Useful functions in R

Maria Novosolov

8 December, 2014

In this tutorial we will go over many useful functions in R that will either help you to manipulate your data or summarize it.

For starters it is recommended to clean your R. Every variable you create is saved to R memory and many times R doesn't forget just because you closed it and reopened it later. Thus you should always clean your environment when you start working on something new or with updated data. This can be done with the function

```
rm(list=ls()) # rm = remove; ls = list
```

It also possible to just see what is stored in R's memory and then remove only a specific object. To see what is stored we use just ls()

```
ls()
```

To remove an object we use: ('object' represents the object we want to remove)

```
rm(object)
```

Creating data from typing

We have 5 different types of data

1. Vector
2. Matrix
3. Array
4. Data frame
5. List

Even though usually we use excel to create our data it is very useful to know how to create a different data types from R. When we'll be learning about loops you'll see where it's implemented 1. Creating a vector

```
# a numerical vector
a<-c(1,2,3,4,5,6,7)
a
```

```
## [1] 1 2 3 4 5 6 7
```

```
# or character vector
b<-c("one","two","three")
b
```

```
## [1] "one"   "two"   "three"
```

```r
# or logical vector
c<-c(TRUE,FALSE,FALSE,TRUE)
c
```

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

2. Creating a matrix

```r
# Matrix with data in it
your.matrix<-matrix(1:20,nrow=5,ncol=4)
your.matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    6   11   16
## [2,]    2    7   12   17
## [3,]    3    8   13   18
## [4,]    4    9   14   19
## [5,]    5   10   15   20
```

```r
# empty matrix
empty.matrix<-matrix(NA,nrow=5,ncol=4)
```

3. Array

```r
# this is an empty array
my.array<-array(NA,dim=c(3,2,4),dimnames=list(c("a","b","c"),c("c","d"),c("f","g","h","i")))
my.array
```

```
## , , f
##
##    c  d
## a NA NA
## b NA NA
## c NA NA
##
## , , g
##
##    c  d
## a NA NA
## b NA NA
## c NA NA
##
## , , h
##
##    c  d
## a NA NA
## b NA NA
## c NA NA
##
## , , i
##
```

```
##    c  d
## a NA NA
## b NA NA
## c NA NA
```

4. Creating a data frame

```
patientID <- c(1, 2, 3, 4)
age <- c(25, 34, 28, 52)
diabetes <- c("Type1", "Type2", "Type1", "Type1")
status <- c("Poor", "Improved", "Excellent", "Poor")
patientdata <- data.frame(patientID, age, diabetes, status)
patientdata
```

```
##   patientID age diabetes    status
## 1         1  25    Type1      Poor
## 2         2  34    Type2  Improved
## 3         3  28    Type1 Excellent
## 4         4  52    Type1      Poor
```

```
# Or we can create an empty data frame
data_frame<-data.frame(patientID=factor(),age=numeric(),diabetes=factor(),status=factor())
```

5. list

```
my.list<-list(patientdata,my.array)
my.list
```

```
## [[1]]
##   patientID age diabetes    status
## 1         1  25    Type1      Poor
## 2         2  34    Type2  Improved
## 3         3  28    Type1 Excellent
## 4         4  52    Type1      Poor
##
## [[2]]
## , , f
##
##    c  d
## a NA NA
## b NA NA
## c NA NA
##
## , , g
##
##    c  d
## a NA NA
## b NA NA
## c NA NA
##
## , , h
```

```
##
##    c  d
## a NA NA
## b NA NA
## c NA NA
##
## , , i
##
##    c  d
## a NA NA
## b NA NA
## c NA NA
```

In this tutorial we will use the data from the course website () called "island_type_final2.csv". To upload a file we'll start setting our working directory using the function **setwd** (=set working directory). Remember: the path always have to be in double quotation marks ("") and with either forward slash (/) or two backward slashes(\\)

`setwd("~/Dropbox/Rworkshop/2014/Databases")`

If you are not sure in which directory you are you can look on the headline of the console, there you'll find written in grey the directory you are working in now. An easier way is to ask R to tell you in which directory you are

```
getwd()
```

```
## [1] "/Users/marianovosolov/Dropbox/Rworkshop/2015/Scripts"
```

After we set the working directory we want to upload the data into R. This we will do using the function **read.csv** or **read.txt** depends on how you saved your data. We will save it into a variable so that we can use it later. In our case we will call the variable **data**.

The general code for uploading data

`data<-read.csv("file_name.csv",header = T, row.names=1)` **header = T** tells R that we have names for the columns. **row.names = 1** tells R that column number 1 represents the row names. This is relevant when we want to work on matrices and is possible only when we do not have repetition in your ID names. In our case we do not need it. Now lets upload the data

```
# set working directory
# setwd("C:\\Users\\User\\Dropbox\\Rworkshop\\2014\\Databases")
setwd("~/Dropbox/Rworkshop/2014/Databases")
# read the data into R
data<-read.csv("island_type_final2.csv",header=T)
```

Getting information about your data

To check the names of the columns we use

`names(your_data_name)`

```
names(data)
```

```
##  [1] "species"        "what"           "family"         "insular"
##  [5] "Archipelago"    "largest_island" "area"           "type"
##  [9] "age"            "iso"            "lat"            "mass"
## [13] "clutch"         "brood"          "hatchling"      "productivity"
```

To find out the length of your data or vector we use

`length(object)`

If we use this function for data.frame we will get the number of columns we have

```
length(data)
```

```
## [1] 16
```

We can also ask for the length of a specific column in the data using the $ operator

`length(object$variable)`

In our case lets see the length of the column species (it will give us the number of rows in our data)

```
length(data$species)
```

```
## [1] 319
```

We can look at the dimensions of our data. This is very useful when we use matrices especially multidimensional matrices.

`dim(our_data)`

in our case

```
dim(data)
```

```
## [1] 319  16
```

We can also look at the first or the last few rows of our data just to see that it is indeed our data this we will do with the function **head()** or **tail()** respectively

`head(our_data)`

in our specific case it will give us the number of rows and columns

```
head(data)
```

```
##                       species  what     family insular
## 1 Afroablepharus_annobonensis  else  Scincidae     yes
## 2     Ailuronyx_seychellensis gecko Gekkonidae     yes
## 3     Ailuronyx_tachyscopaeus gecko Gekkonidae     yes
## 4      Ailuronyx_trachygaster gecko Gekkonidae     yes
## 5       Algyroides_fitzingeri  else Lacertidae     yes
## 6     Amblyrhynchus_cristatus  else  Iguanidae     yes
##             Archipelago largest_island area        type   age  iso lat
## 1 Sao_Tome_and_Principe        Annobon 1.20     Oceanic  0.04 2.53   1
## 2    Seychelles_Islands           Mahe 2.17 Continental  1.81 3.02   5
```

```
## 3    Seychelles_Islands           Mahe 2.17 Continental  1.81 3.02   5
## 4    Seychelles_Islands        Praslin 1.41 Continental  1.81 3.04   4
## 5                  None       Sardinia 4.38 Land_bridge -2.00 2.30  41
## 6 Galapagos_Archipelago        Isabela 3.66     Oceanic -0.15 2.93  21
##    mass clutch brood hatchling productivity
## 1 -0.21     NA    NA     -1.15           NA
## 2  1.21   0.18    NA     -0.16           NA
## 3  0.83   0.18    NA     -0.16           NA
## 4  1.84     NA    NA      0.47           NA
## 5 -0.06   0.40  0.00     -0.71        -0.31
## 6  3.15   0.35 -0.12      1.77         2.00
```

```
tail(our_data)
```

```
tail(data)
```

```
##                 species what      family insular
## 314 Varanus_lirungensis else   Varanidae     yes
## 315    Varanus_mabitang else   Varanidae     yes
## 316     Varanus_macraei else   Varanidae     yes
## 317     Varanus_melinus else   Varanidae     yes
## 318    Varanus_nuchalis else   Varanidae     yes
## 319 Xantusia_riversiana else Xantusiidae     yes
##                    Archipelago largest_island area        type   age  iso
## 314             Talaud_Islands       Salibabu 1.94     Oceanic  1.30 2.95
## 315         Philippine_Islands          Panay 4.08     Oceanic  0.70 1.64
## 316                New_Guinea        Batanta 2.66 Land_bridge -1.77 1.57
## 317             Maluku_Islands        Taliabu 3.46     Oceanic  0.48 2.13
## 318         Philippine_Islands         Negros 4.12     Oceanic  0.70 1.67
## 319 California_Channel_islands   San_Clemente 2.18     Oceanic  1.15 1.53
##     lat mass clutch brood hatchling productivity
## 314   4 2.91   0.40    NA        NA           NA
## 315  12 3.50   0.95  0.00        NA           NA
## 316   1 2.65   0.60    NA      1.01           NA
## 317   2 3.04   0.69  0.40      1.33         2.42
## 318  11 3.34   1.02    NA        NA           NA
## 319  33 1.06   0.64 -0.12     -0.29         0.22
```

