TOO COMPLICTAED
Examples how to change list elements
Solutions to everything
Real life examples, not a mean of wweight and age

Another type of data structure: a list.

```
>l <- list("first",3,T,10:5)

>l
```

List elements may be of different mode. They may be even lists (“recursive” object)

```
 >l<-list(a = 1:3,

       b = c("tre","de","ge"),

       thirdElement = list(1:3, (1:3)>2, 1:15)

       )

 >l
```

```
> l
$a
[1] 1 2 3

$b
[1] "tre" "de"  "ge"

$thirdElement
$thirdElement[[1]]
[1] 1 2 3

$thirdElement[[2]]
[1] FALSE FALSE  TRUE

$thirdElement[[3]]
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

```
>Lst<-list(Name="Charles",
       Wife="Emma",
       no.children=5,
       children.age=c(2,5,7,8,12),
   favor.food=list(fruit="apple",drink="wine"))
```

Have a look on resulting object

Check what is the lenght of list Lst?

What are the names of list elements?

```
> Lst
$Name
[1] "Charles”
$Wife
[1] "Emma”
$no.children
[1] 5
$children.age
[1]  2  5  7  8 12
$favor.food
$favor.food$fruit
[1] "apple”
$favor.food$drink
[1] "wine”

> names(Lst)
[1] "Name"         "Wife"           "no.children"
"children.age" "favor.food"
> length(Lst)
[1] 5
```

To access a list element, one uses [[]].
So first Lst element is Lst[[1]] and so on.
One may use also element name:

```
>Lst[[2]]
>Lst[["Wife"]]
```

What is fifth element of the Lst?	`Lst[[5]]`
Second element of of fifth element?	`Lst[[5]][[2]]`
Get age of third kid.	`Lst[[4]][3]`
Get favorite drink.	`Lst[[“favor.food”]][[“drink”]]`

Change name into “Alfred”.	`Lst[[“name”]]<-”Alfred”`
Make children one year older.	`Lst[[“children.age”]]<- Lst[[“children.age”]]+1`
Change name of second argument into “sister”	`names(Lst)[2]<-”Sister”`

Another way of accesing an element : listname$elementname

```
>Lst$Wife

>Lst$favor.food$drink
```

What are:

```
>Lst[1]          #(compare with Lst[[1]])
>Lst["Name”]     #(compare with Lst[["Name”]]
>Lst[c(4,2,1)]
>Lst[c("Name","Wife")]
```

Applying a function to each list element:

```
lapply (list, function)

unlist(list)
```

Prepare a list with three elements:

- A: vector with integers from 1 to 10
- B: 25 numbers from uniform distribution between 1 and 10
- C: 7 numbers from normal distribution, with mean 10 and stand. dev. 2

What is the maximal element of A, B, C?
Average?
Variance?
Maximum/mean/variance for all elements of the list?

```
Mylist<-list(A=1:10, B=runif(25,1,10),C=rnorm(7,10,2))


lapply(Mylist, max)

lapply(Mylist,mean)

lapply(Mylist,var)


max(unlist(Mylist))

mean(unlist(Mylist))

var(unlist(Mylist))
```

Make a list with food items: vegetables (onion, cucumber), fruits (apples, pears...), dairy (cheese).
Add "bread" element, containing "rye bread" and "croissant".
Add milk and cream to dairy.
Get length of each type of food (fruits vegetables etc.)

```
food<-list(vegetables=c(“onion”,”cucumber”),
          fruits<-c(“apples”,”pears”,”oranges”),
          dairy<-”cheese”)
```

```
food$bread<-c(“rye bread”,”coissant”),
food$dairy<-c(food$dairy,“milk”,”cream”)
lapply(food,length)
```

Using own functions in apply(), lapply() etc.

Define a function with one argument myfunction()
and
lapply (list,myfunction)

Or define a function, without a name, inside lapply() call:

```
lapply(list,function(x){function_body})
```

Construct a list of dataframes - each one contains results of one experiment

```
birds<-lapply(1:3,function(x){

          data.frame (offspring = sample(1:(10+x),10,replace=T)
                        weight = x+round(runif(10,12,20),1) )

                    }
          )
```

Name list elements "smallBird", "mediumBrid" and "Bigbird".
What can you say about offspring number and weight for the list elements?

Compute mean number of offspring and mean body weight for each bird and mean of both body weight and offspring number (use lapply and apply)

```
>str(birds)
List of 3
 $ :'data.frame':	10 obs. of  2 variables:
  ..$ offspring: int [1:10] 1 4 4 8 10 7 10 10 11 4
  ..$ weight   : num [1:10] 16.6 19.3 20.2 19.2 15.9 17.8 19.8 16.2 19.5 17.6
 $ :'data.frame':	10 obs. of  2 variables:
  ..$ offspring: int [1:10] 2 2 3 12 8 9 1 11 8 6
  ..$ weight   : num [1:10] 16.4 21.6 18 16.4 14.4 18.7 20.2 16.5 18 18
 $ :'data.frame':	10 obs. of  2 variables:
  ..$ offspring: int [1:10] 2 9 11 5 5 12 12 1 6 1
  ..$ weight   : num [1:10] 17.2 22.8 16.9 21.1 19.5 15.5 18.2 21.7 16.2 20.9
```

