (NA)

> data<-read.csv2("data.csv") #for comma delineated files (Eur)

> data<-read.delim("data.txt") #for tab delineated files (NA)

> data<-read.delim2("data.txt") #for tab delineated files (Eur)
```

* spaces between words and within a cell are not allowed

Importing Data into R

```
> data #show in console
> View(data) #show in a separate window
> fix(data) #allows you to edit data in R *

#Example Dataset
> data<-read.csv2("http://nicolascampione.weebly.com/uploads/
1/9/4/1/19411255/dataset.csv")
> View(data) #look at dataset-tetrapod limb measurements, see next slide
#Subset the same as before
> which(data$Family=="Felidae") #row numbers corresponding to felids
> cats<-data[which(data$Family=="Felidae"),] #dataset of just felids
#dataset for felids over 50kg
> big.cats<-data[which(data$Family=="Felidae"&data$Body.Mass..g.>50000),]
> View(big.cats)
```

	row.names	Higher.Clade	Family	Species	Common.Name	SP.	Body.Mass..g.	Humerus.Length	Humerus.Circumference	Femur.Length	Femur.Circumference
1	15	Carnivora	Felidae	Acinonyx jubatus	Cheetah	ROM 111488	60400	243.5	62.5	268.5	67.5
2	22	Carnivora	Felidae	Panthera leo	Lion	ZMUC 7231-CH2005	203000	366.0	116.0	401.0	109.0
3	23	Carnivora	Felidae	Panthera onca	Jaguar	ZMUC 6221-CH2005	71000	239.0	76.0	265.5	72.5
4	24	Carnivora	Felidae	Panthera pardus	Leopard	ZMUC 5661-CH2005	61000	200.0	62.0	231.0	61.0
5	25	Carnivora	Felidae	Panthera tigris_altaica	Siberian tiger	ZMUC 5698-CH2005	230000	350.0	113.0	411.0	102.5
6	26	Carnivora	Felidae	Panthera tigris_tigris	Bengal tiger	ZMUC 5667-CH2005	145000	310.0	94.0	360.5	90.5

* does not save changes to the .csv or .txt file

Column Names

	Higher.Clade	Family	Species	Common.Name	SP.	Body.Mass..g.	Humerus.Length	Humerus.Circumference	Femur.Length	Femur.Circumference
1	Afrotheria	Elephantidae	Elephas maximus	Indian elephant	ZMUC 1399	3534000	830	310	980	315
2	Afrotheria	Elephantidae	Loxodonta africana	African elephant	ROM R6000	6435000	1035	416.3	1147.5	399
3	Afrotheria	Macroscelididae	Petrodromus tetradactylus	Four-toed Elephant Shrew	ROM 85723	275	33.7	9.5	47.3	12.5
4	Afrotheria	Macroscelididae	Rhynchocyon cirnei	Checkered Elephant Shrew	ROM 47023	445	NA	9.5	55.45	13.75
5	Afrotheria	Procaviidae	Heterohyrax brucei	Yellow-spotted Rock Hyrax	ROM 64992	3650	78.2	21.75	79.5	24.25
6	Carnivora	Canidae	Alopex lagopus	Arctic Fox	ROM 105014	3710	112.15	23	114.68	24.85
7	Carnivora	Canidae	Canis latrans	Coyote	ROM R6654	13608	166.2	41	178.35	40.6
8	Carnivora	Canidae	Canis lupus	Wolf	ROM 101841	32000	211	53.25	226.5	51.75
9	Carnivora	Canidae	Cuon alpinus	Dhole	ROM 105074	15400	158.3	42.35	173.75	44.15
10	Carnivora	Canidae	Nyctereutes procyonoides	Racoon Dog	ROM 94037	3500	84.3	25.25	89.74	25.75
11	Carnivora	Canidae	Urocyon cinereoargentus	Gray fox	ROM 14309	4800	102.74	24.1	115	27.15
12	Carnivora	Canidae	Vulpes velox	Swift Fox	ROM 105398	2300	99.88	21.55	99.17	23.3
13	Carnivora	Canidae	Vulpes vulpes	Red Fox	ROM 28025	6000	125.1	26.25	127.62	26.95
14	Carnivora	Canidae	Vulpes zerda	Fennec Fox	ROM 91467	1000	68.2	16.25	70.4	16.25
15	Carnivora	Felidae	Acinonyx jubatus	Cheetah	ROM 111488	60400	243.5	62.5	268.5	67.5
16	Carnivora	Felidae	Felis catus	Domestic cat	ZMUC 3629-CH2005	8400	104	28.5	119.5	30
17	Carnivora	Felidae	Leopardus pardalis	Ocelot	ZMUC 1111-CH2005	13900	144.5	40.5	165	42
18	Carnivora	Felidae	Lynx canadensis	Canadian Lynx	ROM 74772	17710	178.75	38.75	215.55	40.85
19	Carnivora	Felidae	Lynx lynx	Eurasian lynx	ZMUC 120-CH2005	9700	170.5	38	202.5	39
20	Carnivora	Felidae	Lynx rufus	Bobcat	ROM 67947	8800	151.25	36.5	175.95	38.25
21	Carnivora	Felidae	Neofelis nebulosa	Cloud lepard	AHR1985	13478	NA	44.7	NA	41.4
22	Carnivora	Felidae	Panthera leo	Lion	ZMUC 7231-CH2005	203000	366	116	401	109
23	Carnivora	Felidae	Panthera onca	Jaguar	ZMUC 6221-CH2005	71000	239	76	265.5	72.5
24	Carnivora	Felidae	Panthera pardus	Leopard	ZMUC 5661-CH2005	61000	200	62	231	61
25	Carnivora	Felidae	Panthera tigris_altaica	Siberian tiger	ZMUC 5698-CH2005	230000	350	113	411	102.5
26	Carnivora	Felidae	Panthera tigris_tigris	Bengal tiger	ZMUC 5667-CH2005	145000	310	94	360.5	90.5
27	Carnivora	Felidae	Panthera unica	Snow Leopard	ZMUC 6047-CH2005	43100	208.5	63.5	234.5	63
28	Carnivora	Felidae	Puma concolor	Cougar	ZMUC 5663-CH2005	47000	231.5	65.5	276.5	62.5
29	Carnivora	Herpestidae	Crossarchus obscurus	Common Kusimanse	ROM 89636	1190	48.86	14.9	54.76	15.5
30	Carnivora	Herpestidae	Galerella sanguinea	Slender Mongoose	ROM 58390	1100	45.5	14	51.25	14.5
31	Carnivora	Herpestidae	Helogale parvula	Common Dwarf Mongoose	ROM 58387	450	34.6	11	36	12
32	Carnivora	Herpestidae	Herpestes javanicus	Small Asian Mongoose	ROM 74045	384	42.2	11.6	46.84	12.5
33	Carnivora	Herpestidae	Ichneumia albicauda	White-tailed Mongoose	ROM 58383	4760	93.3	23.1	108.72	27.05
34	Carnivora	Herpestidae	Liberiictis kuhni	Liberian Mongoose	ROM 102286	2430	68	20.5	76.1	22.25
35	Carnivora	Herpestidae	Suricata suricata	Meerkat	ROM 74547	717	50.74	13.2	55.83	13.65
36	Carnivora	Mustelidae	Arctonyx collaris	Hog Badger	ROM 97264	5900	96.56	32.1	109.04	29.3
37	Carnivora	Mustelidae	Gulo gulo	Wolverine	ROM 32287	16556	136.25	39.75	141.5	35
38	Carnivora	Mustelidae	Lutra canadensis	Otter	ROM R6629	6400	74.66	26.1	73.69	24.4

Row Numbers

class-factor

class-numeric

UPPSALA UNIVERSITET

Exercise 4

1. Set a working directory on your computer.
2. If you have a dataset. Save it using the appropriate file extension and move it into your working directory.
3. Return to R (R-studio) and upload your dataset.
4. Did it work? If not, open the file in textedit (Mac) or NotePad (Windows)
5. Upload my example dataset from the internet:

```
data<-read.csv2("http://nicolascampione.weebly.com/uploads/1/9/4/1/19411255/dataset.csv")
```

Other Useful Functions for Manipulating Datasets

```
> which(data$ID==???) #calling out particular data
> attach(data) #attach column names
#apply functions repeat a function for a subset of the data
#apply acts on the rows (MARGIN=1) or columns (MARGIN=2)
> apply(data[,6:10],MARGIN=2,mean)
Body.Mass..g.  Humerus.Length  Humerus.Circumference  Femur.Length  Femur.Circumference
103597.54569   NA              39.15004               NA            39.51167
> apply(data[,6:10],MARGIN=2,mean,na.rm=TRUE)
Body.Mass..g.  Humerus.Length  Humerus.Circumference  Femur.Length  Femur.Circumference
103597.54569   104.81064       39.15004               121.60798     39.51167
#tapply acts on subsets specified by a factor
> tapply(data$Body.Mass..g.,data$Higher.Clade,mean)
Afrotheria     Amphibia     Carnivora     Euarchonta     Eulipotyphla   Glires
1994674.0000   155.7875     47132.9583    15094.9333     369.5000       2111.5742
Marsupialia    Reptilia     Ungulata      Xenarthra
7103.6214      6723.4734    325565.1220   9749.4000
```

missing data

Missing Data (NAs)

```
> View(data) #scroll through the dataset
```

- NAs are present in two of the variables
- many functions **cannot run** if you have missing data

1. omit NAs prior to running functions using `na.omit` or `na.exclude`
   - some functions remove them automatically (e.g., `plot`)
   - some functions have arguments that deal with missing data:

```
> sum(x,na.rm=TRUE) #x is a vector missing at least one value
> mean(x,na.rm=TRUE)
> cor(x,y,use="complete.obs")
```

2. Estimate missing data:
   - Packages: LOST, pcaMethods

Exercise 5

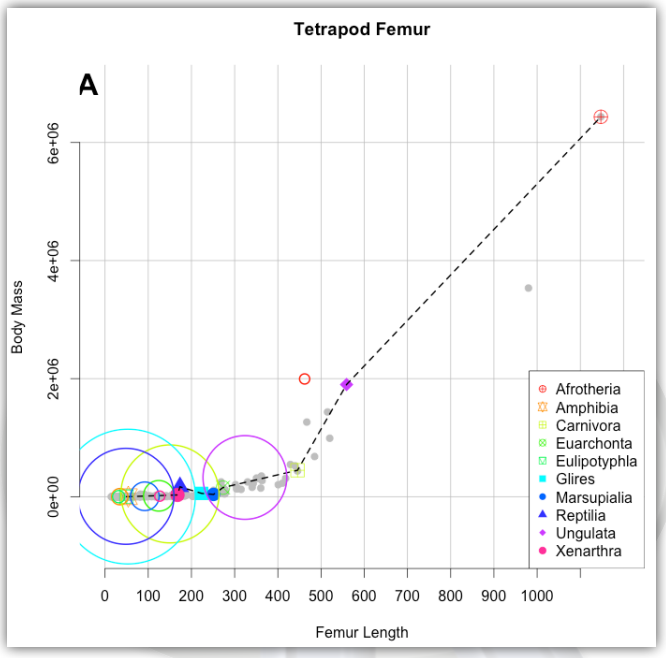
1. Spend the next little while using the various functions you have learned to manipulate my dataset and/or yours.

2. Try the functions:
   attach
   apply
   tapply
   which

3. From my dataset, extract the reptiles less than 5kg.

Plotting Data

- highly customisable
- publishable quality
- can be exported as:
  - .pdf, .eps, .svg
- subsequent vector-based editing can be done in Adobe Illustrator or Inkscape (free)
- NB. Can add to plot, but cannot undo



Plotting Data

- Anatomy of `plot`

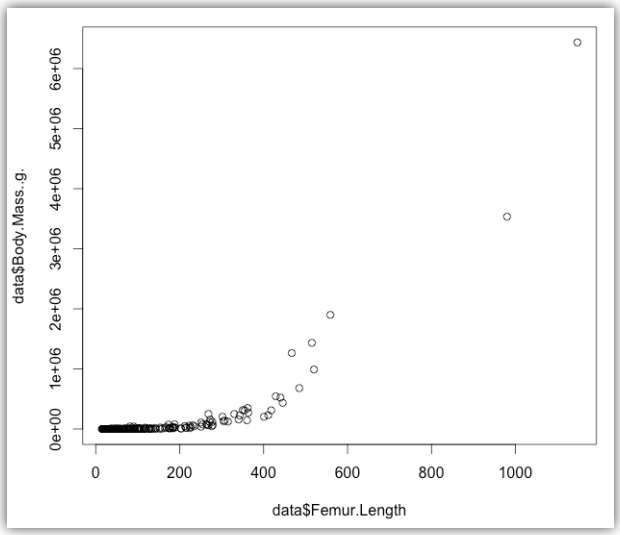
```
> plot(x, y, ...)
> plot(data$Femur.Length,data$Body.Mass..g.)
#...
> plot(x,y,type,main,sub,xlab,ylab,...) #see next slide
```

- type: `"p"` — points
  `"l"` — lines
  `"b"` — both
- main: plot title
- sub: sub title (bottom)
- xlab: x-axis label
- ylab: y-axis label

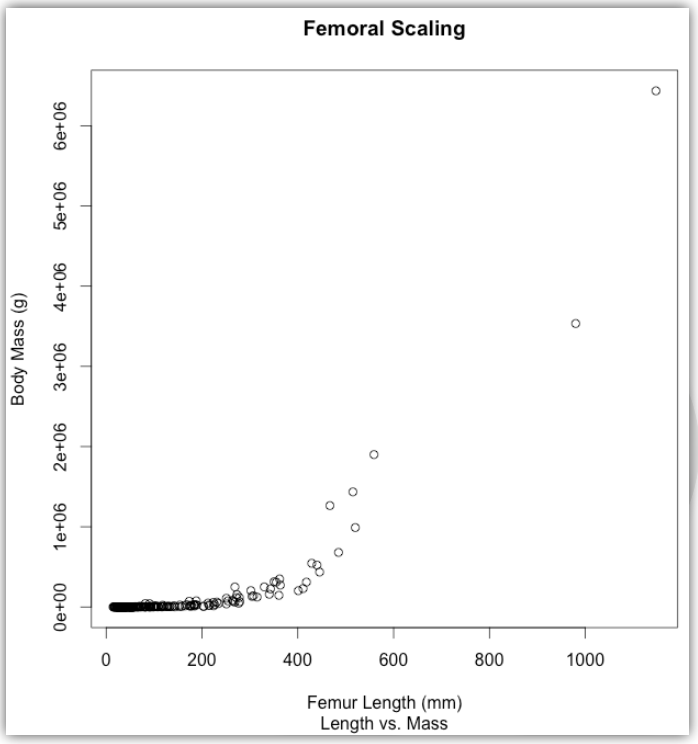
```
> x<-data$Femur.Length
> y<-data$Body.Mass..g.
> plot(x,y,type="p",
       main="Femoral Scaling",
       sub="Length vs. Mass",
       xlab="Femur Length (mm)",
       ylab="Body Mass (g)")
```

Plotting Data

Default



Names Specified



Other Optional Plotting Arguments

Argument	Description
`xlim, ylim`	scale to be used on x and y axis
`log`	axes to be log transformed
`axes`	whether axes should be drawn
`frame.plot`	whether box around plot is added
`cex` *	value by which objects in plot should be scaled