We can also look at the head or the tail of only several columns

```
head(data[,1:4])
```

```
##                       species  what     family insular
## 1 Afroablepharus_annobonensis  else  Scincidae     yes
## 2     Ailuronyx_seychellensis gecko Gekkonidae     yes
## 3     Ailuronyx_tachyscopaeus gecko Gekkonidae     yes
## 4      Ailuronyx_trachygaster gecko Gekkonidae     yes
## 5       Algyroides_fitzingeri  else Lacertidae     yes
## 6     Amblyrhynchus_cristatus  else  Iguanidae     yes
```

```
tail(data[,5:6])
```

```
##                    Archipelago largest_island
## 314             Talaud_Islands       Salibabu
## 315         Philippine_Islands          Panay
## 316                New_Guinea        Batanta
## 317             Maluku_Islands        Taliabu
## 318         Philippine_Islands         Negros
## 319 California_Channel_islands   San_Clemente
```

The most useful function is **str()**. It gives you the structure of your data. Tells you which type of data it is, how each variable is stored and how it looks

```
str(data)
```

```
## 'data.frame':    319 obs. of  16 variables:
##  $ species       : Factor w/ 319 levels "Afroablepharus_annobonensis",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ what          : Factor w/ 3 levels "anoles","else",..: 2 3 3 3 2 2 2 2 2 2 ...
##  $ family        : Factor w/ 19 levels "Agamidae","Anguidae",..: 14 7 7 7 10 9 16 16 16 16 ...
##  $ insular       : Factor w/ 1 level "yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Archipelago   : Factor w/ 54 levels "Andaman_and_Nicobar_Islands",..: 44 46 46 46 36 16 20 17 54 1
##  $ largest_island: Factor w/ 141 levels "Amami_Oshima",..: 6 64 64 99 114 54 81 100 29 104 ...
##  $ area          : num  1.2 2.17 2.17 1.41 4.38 3.66 1.97 3.96 2.9 1.96 ...
##  $ type          : Factor w/ 3 levels "Continental",..: 3 1 1 1 2 3 3 1 3 3 ...
##  $ age           : num  0.04 1.81 1.81 1.81 -2 -0.15 0.3 1.08 0.3 0.7 ...
##  $ iso           : num  2.53 3.02 3.02 3.04 2.3 2.93 2.79 2.06 2.7 2.74 ...
##  $ lat           : int  1 5 5 4 41 21 17 18 15 18 ...
##  $ mass          : num  -0.21 1.21 0.83 1.84 -0.06 3.15 1.08 1.39 1.76 1.27 ...
##  $ clutch        : num  NA 0.18 0.18 NA 0.4 0.35 0.6 0.43 0.6 0.41 ...
##  $ brood         : num  NA NA NA NA 0 -0.12 NA 0.3 0.4 0.3 ...
##  $ hatchling     : num  -1.15 -0.16 -0.16 0.47 -0.71 1.77 NA 0.09 0.56 NA ...
##  $ productivity  : num  NA NA NA NA -0.31 2 NA 0.82 1.56 NA ...
```

In cases where are categorical variable is structured of numbers we want to make sure that R treats it like this and not as numbers. **This is also relevant for many other cases but this is the case we will address here* To learn about this we can use **is.factor**. Which will give us a logical answer of whether something is a factor or not

```
is.factor(data$type)
```

```
## [1] TRUE
```

If the result is **FALSE** than we can change it by using **as.factor()**. If we store it back into the variable in the data then the variable will become a factor

```
is.factor(data$age)
```

```
## [1] FALSE
```

```
data$age<-as.factor(data$age) #this will change the variable age into a factor and store it into the va
is.factor(data$age) # answers the question "is data$age is a factor"
```

```
## [1] TRUE
```

If we have a numeric variable that R for some reason decided to treat as factorial we can tell R to change it to numeric

first we will test whether it is numeric with **is.numeric()**

```
is.numeric(data$age)
```

```
## [1] FALSE
```

If it isn't numeric we will change it using **as.numeric()**. REMEMBER: in order to do any change in the dataset you have to store it back in the dataset

```
data$age<-as.numeric(data$age)
is.numeric(data$age)
```

```
## [1] TRUE
```

If we are interested to know what are the unique values of a certain variable we can use the function **unique()**. It will gives use the levels of a categorical variable or all the unique values of a numeric value

```
unique(data$type)
```

```
## [1] Oceanic     Continental Land_bridge
## Levels: Continental Land_bridge Oceanic
```

we can also just use the **levels()** function to see what are the levels of a categorical variable

```
levels(data$type)
```

```
## [1] "Continental" "Land_bridge" "Oceanic"
```

**There are a few different uses for each one. For example in loops we use unique() to run a loop on each group of lines that belong to the same category. If we want to store the level names we will use the level.*

the function which gives you the possibility to find out which rows in your data contain a certain logical value. Very useful for factorial variables For example we want to know which rows have Oceanic in their type variable

```
which(data$type=='Oceanic')
```

```
##   [1]   1   6   7   9  10  11  13  14  15  16  17  18  23  24  25  28  29
##  [18]  30  31  32  33  34  35  36  38  39  40  41  42  46  47  48  58  59
##  [35]  60  62  63  76  77  78  80  81  82  83  86  89  90  91  92  93  94
##  [52]  97  98  99 101 102 103 104 106 107 108 110 112 115 122 124 125 129
##  [69] 130 131 132 133 134 136 137 138 139 140 141 142 143 144 145 146 147
##  [86] 148 152 153 154 155 156 157 158 159 160 162 163 164 166 168 170 171
## [103] 174 176 177 187 188 189 190 191 192 193 194 195 201 202 203 204 205
## [120] 209 210 211 220 221 225 226 227 229 230 231 232 233 234 235 236 237
## [137] 239 240 244 245 246 251 253 262 268 271 272 275 276 277 278 279 284
## [154] 286 288 289 290 291 292 293 295 298 299 300 301 302 307 308 310 311
## [171] 312 313 314 315 317 318 319
```