```
names(birds)=c("smallBird","mediumBird","bigBird")
lapply(birds,mean)
lapply(birds,function(x){apply(x,1,mean)})
```

Set operations

A=c(1,4,7,19)		B=c(7,2,31,19)
`intersect(A,B)`	$A \cap B$	7,19
`union(A,B)`	$A \cup B$	1,4,2,7,19,31
`setdiff(A,B)`	A - B	1,4

`%in%`

a logical vector indicating if there is a match or not for its left operand
match() a vector giving the position in table of the first match if there is a match, otherwise nomatch

```
>1:10 %in% c(1,3,5,9)

> match(1:10,  c(1,3,5,9))

> which(1:10 %in% c(1,3,5,9))
> fruits<-
c("apple","pear","plum","grape","tomatoe","cucumber")

> vegetables<-c("potato","tomatoe","onion","cucumber")
```

Check with intersect common elements of both vectors

Add sets.

What elements are only in fruits?

What elements are only in vegetables?

Prepare a vector with 10 random letters of alphabet (use sample() and letters)
Prepare a vector with another 10 random letters of alphabet
Check how many letters appear in one/both sets.
Repeat, using sampling with replacement.
Check which letters reappeared.

Load the data from a file exercices6.Rdata
Check, how many genes have genes1 and genes2 in common.
How many genes do they contain in total?
How many genes are present on genes1 and not on genes2 and vice versa?
For common genes, prepare a dataframe with expression values from both lists.

```
source (file)
```

```
source (“aniaFunc.R”)
```

```
ifelse(condition, yes, no)
switch(expression, var1=whatToDo,var2=whatToDo)
break
next
```

```
>example.ifelse

function(x){

       z=ifelse(x >= 0, sqrt(x), NA)

       return(z)
       }


> example.break
function(x){

   for(i in 1:x){

       print(i)

       if(i>5){
          break

          }
       }
   }
```

```
example.ifelse(9)
example.ifelse(-2)
```

```
example.break(3)
example.break(23)
```

```
> centre
function(x, type) {

  switch(type,
         mean = mean(x),
         median = median(x),
         trimmed = mean(x, trim = .
1))

}
> centre(1:20,"mean")
> example.next
function(A,B){

    for (i in 1:A){

        for (j in 1:B){

            if (j==3) {

                next

                }

            else {

                cat (i,j,"\n")

                }
        }

    }
}
> example.next(3,5)
```

```
> example.break2
function(A,B){

    for (i in 1:A){

        for (j in 1:B){

            if (j==3) {

                break

                }

            else {

                cat (i,j,"\n")
                }
        }

    }

}
> example.break2(3,5)
```

HOME
Write a function which, given 2 vectors, writes all possible pairwise combinations of elements from two vectors, but
a) excludes pairs which sum is equal 21
b) for a given element from first vector, writes pairs only until it finds first pair with sum equal 21.
c) Depending on third argument, “unit”, prints out a sentence “this are results in kg” or “this are results in lb” or “this are results in tons”.

Distributions: d, p, q, r (density, distribution function, quantile function and random generation)

```
>a=rpois(1000,3)
>hist(a)
>summary(a)

>table(a)
a
  0   1   2   3   4   5   6   7   8   9  10 
 46 133 247 224 165 113  35  27   7   2   1 
```

Statistical tests:

```
> data(sleep)
>sleep

>t.test(extra~group, data=sleep)
>var.test(extra~group,data=sleep)
>t.test(extra~group,data=sleep,var.equal=TRUE)
>a<-t.test(extra~group,data=sleep,var.equal=TRUE)
>str(a)
>a$p.value
```

Check what is the probability of having 9 daughters and 1 son (with binom.test())

Check if mean length of extra sleep is equal 0.5 (with t.test, one sample)

Repeat it with nonparametric wilcoxon test (wilcox.test)

Apply/tapply/lapply with your own function.

Compare, gene by gene, gene expression (data from expression.txt) with t.test.
Samples 1:3 are cancer samples before treatment, 4:6 after treatment, 7:9 are control samples.
For each gene get p.value of a difference between cancer and control group.

Anova

Fit linear model to the data with lm(y~as.factor(x))

```
lm(weight~Time, data=ChickWeight)

anova(lm(weight~Time, data=ChickWeight))
```

Additional terms/interactions may be added:

```
anova(lm(weight~Time+Diet, data=ChickWeight))
```

Which groups are different?

```
pairwise.t.test(ChickWeight$weight,ChickWeight$Diet)
```

Are the assumptions about homogeneity of variances valid?

```
bartlett.test(ChickWeight$weight~ChickWeight$Diet)
```

What if you compare only weight at the last day of the study? Check again which groups are different and homogeneity of variances.

Run row-by-row anova for expression data, for cancer/treatment/control groups.
For each comparison, check variance homogeneity assumption and which of the comparisons are significant.

How to get ANOVA results for one row of expression?

How to get pval out of it?

Run this for the whole dataframe with apply.

```
>install.packages("ape")
> library(ape)
> vignette(package="ape")
```

Use a built-in dataset woodmouse.
Compute distance between sequences with your substitution model of choice(dist.dna()).
Plot an unrooted tree.

Read in the data from file snpdata.txt
Compute distances between individuals with function dist(), using “Manhattan” metrics
Prepare a tree based on distances, with function bionj()
Plot an unrooted tree.