`col` *	colour specification e.g., `col="blue"`
`family`	specifies the typeface to be used (e.g., serif)
`font` *	specifies the font type (e.g., italics)
`lab`	indicates numbers of tickmarks on the axes
`lty`	line type, if lines are part of the plot
`lwd`	value by which line-width should be scaled
`pch`	specifies the point-type

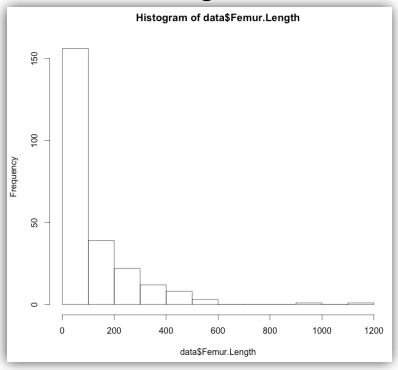
* `.axis`
`.main`
`.lab`
`.sub`

Other Types of Plots

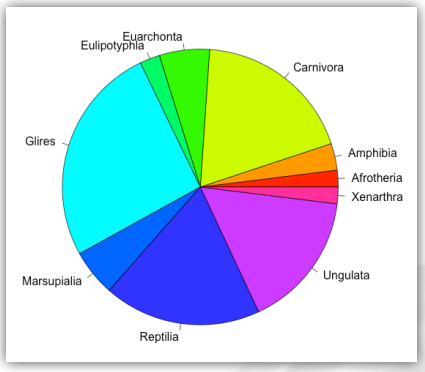
```
#Histogram
> hist(data$Femur.Length)
#Pie Chart
#sample size per clade
> Ns<-tapply(data$Higher.Clade,data$Higher.Clade,length)
> pie(Ns,col=rainbow(10))
#Bar Chart
> barplot(Ns,col=rainbow(10),ylab="Sample Size”)
#Box Plot
> boxplot(data$Femur.Length,ylab="Femur Length") #all species
#multiple box plots
> plot(data$Higher.Clade,data$Femur.Length,col=rainbow(10))
```

Other Types of Plots

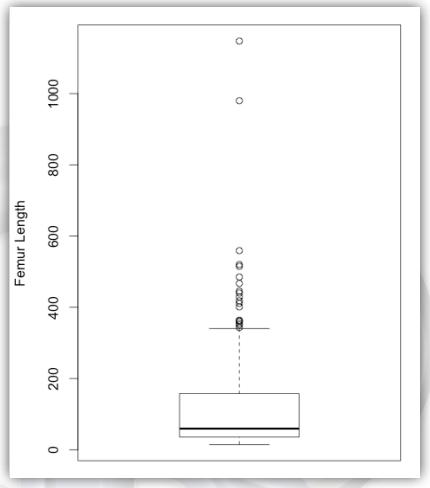
Histogram



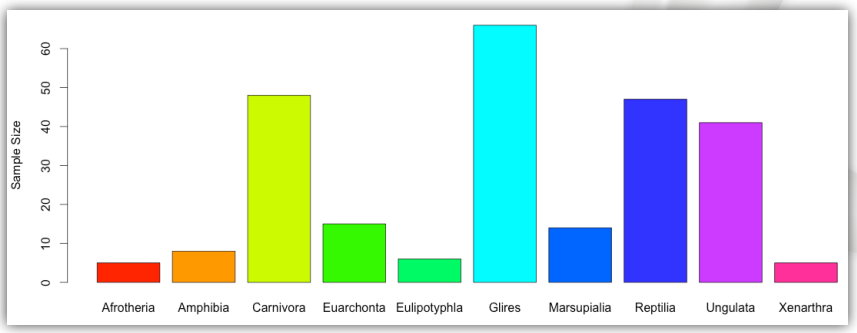
Pie Chart



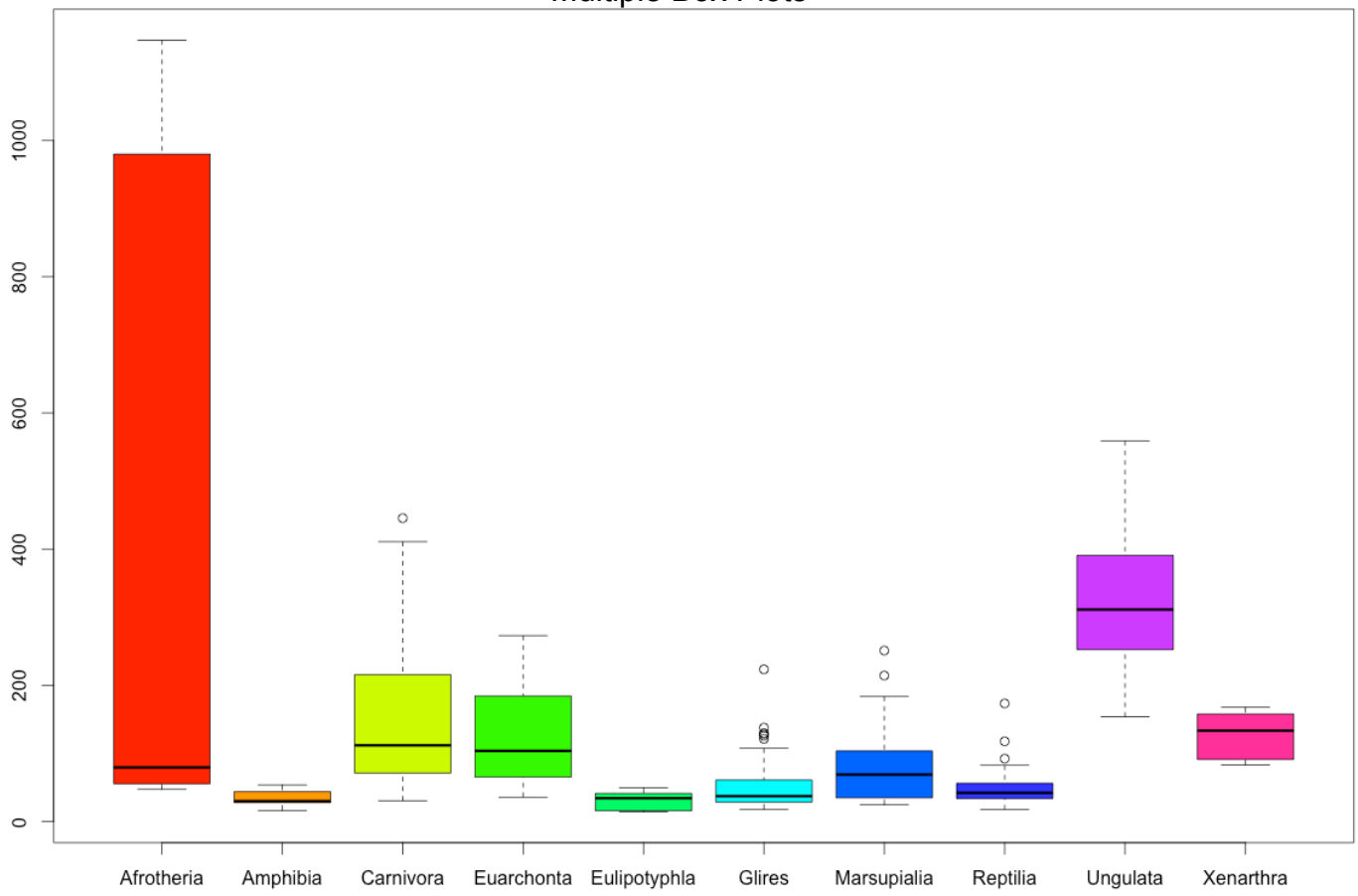
Box Plot



Bar Chart



Multiple Box Plots



Add to Existing Plot

Function	Description
`title()`	add title and other labels (e.g., main, xlab, ylab)
`abline()`	add a line (e.g., horizontal, vertical, regression)
`points()`	add datapoints (specify groups)
`lines()`	join datapoints with a line
`text()`	add text
`polygon()`	add shape based on x and y coordinates
`legend()`	add a legend
`arrows()`	add an arrow between two datapoints
`symbols()`	replace datapoints with symbols (circles, box plots)
`axis()`	add axes
`par(mfrow)`	**run before plotting**, specifies number of plots