We can also store it and then reuse it for loops or changes of names in the variable

```
a<-which(data$type=='Oceanic')
```

We can also use it to detect in which row we have NA's. **is.na()** answers the question if there are NA's. which returns which rows in the clutch variable are NA's

```
which(is.na(data$clutch))
```

```
##  [1]   1   4  11  28  33  61  81  82  86  89  92 102 105 122 133 144 146
## [18] 161 164 174 186 192 205 215 299 307 312
```

or to find out how many NA's we have. Adding the length before the which sums the number of rows that have NA in the clutch variable

```
length(which(is.na(data$clutch)))
```

```
## [1] 27
```

We can reorder the dataset according to a certain variable. For example we want to reorder the data according to the ascending order of the clutch size

general code

`newdata<-your.data.name[order(your.data.name$the.column),]`

for descending order

`newdata<-your.data.name[order(-your.data.name$the.column),]`

In our case:

```
newdata<-data[order(data$clutch),]
head(newdata$clutch)
```

```
## [1] 0 0 0 0 0 0
```

```
#or descending
newdata<-data[order(-data$clutch),]
head(newdata$clutch)
```

```
## [1] 1.62 1.34 1.26 1.23 1.23 1.13
```

Data subsetting

There are several ways to subset your data. The simplest way to subset the data is by rows or by columns

the number before the comma will be the row part and after the comma the column part

`your.data.name[,]`

Using this we can ask R to create a new dataset with only specific rows or columns or both

examples for syntax:

```
newdata<-data[3,] #gives you the 3 row
newdata<-data[,4] #gives you the 4th column
newdata<-data[,1:4] #gives you from the 1st to the 4th column
newdata<-data[,c(1,2,4)] #for specific column choice
# you can also use names of the columns instead of the numbers
newdata<-data[,c("species","type","iso")]
```

In a similar metter you can subset your data according to a specific factor in a variable

```
newdata<-your.data.name[your.data.name$column.name=='value',]
```

```
newdata<-data[data$type=='Oceanic',]
```

You can also subset the data based on continues values. For example you want all the raw that have a value smaller than 2 in a specific column

```
newdata<-your.data.name[your.data.name$column.name>2,]
```

```
newdata<-data[data$area>2,]
```

If we want to subset data based on more than one value in a variable we can use the %in% symbol that means "which contains" For example: if we want to subset our data to have only the rows that have the values 'Oceanic' and 'Land bridge' in the variable type

```
newdata<- your.data.name[your.data.name$column.name %in% c('value1','value2')]
```

```
newdata <- data[data$type %in% c('Oceanic','Land_bridge'), ]
levels(newdata$type)
```

```
## [1] "Continental" "Land_bridge" "Oceanic"
```

Another way to subset your data is to use the function subset. *NOTICE: This function needs to be logical*

```
newdata<-subset(your.data.name,column.name==value)
```

 # the value stands for the name of a categorical value from the column you are subsetting from

In our example: we want to subset the data so that we only have data for oceanic islands

```
newdata<-subset(data,type=='Oceanic')
levels(newdata$type)
```

```
## [1] "Continental" "Land_bridge" "Oceanic"
```

you can subset the data to have all the rows except the rows with species from oceanic islands

General code

```
newdata<-subset(your.data.name,column.name!=value)
```

```
newdata<-subset(data,type!='Oceanic')
levels(newdata$type)
```

```
## [1] "Continental" "Land_bridge" "Oceanic"
```

We can also subset the data based on several specific categorical variables

A general code

```
newdata<-subset(your.data.name,column.name %in% value)
```

 # or

```
newdata<-subset(your.data.name,column.name %in% c(value,value))
```

In our example

```
newdata<-subset(data,type%in%c('Oceanic','Land_bridge'))
levels(newdata$type)
```

```
## [1] "Continental" "Land_bridge" "Oceanic"
```

Summary statistics of your data

The first and most easy one is to just look at general summary of your data - the mean, median and quantiles of each variable in your data this is done using the **summary()** function

```
summary(data)
```

```
##                          species        what                 family
##  Afroablepharus_annobonensis:  1   anoles: 31   Scincidae      :99
##  Ailuronyx_seychellensis    :  1   else  :185   Gekkonidae     :49
##  Ailuronyx_tachyscopaeus    :  1   gecko :103   Dactyloidae    :31
##  Ailuronyx_trachygaster     :  1                Lacertidae     :29
##  Algyroides_fitzingeri      :  1                Diplodactylidae:25
##  Amblyrhynchus_cristatus    :  1                Iguanidae      :14
##  (Other)                    :313                (Other)        :72
##  insular               Archipelago        largest_island      area
##  yes:319   New_Caledonia     : 54   New_Caledonia: 54    Min.   :-1.050
##            Greater_Antilles  : 23   Puerto_Rico  : 12    1st Qu.: 2.460
##            None              : 22   Taiwan       : 11    Median : 3.330
##            Philippine_Islands: 18   Socotra      :  9    Mean   : 3.193
##            Canary_Islands    : 14   Viti_Levu    :  9    3rd Qu.: 4.120
##            Seychelles_Islands: 11   Jamaica      :  8    Max.   : 4.570
##            (Other)           :177   (Other)      :216
##           type          age              iso              lat
##  Continental: 97   Min.   : 1.00   Min.   :-0.250   Min.   : 0.0
##  Land_bridge: 45   1st Qu.:11.00   1st Qu.: 2.080   1st Qu.:11.0
##  Oceanic    :177   Median :25.00   Median : 2.540   Median :18.0
##                    Mean   :22.59   Mean   : 2.463   Mean   :17.4
##                    3rd Qu.:35.00   3rd Qu.: 3.020   3rd Qu.:22.0
##                    Max.   :41.00   Max.   : 3.430   Max.   :47.0
##
##       mass             clutch          brood           hatchling
##  Min.   :-0.6700   Min.   :0.000   Min.   :-0.4000   Min.   :-1.5900
##  1st Qu.: 0.3400   1st Qu.:0.180   1st Qu.: 0.0000   1st Qu.:-0.6475
##  Median : 0.6900   Median :0.300   Median : 0.2400   Median :-0.3600
##  Mean   : 0.8615   Mean   :0.361   Mean   : 0.2969   Mean   :-0.2029
##  3rd Qu.: 1.0900   3rd Qu.:0.480   3rd Qu.: 0.4000   3rd Qu.: 0.0500
##  Max.   : 4.5200   Max.   :1.620   Max.   : 1.4500   Max.   : 1.9100
```

```
##                   NA's   :27      NA's   :242       NA's   :149
##   productivity
##  Min.   :-0.5700
##  1st Qu.: 0.4200
##  Median : 0.6200
##  Mean   : 0.9851
##  3rd Qu.: 1.6800
##  Max.   : 3.4200
##  NA's   :262
```

In order to save this data we need to first save it into a variable

```
summ<-summary(data)
```

Then we can use the function **capture.output()** to save it as .txt

```
capture.output(summ,file="summary.txt")
```

If there is more than one data summarized we can use the **c()** to bind them together in the same .txt

`capture.output(c(summ,summ2),file="summary.txt")` If we want to know only the summary for a specific variable we can use a specific functions. This is done using the **$** sign. Here are some examples IMPORTAT: remember to put na.rm(=na remove) to true (=T/TRUE) in order for R to ignore the NA's in your data x

```
mean(data$mass,na.rm=T) # Gives you the mean of each of the variables in your data
```

```
## [1] 0.8615361
```

```
sd(data$mass,na.rm=T) #gives you the standard deviation of each of the variables in your data
```

```
## [1] 0.820978
```

```
median(data$mass,na.rm=T) #gives you the midpoint of each of the variables in your data
```

```
## [1] 0.69
```

