Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

UNIVERSITY OF CAMBRIDGE

Introduction to R, Github and Gitlab

27/11/2018

Pierpaolo Maisano Delser

mail: maisanop@tcd.ie ; pm604@cam.ac.uk

Outline:

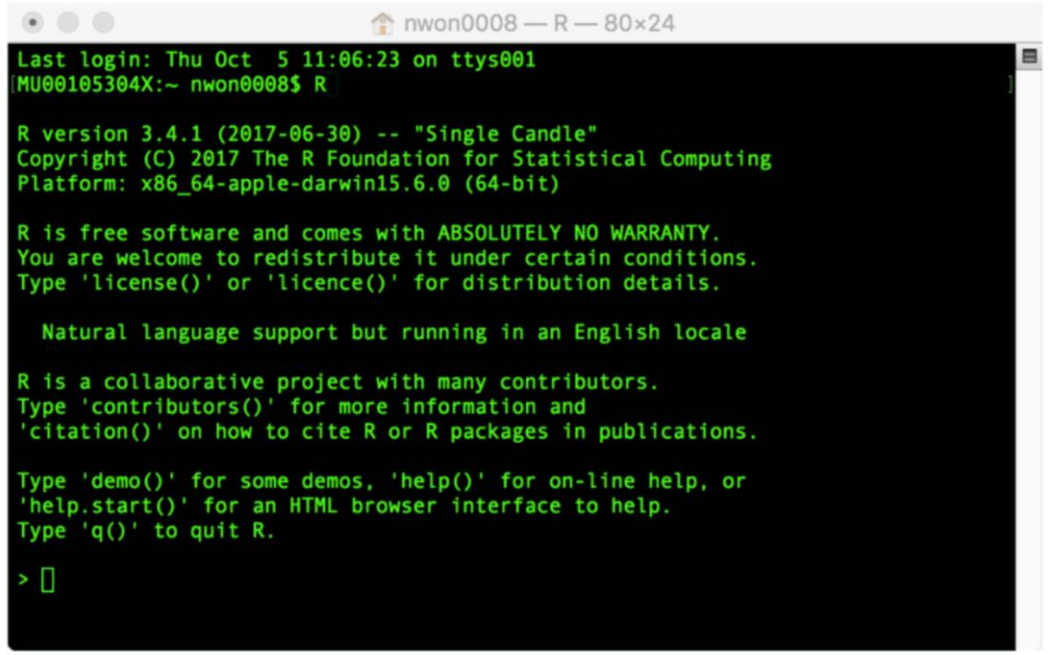
- Why R? What can R do?
- Basic commands and operations
- Data analysis in R
- Github and Gitlab

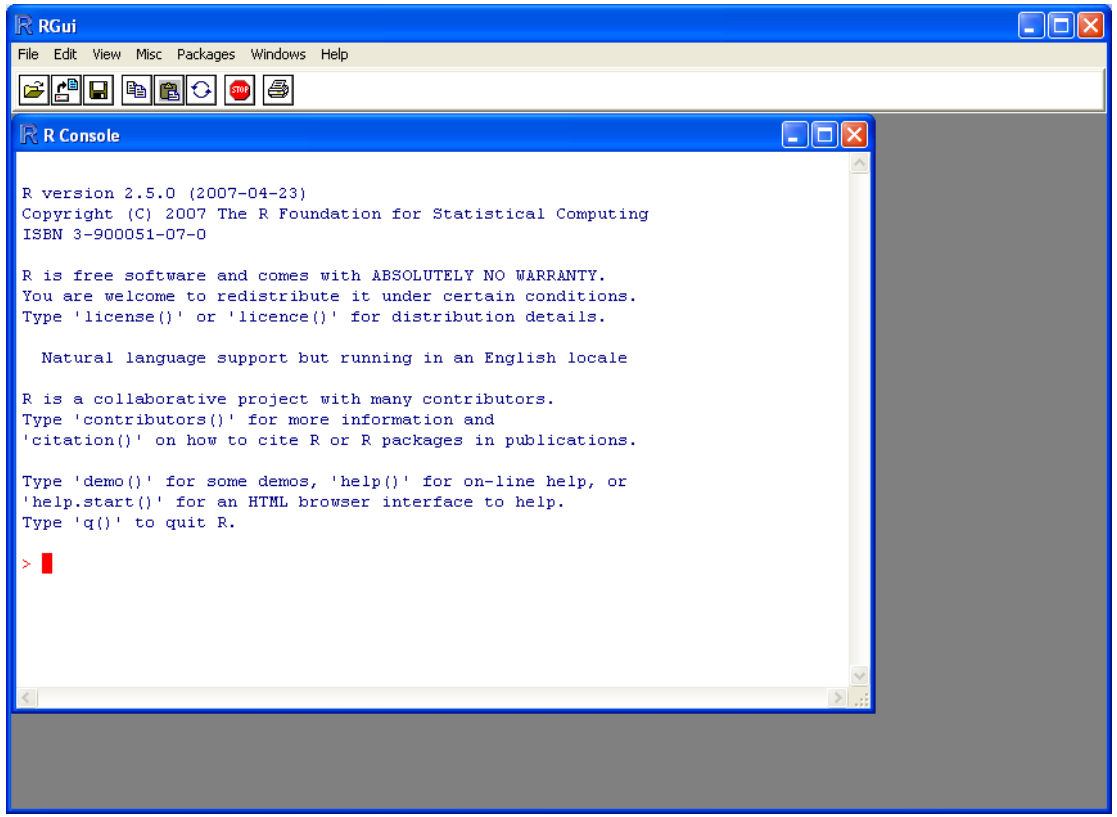
Outline:

- Why R? What can R do?
- Basic commands and operations
- Data analysis in R
- Github and Gitlab

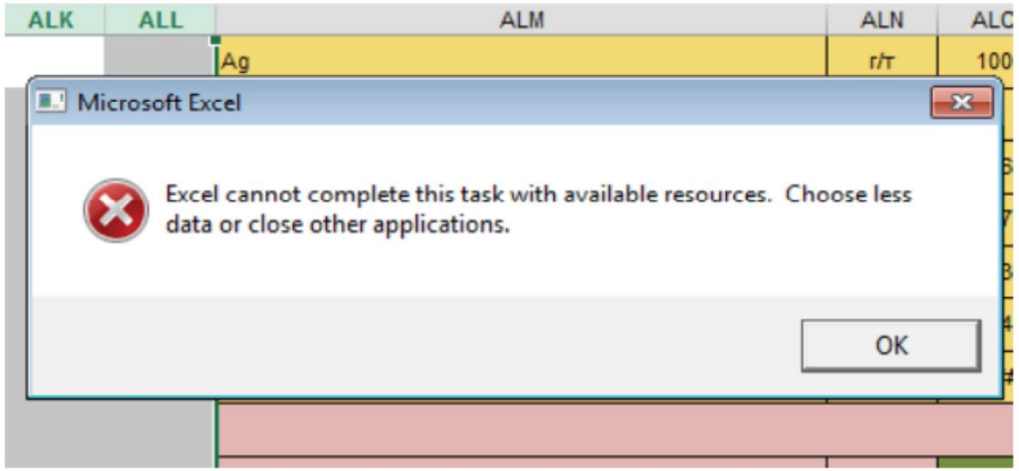
Why R? What can R do?

- R is a language and environment for statistical computing and graphics;
- It is open source and free software package;
- Lots of resources, packages and support





R vs Excel



Ziemann et al. Genome Biology (2016) 17:177
DOI 10.1186/s13059-016-1044-7

Genome Biology

COMMENT Open Access

Gene name errors are widespread in the scientific literature

Mark Ziemann[1], Yotam Eren[1,2] and Assam El-Osta[1,3*]

Abstract

The spreadsheet software Microsoft Excel, when used with default settings, is known to convert gene names to dates and floating-point numbers. A programmatic scan of leading genomics journals reveals that approximately one-fifth of papers with supplementary Excel gene lists contain erroneous gene name conversions.

Keywords: Microsoft Excel, Gene symbol, Supplementary data

Abbreviations: GEO, Gene Expression Omnibus; JIF, journal impact factor

frequently reused. Our aim here is to raise awareness of the problem.