```
#interacting with plots
> locator() #returns coordinates
> identify(x,y,labels) #identify of nearest point
```

```
#more options and help
> ?plot
> ?plot.default
> ?par
```

```
#special plotting libraries
> install.packages("plotrix")
> install.packages("ggplot2")
> install.packages("scatterplot3d")
> install.packages("maps")
```

Exercise 6

```
#more options and help
> ?plot
> ?plot.default
> ?par
```

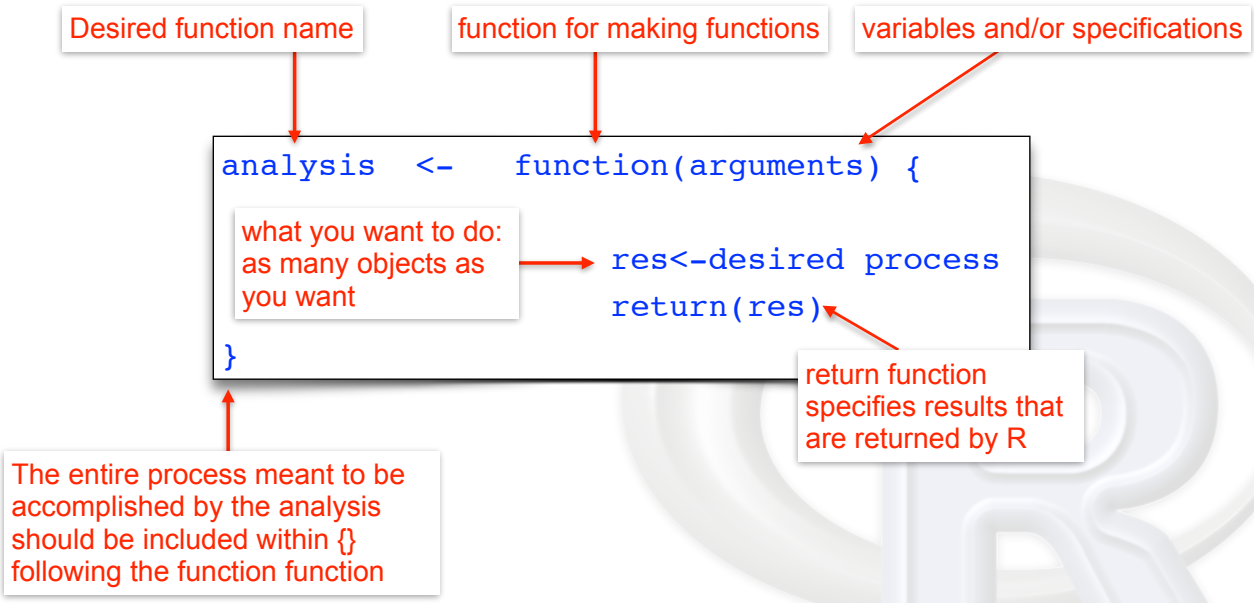
1. Experiment with the plot function.

2. Choose two variables from your personal dataset and plot them. If you have categorical columns try to call them out using the tapply function.

3. Try using different colours. Use the function `rainbow`, or try some of the other functions like it: `heat.colors`, `terrain.colors`, `topo.colors`

4. Try making your own colour vector with the colour chart, available at: http://nicolascampione.weebly.com/r-worshop.html

Making your own Functions



Making your own Functions