There are a few ways to summarize more specific features of our data. For example, if we want to find the mean of a continuous trait in each category of a categorical trait. In our case we'll take the example of finding the mean island age in each type of island. For this we'll use the **aggregate()** function

```
aggregate(data$age,list(data$type),mean)
```

```
##       Group.1         x
## 1 Continental 35.793814
## 2 Land_bridge  3.555556
## 3     Oceanic 20.197740
```

The problem in this is the output datatype which in this case is list. The list is very useful in many cases but a bit less in others. If we are interested to use this data for something else we might need it to be a dataframe . To transform it to a dataframe we'll use the function **as.data.frame()**. And to make sure it is all ok we can use the **str()** function or just look at the output

```
a<-aggregate(data$age,list(data$type),mean)
a<-as.data.frame(a)
str(a)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ Group.1: Factor w/ 3 levels "Continental",..: 1 2 3
##  $ x      : num  35.79 3.56 20.2
```

```
a
```

```
##       Group.1         x
## 1 Continental 35.793814
## 2 Land_bridge  3.555556
## 3     Oceanic 20.197740
```

tapply() is very similar to aggregate. We use the same arguments for both of them. If we want to find the mean body mass on both islands and mainlands for each family and divide it to each type of island this will be the function

```
tapply(data$mass,list(data$type, data$insular, data$family),mean)
```

```
## , , Agamidae
##
##                   yes
## Continental 1.0950000
## Land_bridge 0.8383333
## Oceanic     1.0100000
##
## , , Anguidae
##
##              yes
## Continental 1.27
## Land_bridge   NA
## Oceanic     2.49
##
## , , Chamaeleonidae
##
##              yes
## Continental 1.50
## Land_bridge   NA
## Oceanic     0.77
##
## , , Dactyloidae
##
##                   yes
## Continental 0.4228571
## Land_bridge 0.2300000
## Oceanic     0.5291304
##
## , , Diplodactylidae
##
##              yes
```

```
## Continental 0.77
## Land_bridge 0.80
## Oceanic       NA
##
## , , Eublepharidae
##
##              yes
## Continental   NA
## Land_bridge 1.23
## Oceanic     1.03
##
## , , Gekkonidae
##
##                   yes
## Continental 0.9360000
## Land_bridge 0.6520000
## Oceanic     0.4951724
##
## , , Gymnophthalmidae
##
##              yes
## Continental   NA
## Land_bridge 0.25
## Oceanic       NA
##
## , , Iguanidae
##
##                  yes
## Continental       NA
## Land_bridge 3.265000
## Oceanic     2.923333
##
## , , Lacertidae
##
##              yes
## Continental 0.67
## Land_bridge 0.60
## Oceanic     1.12
##
## , , Leiocephalidae
##
##                  yes
## Continental       NA
## Land_bridge       NA
## Oceanic     1.053333
##
## , , Phrynosomatidae
##
##              yes
## Continental   NA
## Land_bridge   NA
## Oceanic     0.88
##
## , , Phyllodactylidae
```

```
##
##               yes
## Continental 0.870
## Land_bridge    NA
## Oceanic     0.718
##
## , , Scincidae
##
##                   yes
## Continental 0.5732432
## Land_bridge 0.8333333
## Oceanic     0.6580357
##
## , , Sphaerodactylidae
##
##                     yes
## Continental  0.03857143
## Land_bridge  0.29000000
## Oceanic     -0.08666667
##
## , , Teiidae
##
##                  yes
## Continental 0.90500
## Land_bridge      NA
## Oceanic     1.23125
##
## , , Tropiduridae
##
##                   yes
## Continental       NA
## Land_bridge       NA
## Oceanic     1.303333
##
## , , Varanidae
##
##                  yes
## Continental 2.62000
## Land_bridge 2.65000
## Oceanic     3.24125
##
## , , Xantusiidae
##
##              yes
## Continental   NA
## Land_bridge   NA
## Oceanic     1.06
```

plyr package

This is a very useful package for many things. Here we will give only a few examples of what it can do but I advise you to read more about it in http://plyr.had.co.nz/ and http://www.r-bloggers.com/a-fast-intro-to-plyr-for-r/. Some more specific information on how to work with this package http://

seananderson.ca/2013/12/01/plyr.html

The plyr package is based in the **split-apply-combine** method. It takes the data you give it, splits it according to one or a few specific variables in your data, applies a function or a few functions and combines it together. It can combine it into a new dataset (using summarize) or modify the existing one (using transform or mutate). You can also plot using ddply. This is useful when you want to plot several plots from your data. It allows you to change between types of datasets for example from dataframe to list and the opposite.

**Here I will talk only about the ddply but you can inquire about the other options of the package*

Lets start with the basics. The plyr package has a specific syntax that is fixed in all its functions. These are the things you need to follow:

1. specify your data
2. specify according to what it should split the data. It is important to remember to put the variable or variables in the following way **.(variable1,variable2)**
3. Specify what it should do after the split (summarize, transform, mutate)
4. Give it the name of the new columns and the function to run on the split data

Here is a general view on the code

```
a<-ddply(data,.(variable),summarize,column_name = function_name (variable_for_the_function))
```

In our case, for example we want to create a new table with the mean values of body mass according to family

```
library(plyr)
a<-ddply(data,.(family),summarize,mean_mass=mean(mass))
a
```

```
##               family   mean_mass
## 1           Agamidae 0.924000000
## 2           Anguidae 1.880000000
## 3     Chamaeleonidae 1.135000000
## 4        Dactyloidae 0.495483871
## 5    Diplodactylidae 0.772400000
## 6      Eublepharidae 1.096666667
## 7         Gekkonidae 0.617142857
## 8   Gymnophthalmidae 0.250000000
## 9          Iguanidae 2.972142857
## 10        Lacertidae 0.837931034
## 11    Leiocephalidae 1.053333333
## 12   Phrynosomatidae 0.880000000
## 13  Phyllodactylidae 0.775000000
## 14         Scincidae 0.636969697
## 15 Sphaerodactylidae 0.002857143
## 16           Teiidae 1.166000000
## 17      Tropiduridae 1.303333333
## 18         Varanidae 3.120000000
## 19       Xantusiidae 1.060000000
```

As you can see it gives us nine figures after the dot. To change that we can use the function **round()**

```
a<-ddply(data,.(family),summarize,mean_mass=round(mean(mass),3)) #this gives us only 3 figures after th
a
```

```
##               family mean_mass
## 1           Agamidae     0.924
## 2           Anguidae     1.880
## 3     Chamaeleonidae     1.135
## 4        Dactyloidae     0.495
## 5    Diplodactylidae     0.772
## 6      Eublepharidae     1.097
## 7         Gekkonidae     0.617
## 8   Gymnophthalmidae     0.250
## 9          Iguanidae     2.972
## 10        Lacertidae     0.838
## 11    Leiocephalidae     1.053
## 12   Phrynosomatidae     0.880
## 13  Phyllodactylidae     0.775
## 14         Scincidae     0.637
## 15 Sphaerodactylidae     0.003
## 16           Teiidae     1.166
## 17      Tropiduridae     1.303
## 18         Varanidae     3.120
## 19       Xantusiidae     1.060
```