We downloaded and screened supplementary files from 18 journals published between 2005 and 2015 using a suite of shell scripts. Excel files (.xls and.xlsx suffixes) were converted to tabular separated files (tsv) with ssconvert (v1.12.9). Each sheet within the Excel file was converted to a separate tsv file. Each column of data in the tsv file was screened for the presence of gene symbols. If the first 20 rows of a column contained five or more gene symbols, then it was suspected to be a list of gene symbols, and then a regular expression (regex) search of the entire column was applied to identify gene symbol errors. Official gene symbols from Ensembl version 82, accessed November 2015, were obtained for

- Dragging and updating formula;
- Spreadsheets can get really big and confusing;
- Lack of quick way to get a summary of the data;
- …

R vs Excel

- Point-and-click software is not time efficient;
- Automating tasks will pay off within the time frame of a PhD and thereafter

R is more efficient

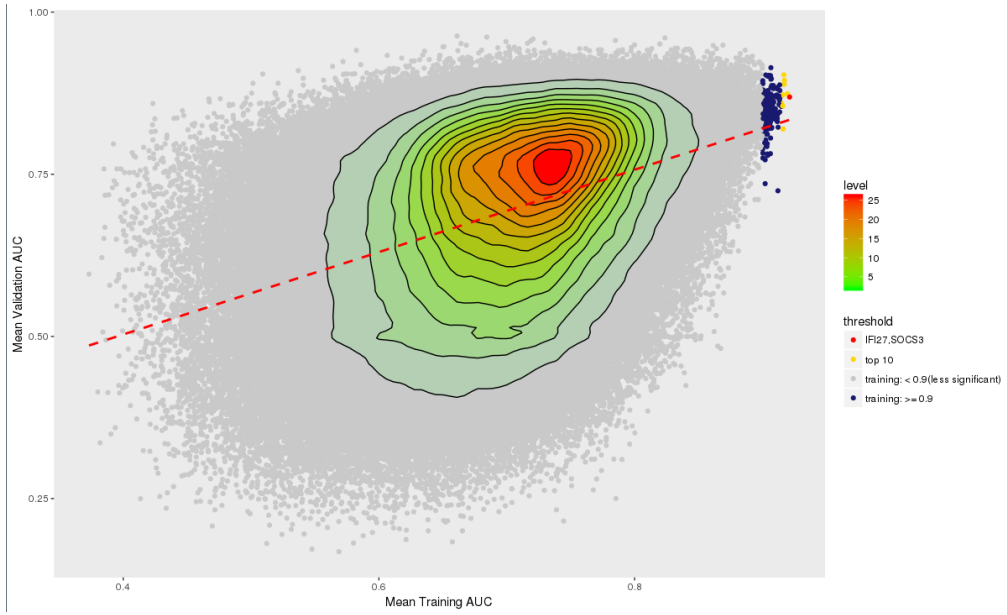
R vs Excel

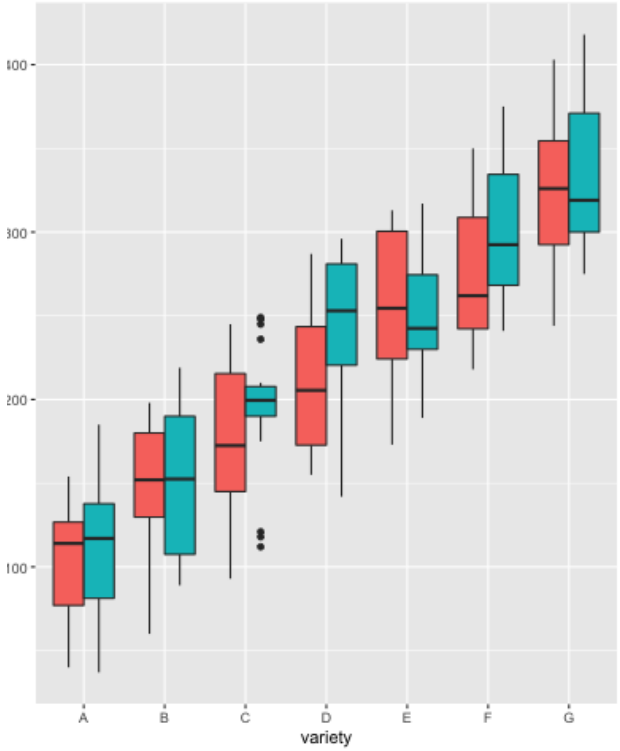
- **Reproducibility:** there is an increasing expectation that material, data and analysis details are provided alongside the research, this is easier when analyses are script based.