```
> stats<-function(X) {
    mean=mean(X,na.rm=TRUE) #na.rm removes the missing values
    median=median(X,na.rm=TRUE)
    max=max(X,na.rm=TRUE)
    min=min(X,na.rm=TRUE)
    sd=sd(X,na.rm=TRUE)
    res<-c(mean,median,max,min,sd) #summarize the results in one vector
    names(res)<-c("mean","median","max","min","sd") #add names to result vector
    return(res)
  }
> stats(data$Body.Mass..g.)
     mean    median       max       min        sd
 103597.5    1294.0 6435000.0      51.0  500139.9
> apply(data[,6:10],2,stats)
       Body.Mass..g. Humerus.Length Humerus.Circumference Femur.Length Femur.Circumference
mean         103597.5       104.8106              39.15004     121.6080            39.51167
median         1294.0        52.2750              16.25000      59.2575            18.35000
max         6435000.0      1035.0000             416.30000    1147.5000           399.00000
min              51.0        13.9000               4.36000      14.3500             4.00000
sd           500139.9       125.8172              53.28310     144.1260            50.16371
```

Exercise 7

1. Make a function that takes two numeric vectors of equal length (N).

2. Write the function so that it binds the two vectors and then calculates the sum and product of the vectors.

3. Have the function return the resultant matrix (Nx4).
   - If you are feeling adventurous, give the matrix column names (e.g., Var1, Var2, Sums, Products)

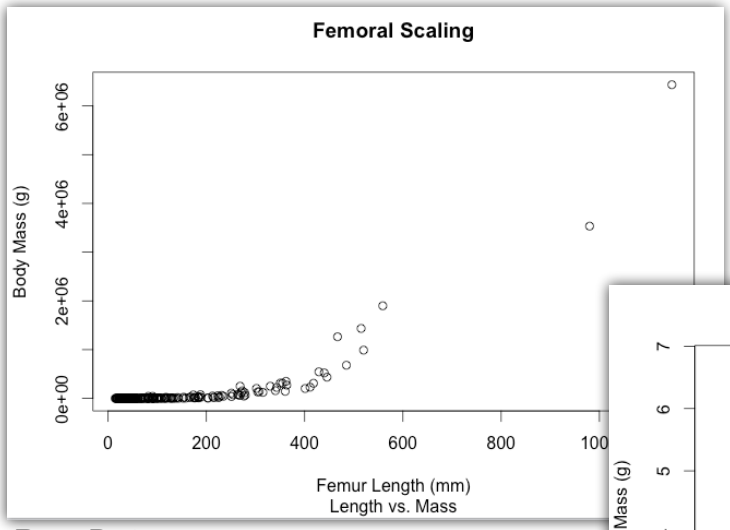
4. If you have time, make the function more challenging by making it return a list, with the above matrix and other information (e.g., colSums, colMeans, dim, etc.)

5. Apply the function to two data vectors from your dataset or from mine.

Loops

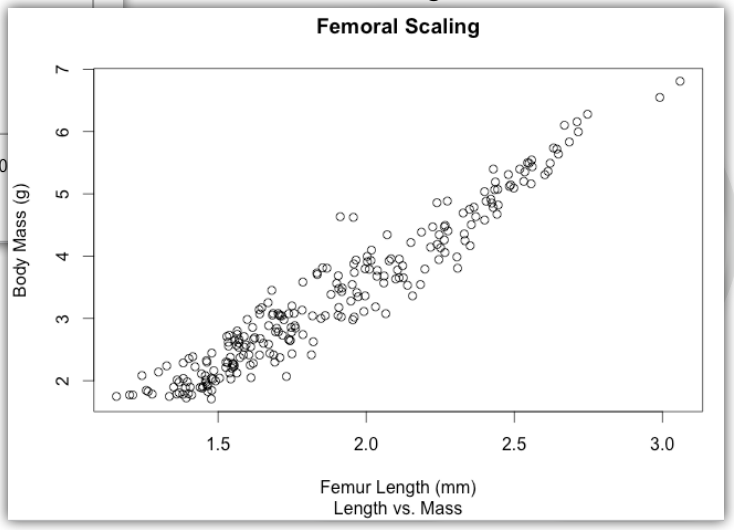
- running repetitive code
- useful for plotting data subsets on a plot
- can be used to run randomization tests
- in some instances can be replaced by apply-type functions

Loops



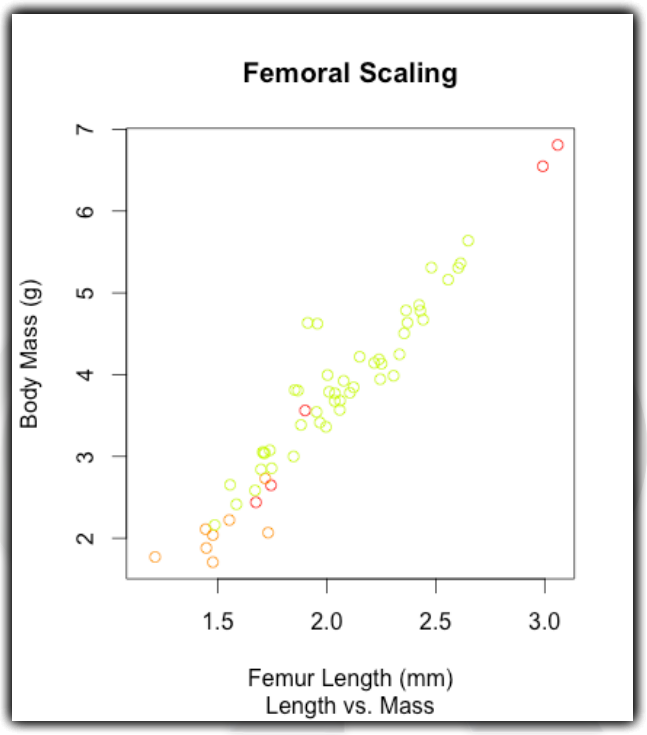
Raw Data

Log-Transformed Data



Loops