We can also summarize according to two variables. Lets see the mean body mass for each family depending on the type of island it's found on

```
a<-ddply(data,.(type,family),summarize,mean_mass=round(mean(mass),3))
a
```

```
##           type            family mean_mass
## 1  Continental          Agamidae     1.095
## 2  Continental          Anguidae     1.270
## 3  Continental    Chamaeleonidae     1.500
## 4  Continental       Dactyloidae     0.423
## 5  Continental   Diplodactylidae     0.770
## 6  Continental        Gekkonidae     0.936
## 7  Continental        Lacertidae     0.670
## 8  Continental  Phyllodactylidae     0.870
## 9  Continental         Scincidae     0.573
## 10 Continental Sphaerodactylidae     0.039
## 11 Continental           Teiidae     0.905
## 12 Continental         Varanidae     2.620
## 13 Land_bridge          Agamidae     0.838
## 14 Land_bridge       Dactyloidae     0.230
## 15 Land_bridge   Diplodactylidae     0.800
## 16 Land_bridge     Eublepharidae     1.230
## 17 Land_bridge        Gekkonidae     0.652
## 18 Land_bridge  Gymnophthalmidae     0.250
## 19 Land_bridge         Iguanidae     3.265
## 20 Land_bridge        Lacertidae     0.600
## 21 Land_bridge         Scincidae     0.833
## 22 Land_bridge Sphaerodactylidae     0.290
## 23 Land_bridge         Varanidae     2.650
## 24     Oceanic          Agamidae     1.010
## 25     Oceanic          Anguidae     2.490
## 26     Oceanic    Chamaeleonidae     0.770
```

```
## 27     Oceanic       Dactyloidae     0.529
## 28     Oceanic     Eublepharidae     1.030
## 29     Oceanic        Gekkonidae     0.495
## 30     Oceanic         Iguanidae     2.923
## 31     Oceanic        Lacertidae     1.120
## 32     Oceanic    Leiocephalidae     1.053
## 33     Oceanic   Phrynosomatidae     0.880
## 34     Oceanic  Phyllodactylidae     0.718
## 35     Oceanic         Scincidae     0.658
## 36     Oceanic Sphaerodactylidae    -0.087
## 37     Oceanic           Teiidae     1.231
## 38     Oceanic      Tropiduridae     1.303
## 39     Oceanic         Varanidae     3.241
## 40     Oceanic       Xantusiidae     1.060
```

We can also have several functions in the same table. Lets say we want to add the SE of each body mass in each island type for each family. For this we'll need a new package named **sciplot**

```
library(sciplot)
a<-ddply(data,.(type,family),summarize,mean_mass=round(mean(mass),3),se_mass = round(se(mass),3))
a
```

```
##           type            family mean_mass se_mass
## 1  Continental          Agamidae     1.095   0.005
## 2  Continental          Anguidae     1.270      NA
## 3  Continental    Chamaeleonidae     1.500   0.290
## 4  Continental       Dactyloidae     0.423   0.196
## 5  Continental   Diplodactylidae     0.770   0.135
## 6  Continental        Gekkonidae     0.936   0.146
## 7  Continental        Lacertidae     0.670   0.020
## 8  Continental  Phyllodactylidae     0.870   0.430
## 9  Continental         Scincidae     0.573   0.082
## 10 Continental Sphaerodactylidae     0.039   0.137
## 11 Continental           Teiidae     0.905   0.485
## 12 Continental         Varanidae     2.620      NA
## 13 Land_bridge          Agamidae     0.838   0.064
## 14 Land_bridge       Dactyloidae     0.230      NA
## 15 Land_bridge   Diplodactylidae     0.800   0.110
## 16 Land_bridge     Eublepharidae     1.230      NA
## 17 Land_bridge        Gekkonidae     0.652   0.116
## 18 Land_bridge  Gymnophthalmidae     0.250      NA
## 19 Land_bridge         Iguanidae     3.265   0.345
## 20 Land_bridge        Lacertidae     0.600   0.063
## 21 Land_bridge         Scincidae     0.833   0.253
## 22 Land_bridge Sphaerodactylidae     0.290      NA
## 23 Land_bridge         Varanidae     2.650      NA
## 24     Oceanic          Agamidae     1.010   0.060
## 25     Oceanic          Anguidae     2.490      NA
## 26     Oceanic    Chamaeleonidae     0.770   0.030
## 27     Oceanic       Dactyloidae     0.529   0.065
## 28     Oceanic     Eublepharidae     1.030   0.090
## 29     Oceanic        Gekkonidae     0.495   0.064
## 30     Oceanic         Iguanidae     2.923   0.123
## 31     Oceanic        Lacertidae     1.120   0.158
```

```
## 32     Oceanic    Leiocephalidae     1.053   0.098
## 33     Oceanic   Phrynosomatidae     0.880   0.078
## 34     Oceanic  Phyllodactylidae     0.718   0.051
## 35     Oceanic         Scincidae     0.658   0.086
## 36     Oceanic Sphaerodactylidae    -0.087   0.186
## 37     Oceanic           Teiidae     1.231   0.099
## 38     Oceanic      Tropiduridae     1.303   0.084
## 39     Oceanic         Varanidae     3.241   0.209
## 40     Oceanic       Xantusiidae     1.060      NA
```

Now lets see what transform does Lets say we want to add the mean mass for each family to my original data. For this we'll use transform

```
data.a<-ddply(data,.(family),transform,mean_mass=round(mean(mass),3))
head(data.a)
```

```
##               species what   family insular          Archipelago
## 1         Draco_biaro else Agamidae     yes  Sangihe_Archipelago
## 2 Draco_bourouniensis else Agamidae     yes       Maluku_Islands
## 3  Draco_palawanensis else Agamidae     yes   Philippine_Islands
## 4   Draco_reticulatus else Agamidae     yes   Philippine_Islands
## 5    Draco_timorensis else Agamidae     yes Lesser_Sunda_Islands
## 6   Japalura_brevipes else Agamidae     yes               Taiwan
##   largest_island area        type age  iso lat mass clutch brood hatchling
## 1          Biaro 1.42     Oceanic  14 1.66   2 0.95   0.30    NA        NA
## 2           Buru 3.93     Oceanic  17 2.67   3 1.07   0.40    NA      0.45
## 3        Palawan 4.09 Continental  35 2.46   3 1.10   0.48    NA        NA
## 4          Samar 4.11 Land_bridge   1 1.28  11 1.11   0.40    NA        NA
## 5          Timor 4.45 Continental  17 2.65   9 1.09   0.40    NA        NA
## 6         Taiwan 4.54 Land_bridge   2 2.11  24 0.76   0.72  0.18        NA
##   productivity mean_mass
## 1           NA     0.924
## 2           NA     0.924
## 3           NA     0.924
## 4           NA     0.924
## 5           NA     0.924
## 6           NA     0.924
```

```
str(data.a)
```

```
## 'data.frame':    319 obs. of  17 variables:
##  $ species       : Factor w/ 319 levels "Afroablepharus_annobonensis",..: 124 125 126 127 128 180 18
##  $ what          : Factor w/ 3 levels "anoles","else",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ family        : Factor w/ 19 levels "Agamidae","Anguidae",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ insular       : Factor w/ 1 level "yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Archipelago   : Factor w/ 54 levels "Andaman_and_Nicobar_Islands",..: 43 27 39 39 22 49 49 49 49 4
##  $ largest_island: Factor w/ 141 levels "Amami_Oshima",..: 14 17 91 106 132 126 126 126 126 126 ...
##  $ area          : num  1.42 3.93 4.09 4.11 4.45 4.54 4.54 4.54 4.54 4.54 ...
##  $ type          : Factor w/ 3 levels "Continental",..: 3 3 1 2 1 2 2 2 2 2 ...
##  $ age           : num  14 17 35 1 17 2 2 2 2 2 ...
##  $ iso           : num  1.66 2.67 2.46 1.28 2.65 2.11 2.11 2.11 2.11 2.11 ...
##  $ lat           : int  2 3 3 11 9 24 24 23 25 24 ...
##  $ mass          : num  0.95 1.07 1.1 1.11 1.09 0.76 0.86 0.82 0.63 0.85 ...
```

```
##  $ clutch        : num  0.3 0.4 0.48 0.4 0.4 0.72 0.74 0.8 0.53 0.6 ...
##  $ brood         : num  NA NA NA NA NA 0.18 NA NA NA 0.4 ...
##  $ hatchling     : num  NA 0.45 NA NA NA NA NA NA NA -0.71 ...
##  $ productivity  : num  NA NA NA NA NA NA NA NA NA 0.28 ...
##  $ mean_mass     : num  0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 ...
```