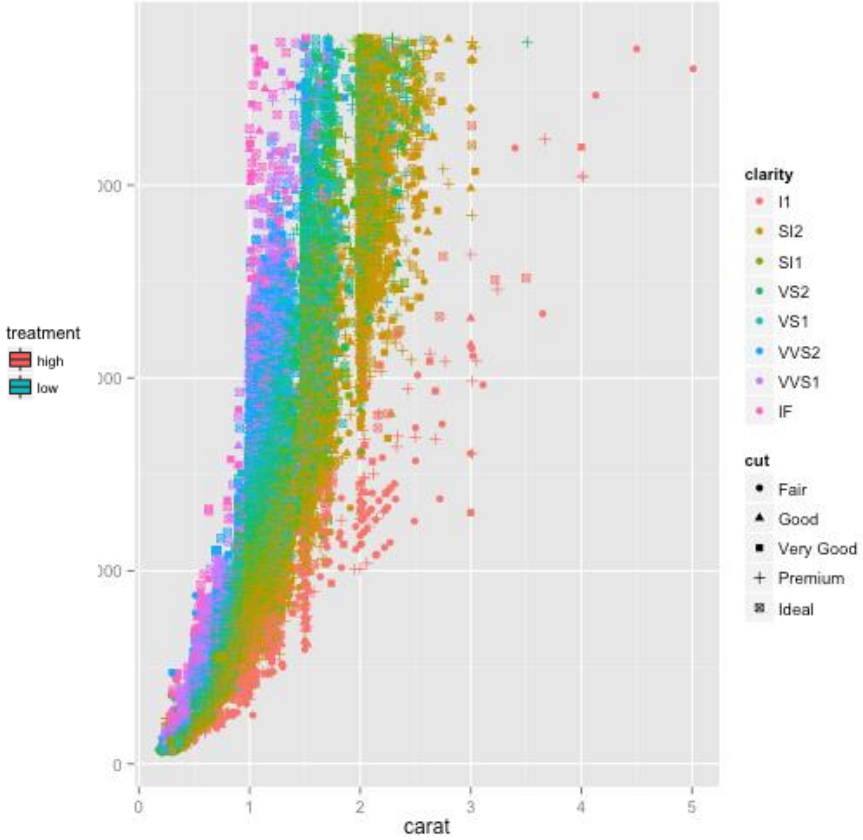
Flexibility:

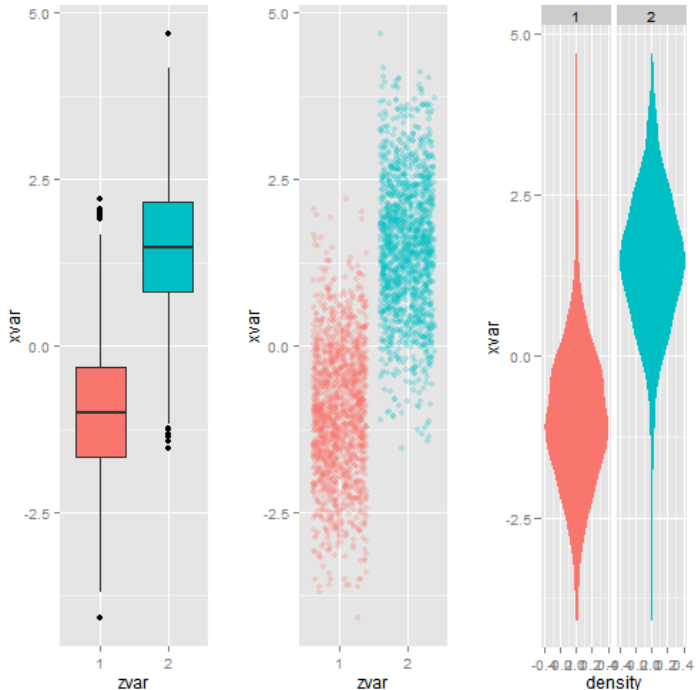
- Read different file formats;
- Compute analysis;
- Generate graphs and plots;
- Summary of your data;
- R works on vector, matrix and dataframe;
- Generate reports;