```
#set up plot, but make points invisible
> plot(log10(x),log10(y),type="n",
        main="Femoral Scaling",
       sub="Length vs. Mass",
       xlab="Femur Length (mm)",
       ylab="Body Mass (g)")
> points(log10(x[which(data$Higher.Clade=="Afrotheria")])
       log10(y[which(data$Higher.Clade=="Afrotheria")]),
       col=rainbow(10)[1])
> points(log10(x[which(data$Higher.Clade=="Amphibia")])
       log10(y[which(data$Higher.Clade=="Amphibia")]),
       col=rainbow(10)[2])
> points(log10(x[which(data$Higher.Clade=="Carnivora")])
       log10(y[which(data$Higher.Clade=="Carnivora")]),
       col=rainbow(10)[3])
#ETC...
```



Making a Loop

function for making a loop

reflects an iteration

sequence upon which the loop will act

```
for(i in 1:reps) {
                x[i]<-x[i]^2
 }
```

All arguments meant to be looped should be included within {} following the 'for' function arguments

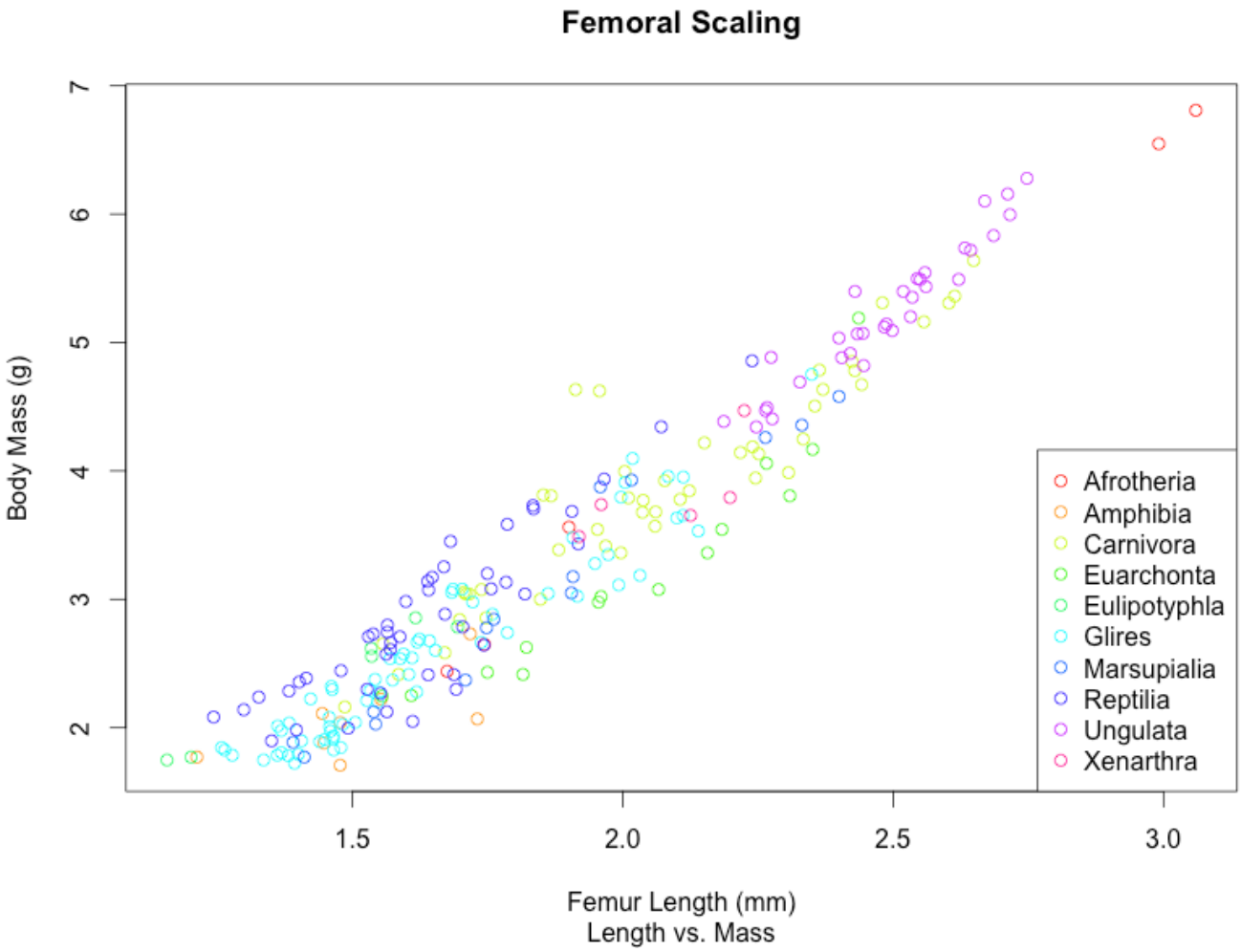
entire process desired to be looped

Loops

```
#simple loop
> m<-1:10 #square values
> m
[1]  1  2  3  4  5  6  7  8  9 10
> for(i in 1:length(m)) {
          m[i]<-m[i]^2}
> m
[1]   1   4   9  16  25  36  49  64  81 100
> m<-1:10
> for(i in 5:length(m)) {
          m[i]<-m[i]^2}
[1]   1   2   3   4  25  36  49  64  81 100
```

Loops