We can also add a few new columns using transform or mutate

```
data.a<-ddply(data,.(family),transform,mean_mass=round(mean(mass),3),se_mass = round(se(mass),3))
head(data.a)
```

```
##               species what   family insular          Archipelago
## 1         Draco_biaro else Agamidae     yes  Sangihe_Archipelago
## 2 Draco_bourouniensis else Agamidae     yes       Maluku_Islands
## 3  Draco_palawanensis else Agamidae     yes   Philippine_Islands
## 4   Draco_reticulatus else Agamidae     yes   Philippine_Islands
## 5    Draco_timorensis else Agamidae     yes Lesser_Sunda_Islands
## 6   Japalura_brevipes else Agamidae     yes               Taiwan
##   largest_island area        type age  iso lat mass clutch brood hatchling
## 1          Biaro 1.42     Oceanic  14 1.66   2 0.95   0.30    NA        NA
## 2           Buru 3.93     Oceanic  17 2.67   3 1.07   0.40    NA      0.45
## 3        Palawan 4.09 Continental  35 2.46   3 1.10   0.48    NA        NA
## 4          Samar 4.11 Land_bridge   1 1.28  11 1.11   0.40    NA        NA
## 5          Timor 4.45 Continental  17 2.65   9 1.09   0.40    NA        NA
## 6         Taiwan 4.54 Land_bridge   2 2.11  24 0.76   0.72  0.18        NA
##   productivity mean_mass se_mass
## 1           NA     0.924   0.053
## 2           NA     0.924   0.053
## 3           NA     0.924   0.053
## 4           NA     0.924   0.053
## 5           NA     0.924   0.053
## 6           NA     0.924   0.053
```

```
str(data.a)
```

```
## 'data.frame':    319 obs. of  18 variables:
##  $ species       : Factor w/ 319 levels "Afroablepharus_annobonensis",..: 124 125 126 127 128 180 18
##  $ what          : Factor w/ 3 levels "anoles","else",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ family        : Factor w/ 19 levels "Agamidae","Anguidae",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ insular       : Factor w/ 1 level "yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Archipelago   : Factor w/ 54 levels "Andaman_and_Nicobar_Islands",..: 43 27 39 39 22 49 49 49 49 4
##  $ largest_island: Factor w/ 141 levels "Amami_Oshima",..: 14 17 91 106 132 126 126 126 126 126 ...
##  $ area          : num  1.42 3.93 4.09 4.11 4.45 4.54 4.54 4.54 4.54 4.54 ...
##  $ type          : Factor w/ 3 levels "Continental",..: 3 3 1 2 1 2 2 2 2 2 ...
##  $ age           : num  14 17 35 1 17 2 2 2 2 2 ...
##  $ iso           : num  1.66 2.67 2.46 1.28 2.65 2.11 2.11 2.11 2.11 2.11 ...
##  $ lat           : int  2 3 3 11 9 24 24 23 25 24 ...
##  $ mass          : num  0.95 1.07 1.1 1.11 1.09 0.76 0.86 0.82 0.63 0.85 ...
##  $ clutch        : num  0.3 0.4 0.48 0.4 0.4 0.72 0.74 0.8 0.53 0.6 ...
##  $ brood         : num  NA NA NA NA NA 0.18 NA NA NA 0.4 ...
##  $ hatchling     : num  NA 0.45 NA NA NA NA NA NA NA -0.71 ...
##  $ productivity  : num  NA NA NA NA NA NA NA NA NA 0.28 ...
##  $ mean_mass     : num  0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 ...
##  $ se_mass       : num  0.053 0.053 0.053 0.053 0.053 0.053 0.053 0.053 0.053 0.053 ...
```

```
data.a<-ddply(data,.(family),mutate,mean_mass=round(mean(mass),3),se_mass = round(se(mass),3))
head(data.a)
```

```
##               species what   family insular          Archipelago
## 1         Draco_biaro else Agamidae     yes  Sangihe_Archipelago
## 2 Draco_bourouniensis else Agamidae     yes       Maluku_Islands
## 3  Draco_palawanensis else Agamidae     yes   Philippine_Islands
## 4   Draco_reticulatus else Agamidae     yes   Philippine_Islands
## 5    Draco_timorensis else Agamidae     yes Lesser_Sunda_Islands
## 6   Japalura_brevipes else Agamidae     yes               Taiwan
##   largest_island area        type age  iso lat mass clutch brood hatchling
## 1          Biaro 1.42     Oceanic  14 1.66   2 0.95   0.30    NA        NA
## 2           Buru 3.93     Oceanic  17 2.67   3 1.07   0.40    NA      0.45
## 3        Palawan 4.09 Continental  35 2.46   3 1.10   0.48    NA        NA
## 4          Samar 4.11 Land_bridge   1 1.28  11 1.11   0.40    NA        NA
## 5          Timor 4.45 Continental  17 2.65   9 1.09   0.40    NA        NA
## 6         Taiwan 4.54 Land_bridge   2 2.11  24 0.76   0.72  0.18        NA
##   productivity mean_mass se_mass
## 1           NA     0.924   0.053
## 2           NA     0.924   0.053
## 3           NA     0.924   0.053
## 4           NA     0.924   0.053
## 5           NA     0.924   0.053
## 6           NA     0.924   0.053
```

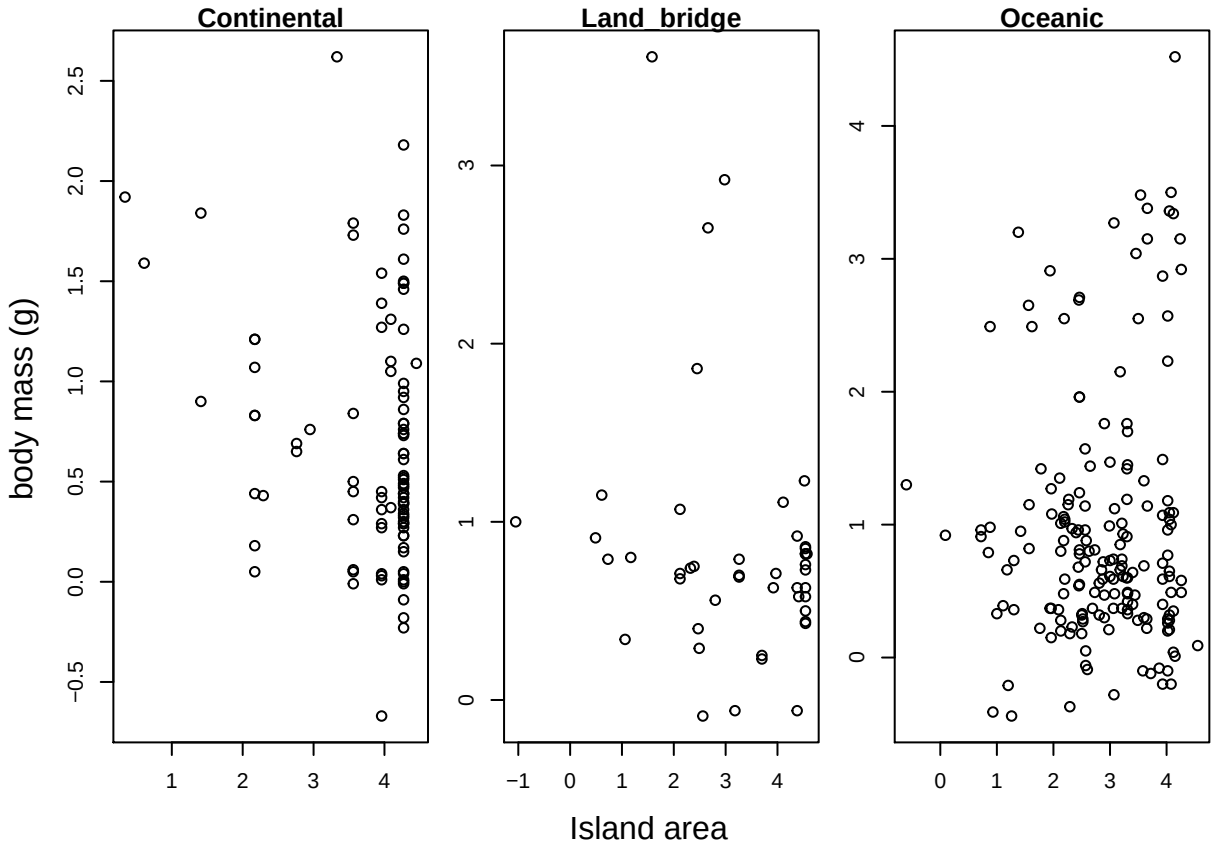
```
str(data.a)
```

```
## 'data.frame':    319 obs. of  18 variables:
##  $ species       : Factor w/ 319 levels "Afroablepharus_annobonensis",..: 124 125 126 127 128 180 181
##  $ what          : Factor w/ 3 levels "anoles","else",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ family        : Factor w/ 19 levels "Agamidae","Anguidae",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ insular       : Factor w/ 1 level "yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Archipelago   : Factor w/ 54 levels "Andaman_and_Nicobar_Islands",..: 43 27 39 39 22 49 49 49 49 4
##  $ largest_island: Factor w/ 141 levels "Amami_Oshima",..: 14 17 91 106 132 126 126 126 126 126 ...
##  $ area          : num  1.42 3.93 4.09 4.11 4.45 4.54 4.54 4.54 4.54 4.54 ...
##  $ type          : Factor w/ 3 levels "Continental",..: 3 3 1 2 1 2 2 2 2 2 ...
##  $ age           : num  14 17 35 1 17 2 2 2 2 2 ...
##  $ iso           : num  1.66 2.67 2.46 1.28 2.65 2.11 2.11 2.11 2.11 2.11 ...
##  $ lat           : int  2 3 3 11 9 24 24 23 25 24 ...
##  $ mass          : num  0.95 1.07 1.1 1.11 1.09 0.76 0.86 0.82 0.63 0.85 ...
##  $ clutch        : num  0.3 0.4 0.48 0.4 0.4 0.72 0.74 0.8 0.53 0.6 ...
##  $ brood         : num  NA NA NA NA NA 0.18 NA NA NA 0.4 ...
##  $ hatchling     : num  NA 0.45 NA NA NA NA NA NA NA -0.71 ...
##  $ productivity  : num  NA NA NA NA NA NA NA NA NA 0.28 ...
##  $ mean_mass     : num  0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 0.924 ...
##  $ se_mass       : num  0.053 0.053 0.053 0.053 0.053 0.053 0.053 0.053 0.053 0.053 ...
```

Plotting with plyr lets say we want to plot the body mass against the island area for each island type. For this we will use d_ply. This will create the data, put will not generate an output as we don't need it. The function will use it to generate the plots

First we will . This is done

```
# specify to R that we want the plots put next to each other
par(mfrow = c(1, 3), mar = c(2, 2, 1, 1), oma = c(3, 3, 0, 0))
# split the data, plot it and throw away the combined output
d_ply(data, .(type), transform, plot(area,mass, main = unique(type)))
# add names for the axes
mtext("Island area", side = 1, outer = TRUE, line = 1)
mtext("body mass (g)", side = 2, outer = TRUE, line = 1)
```



 # In this package there is no need to put quotation marks of any kind in the variable names

reshape and reshape2 packages

Both of these packages are very useful to modify the structure of your data. It works in two steps. First you **melt** you data - put the data in only a few columns based your categorical variables. Then you **cast** new structure to the data. There is not much difference between reshape and reshape2. The downside of these packages that they wont always work. . . .

** for more information on the package http://www.r-statistics.com/tag/data-frame/*

Lets see how it works. First thing - upload our packages

```
library(reshape)
library(reshape2)
```

We'll start with melting our data. When we melt the data it will create columns of al of our categorical variables and use them as id variables. The column names of the continuous variables will go into *variable* column and the values of the continuous variables will go into the *value* column

```
newdata<-melt(data)
```

```
## Using species, what, family, insular, Archipelago, largest_island, type as id variables
```

```
head(newdata)
```

```
##                       species  what     family insular
## 1 Afroablepharus_annobonensis  else  Scincidae     yes
## 2     Ailuronyx_seychellensis gecko Gekkonidae     yes
## 3     Ailuronyx_tachyscopaeus gecko Gekkonidae     yes
## 4      Ailuronyx_trachygaster gecko Gekkonidae     yes
## 5       Algyroides_fitzingeri  else Lacertidae     yes
## 6     Amblyrhynchus_cristatus  else  Iguanidae     yes
##             Archipelago largest_island        type variable value
## 1 Sao_Tome_and_Principe        Annobon     Oceanic     area  1.20
## 2    Seychelles_Islands           Mahe Continental     area  2.17
## 3    Seychelles_Islands           Mahe Continental     area  2.17
## 4    Seychelles_Islands         Praslin Continental     area  1.41
## 5                  None       Sardinia Land_bridge     area  4.38
## 6 Galapagos_Archipelago         Isabela     Oceanic     area  3.66
```

**We can specify which variables to use as id's but without specification it will just use everything.*

In order to return the original structure of the data we will cast using the three dots '**...**'. this in R means use all the variables in the data