```
#set up plot, but make points invisible
> plot(log10(x),log10(y),type="n",
       main="Femoral Scaling",
       sub="Length vs. Mass",
       xlab="Femur Length (mm)",
       ylab="Body Mass (g)")
> clades<-levels(data$Higher.Clade)
> clades
[1] "Afrotheria"   "Amphibia"     "Carnivora"    "Euarchonta"    “Eulipotyphla” "Glires"    "Marsupialia"  "Reptilia"
[9] "Ungulata"     “Xenarthra"
> for(i in 1:length(clades)) {
  points(log10(x[which(data$Higher.Clade==clades[i])]),
         log10(y[which(data$Higher.Clade==clades[i])]),
         col=rainbow(10)[i])
}
> legend("bottomright",legend=clades,col=rainbow(10),pch=1)
```



UPPSALA UNIVERSITET

Exercise 8

1. Generate a loop that will plot each group onto their own plot, instead of all together.

2. Integrate the function `cor(x,y)` into the loop and correlate femur length to body mass for each group. Which group has the highest correlation?

3. If you want to make step 2 more challenging, try making an empty 10x1 matrix called cor.res. Assign names to the rows based on the name of the groups, and then assign the result from each cor analysis to the empty matrix. Hint: the matrix and naming of the rows is done before running the for loop.

Saving Data in R

1. Workspace:

- saves all **objects** (including functions) you have created and **datasets** your have uploaded
- the workspace can be reopened at a later date and you can continue from where you left off.
- if on native platform, this is done through the menus.
- Recommendation: Save to your working directory
- R will ask you if you want to save your workspace when you quit.
- NB. If you are working within a project in R studio, then when you reopen a project, it automatically opens the workspace.

Saving Data in R

2. History:

- saves all the commands you entered during the R session.
- it does not save the objects.
- also done through the menus.
- must be sought, R won't ask to save history.
- automatically done with R-studio Projects

Saving Data in R

3. Results:

- Results output to a table can be saved as .txt or .csv files using specific functions

```
> write.table(object,file="",...)
> write.csv()
> write.csv2()
```

- these will be saved to the working directory, unless you specify a particular location. See `?write.table` for more information.
- specific objects can be saved using the `save` function. Can be reloaded using the `load` function.
- in order to save multiple results or a list with various objects of different lengths use `sink()`.

```
> sink(file="results.txt");object;sink()
```

Saving Data in R

4. Plots:

- Native Platforms:
  - Make sure you are on the quartz window
  - Mac: File>>Save (or Save As) *PDF only
  - Windows: File>>Save As>>Pick a Format
- R-Studio
  - Multiple Formats (.eps, .svg, .pdf, .tiff, etc.)
  - can resize before exporting

Saving Data in R

4. Scripts:

- place to save commands, objects, and functions
- separate file from history and workspace
- can be sourced into other scripts and functions
- automates many processes

There is no need to save the console

Basic Analytical Functions

Function	Description	Function	Description
`mean(x)`	returns average	`summary(reg)`	returns detailed results of 'reg'
`median(x)`	returns median	`fitted(reg)`	returns fitted values along the line 'reg'
`sd(x)`	returns standard deviation	`predict(reg,newdata)`	predict new data from 'reg'
`var(x)`	returns variance	`residuals(reg)`	returns residuals of 'reg'
`range(x)`	returns range	`aov(y~x+A)`	run an analysis of variance (A=groups)
`log(x);` `log10(x)`	log-transform data. Natural or base-10	`prcomp(data)`	runs a Principal Component Analysis of 'data'
`shapiro.test(x)`	test for normality	`lda(data,grouping)`	runs a Discriminant Function Analysis (in 'MASS')
`var.test(x,y)`	test for difference in variance	`cca(data)`	runs a Correspondence Analysis (in 'vegan')
`cor(x,y);` `cov(x,y)`	correlation or covariance of two variables	`diversity(data)`	calculate diversity index (in 'vegan')
`cor.test(x,y)`	test for correlation	`ace(x,tree)`	calculates ancestral states of 'x' on a specified 'tree' (in 'ape')
`t.test(x,y)`	test for significant difference	`fitContinuous(tree,x)`	fit models of evolution (in 'geiger')
`reg<-lm(y~x)`	run an ordinary least squares regression	`disparity(data)`	calculate disparity in a sample (in 'geiger')

Useful Packages

Package	Description	Package	Description
car	Companion to Applied Regression	OUwie	Evolutionary Rates (Ornstein-Uhlenbeck Process)
doMC	Adaptor for Multicore Usege	paleotree	Paleontology and Phylogeny
extremevalues	Outlier Detection	paleoTS	Analysis of Paleontological Time-Series
fossil	Palaeoecological and Palaeogeographical Tools	pcaMethods	Missing Data Estimators (and other stuff)
geiger	Analysis of Evolutionary Diversification	phangorn	Phylogenetic Analyses
geomorph	Geometric Morphometric Analysis	phylobase; phytools	Tools for Investigating Phylogenetic Structure
lmtest	Testing Linear Regression Models	picante	Integrating Phylogenetic and Ecological Data
LOST	Missing Morphometric Data	plotrix	Various Plotting Functions
maps	Draw Geographical Maps	pwr	Functions for Power Analyses
MASSTIMATE	Body Mass Estimation for Vertebrates	shapes	Shape Analysis (Geometric Morphometrics)
nlme	Linear and Nonlinear Models	smatr	Standardised Major Axis (AKA. Model II Regression)
nlstools	Nonlinear Regression Tools	vegan	Analyses in Community Ecology

Useful Links

- R: http://www.r-project.org/
- R Studio: http://www.rstudio.com/
- Quick-R: http://www.statmethods.net/
- Gene Hunt's Lecture Notes: http://paleobiology.si.edu/staff/individuals/hunt.cfm
- David Polly's (morphometrics and other stats): http://mypage.iu.edu/~pdpolly/Software.html
- My Website: http://nicolascampione.weebly.com/quantitative-methods.html
- R iPhone App (payed): http://www.rinstructor.com/