The syntax: first we write the data that we are using to do the casting to. Then we write which variables we want as columns (the left of the tilde) and which we want to be the rows (the right side of the tilde). If we are interested to create summarization of the variables we can write the functions we want, if we want to subset the variable we can also add that a subsetting function. **note: if you have NA in your continuous variables to cast the data you need to add 'na.rm=T'*

```
newdata2<-cast(newdata,...~variable)
names(newdata2)
```

```
##  [1] "species"        "what"           "family"         "insular"
##  [5] "Archipelago"    "largest_island" "type"           "area"
##  [9] "age"            "iso"            "lat"            "mass"
## [13] "clutch"         "brood"          "hatchling"      "productivity"
```

```
head(newdata2)
```

```
##                       species  what     family insular
## 1 Afroablepharus_annobonensis  else  Scincidae     yes
## 2     Ailuronyx_seychellensis gecko Gekkonidae     yes
## 3     Ailuronyx_tachyscopaeus gecko Gekkonidae     yes
## 4      Ailuronyx_trachygaster gecko Gekkonidae     yes
## 5       Algyroides_fitzingeri  else Lacertidae     yes
## 6     Amblyrhynchus_cristatus  else  Iguanidae     yes
##             Archipelago largest_island        type area age  iso lat  mass
## 1 Sao_Tome_and_Principe        Annobon     Oceanic 1.20   8 2.53   1 -0.21
## 2    Seychelles_Islands           Mahe Continental 2.17  41 3.02   5  1.21
```

```
## 3    Seychelles_Islands           Mahe Continental 2.17  41 3.02   5  0.83
## 4    Seychelles_Islands        Praslin Continental 1.41  41 3.04   4  1.84
## 5                  None       Sardinia Land_bridge 4.38   1 2.30  41 -0.06
## 6 Galapagos_Archipelago        Isabela     Oceanic 3.66   6 2.93  21  3.15
##   clutch brood hatchling productivity
## 1     NA    NA     -1.15           NA
## 2   0.18    NA     -0.16           NA
## 3   0.18    NA     -0.16           NA
## 4     NA    NA      0.47           NA
## 5   0.40  0.00     -0.71        -0.31
## 6   0.35 -0.12      1.77         2.00
```

now lets say that you want to find the mean of each contentious variable (e.g. area, isolation etc.) for each type of island. Here we put the variable on the left to represent the rows and the type on the right to represent the columns. Then we ask to calculate the mean of each variable for each type and tell the function to ignore the NA values. We can change the variable and the type in their places to change the order of the dataset

```
newdata2<-cast(newdata,variable~type,mean,na.rm=T)
head(newdata2)
```

```
##   variable Continental Land_bridge    Oceanic
## 1     area   3.7749485   3.1780000  2.8779096
## 2      age  35.7938144   3.5555556 20.1977401
## 3      iso   2.7745361   1.5797778  2.5172316
## 4      lat  17.6907216  22.5777778 15.9322034
## 5     mass   0.6840206   0.8384444  0.9646893
## 6   clutch   0.3314894   0.4554762  0.3533333
```

if we are interested to see the mean only for a subset of the variable, for example for the area and the isolation

```
newdata2<-cast(newdata,type~variable,mean,subset=variable %in% c('area','iso'))
head(newdata2)
```

```
##          type     area      iso
## 1 Continental 3.774948 2.774536
## 2 Land_bridge 3.178000 1.579778
## 3     Oceanic 2.877910 2.517232
```

perhaps we are interested in finding not only the mean but also the sd of these two variables. We'll just bind them into **c()**

```
newdata2<-cast(newdata,type~variable,c(mean,sd,median),subset=variable %in% c('area','iso'))
head(newdata2)
```

```
##          type area_mean   area_sd area_median iso_mean    iso_sd
## 1 Continental  3.774948 0.8759877        4.27 2.774536 0.4515022
## 2 Land_bridge  3.178000 1.4175869        3.26 1.579778 0.6270838
## 3     Oceanic  2.877910 0.9689138        3.00 2.517232 0.4972638
##   iso_median
## 1       3.08
## 2       1.58
## 3       2.58
```

If we are interested to see the general summary for these variables without the island type we can leave one side of the tilde empty

```
newdata2<-cast(newdata,~variable,c(mean,sd,median),subset=variable %in% c('area','iso'))
head(newdata2)
```

```
##   value area_mean  area_sd area_median iso_mean    iso_sd iso_median
## 1 (all)  3.193009 1.089901        3.33 2.463229 0.6280412       2.54
```

if we want to get to the cast result without melting the dataset this is also possible by using recast Note: This function doesn't always work...

Lets create a table with the mean values of each variable on each type of island from our original data

```
newdata2<-recast(data,...~type,mean,na.rm=T,measure.var = c('area','iso'))
newdata2<-as.data.frame(newdata2)
head(newdata2)
```

```
##                       species  what     family insular
## 1 Afroablepharus_annobonensis  else  Scincidae     yes
## 2     Ailuronyx_seychellensis gecko Gekkonidae     yes
## 3     Ailuronyx_tachyscopaeus gecko Gekkonidae     yes
## 4      Ailuronyx_trachygaster gecko Gekkonidae     yes
## 5       Algyroides_fitzingeri  else Lacertidae     yes
## 6     Amblyrhynchus_cristatus  else  Iguanidae     yes
##             Archipelago largest_island age lat  mass clutch brood
## 1 Sao_Tome_and_Principe        Annobon   8   1 -0.21     NA    NA
## 2    Seychelles_Islands           Mahe  41   5  1.21   0.18    NA
## 3    Seychelles_Islands           Mahe  41   5  0.83   0.18    NA
## 4    Seychelles_Islands        Praslin  41   4  1.84     NA    NA
## 5                  None       Sardinia   1  41 -0.06   0.40  0.00
## 6 Galapagos_Archipelago        Isabela   6  21  3.15   0.35 -0.12
##   hatchling productivity variable Continental Land_bridge Oceanic
## 1     -1.15           NA     area         NaN         NaN    1.20
## 2     -0.16           NA     area        2.17         NaN     NaN
## 3     -0.16           NA     area        2.17         NaN     NaN
## 4      0.47           NA     area        1.41         NaN     NaN
## 5     -0.71        -0.31     area         NaN        4.38     NaN
## 6      1.77         2.00     area         NaN         NaN    3.66
```

It is also possible to subset to get only the variables we want. Here it is done using the measure.var addition in the function

```
newdata2<-recast(data,variable~type,mean,na.rm=T,measure.var=c('area','iso'))
newdata2<-as.data.frame(newdata2)
head(newdata2)
```

```
##   variable Continental Land_bridge  Oceanic
## 1     area    3.774948    3.178000 2.877910
## 2      iso    2.774536    1.579778 2.517232
```

System time

http://www.ats.ucla.edu/stat/r/faq/timing_code.htm

In R it is possible to document the time it takes to run an analysis using **system time()**. Lets see what it does using the plyr package. It is very simple to use it, you just put your code inside the parenthesis of system.time and it will give you how long it takes to run your code. the **user** time relates to the execution of the code, the **system** time relates to your CPU, and the **elapsed** time is the difference in times since you started the stopwatch (and will be equal to the sum of user and system times if the chunk of code was run altogether)

```
system.time(data.a<-ddply(data,.(family),transform,mean_mass=round(mean(mass),3),se_mass = round(se(mas
```

```
##    user  system elapsed
##   0.030   0.001   0.031
```

Saving output

There are many ways to save your output and it depends on what it is and what format you want it saved. We will go over a few methods

First and very useful is the **sink()** function. This tells R to send all your output to a file you specify rather than to the console. It is important to remember that in order to use this function you need to put it on the top and the bottom of all the functions for which you want the output to go for a specific file. Also in the top sink() function you need to specify the path of the file and it's name. It should be a file name that does not yet exists. R will create it for you and put all the output inside.

*sends all the output from this function on directly to a txt or a csv file in the working directory

```
sink("file_name.txt")
```

*to finish modifying the file

```
sink()
```

You can also save plots into **jpeg** or **pdf**

```
pdf("filename.pdf")
```

```
jpeg("filename.jpg")
```

In the end of these functions you have to tell R to stop outputting plots into the files with **dev.off()** function