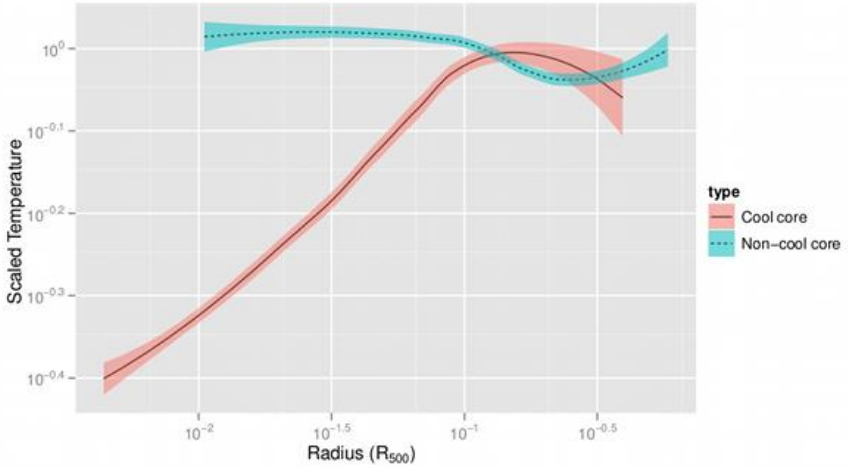
Why R? What can R do?

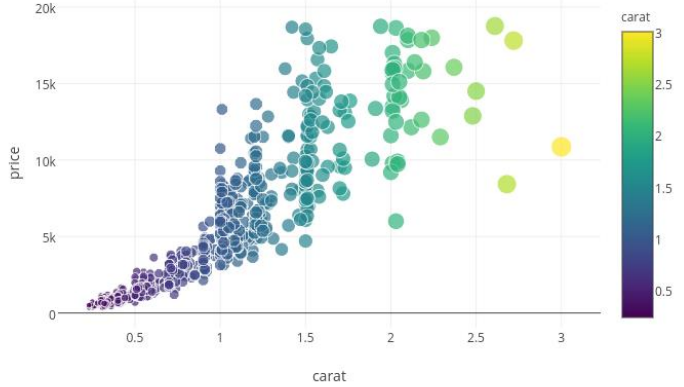












Advantages:

- Fast and free
- Statistical researchers provide their methods as R packages
- Good graphics (MATLAB and python)
- Active user community (great support)
- Excellent data analysis;
- Forces you to think about your analysis
- Functions can be integrated in R packages

Advantages:

- Fast and free
- Statistical researchers provide their methods as R packages
- Good graphics (MATLAB and python)
- Active user community (great support)
- Excellent data analysis;
- Forces you to think about your analysis
- Functions can be integrated in R packages

Disadvantages:

- “Not user friendly at start - steep learning curve, minimal GUI”
- Easy to make mistakes and not know.
- Working with large datasets is limited by RAM

Advantages:

- Fast and free
- Statistical researchers provide their methods as R packages
- Good graphics (MATLAB and python)
- Active user community (great support)
- Excellent data analysis;
- Forces you to think about your analysis
- Functions can be integrated in R packages

Disadvantages:

- "Not user friendly at start - steep learning curve, minimal GUI"
- Easy to make mistakes and not know.
- **Working with large datasets is limited by RAM**

Disadvantages:

- "Not user friendly at start - steep learning curve, minimal GUI"

Advantages:

- **Functions** can be integrated in R packages

Advantages:

- **Functions** can be integrated in R packages

```
t<-read.table("/home/pier/data/input_stat")

dim(t) #number of columns and rows
```

Advantages:

- **Functions** can be integrated in R packages

```
t<-read.table("/home/pier/data/input_stat")

dim(t) #number of columns and rows

#I want to calculate the sum of each column
and if it is > 10, print out "column X has
sum greater than 10"

for (i in 1:ncol(t)) {
      f<-sum(t[,i])
      if (f>10) {
            print (paste("Column ", i, "
has a sum greater than 10", sep=""))
      }
      else {
      }
}
```

Advantages:

- **Functions** can be integrated in R packages

```
sum_10 <- function(table) {

for (i in 1:ncol(table)) {
      f<-sum(table[,i])
      if (f>10) {
            print
(paste("Column ", i, " has a
sum greater than 10", sep=""))
      }
      else {
      }
}

}
```

```
t<-read.table("/home/pier/data/input_stat")

dim(t) #number of columns and rows

#I want to calculate the sum of each column
and if it is > 10, print out "column X has
sum greater than 10"

for (i in 1:ncol(t)) {
      f<-sum(t[,i]
      if (f>10) {
            print (paste("Column ", i, "
has a sum greater than 10", sep=""))
      }
      else {
      }
}
```

Advantages:

- **Functions** can be integrated in R packages

```
sum_10 <- function(table) {

for (i in 1:ncol(table)) {
      f<-sum(table[,i])
      if (f>10) {
            print
(paste("Column ", i, " has a
sum greater than 10", sep=""))
      }
      else {
      }
}

}
```

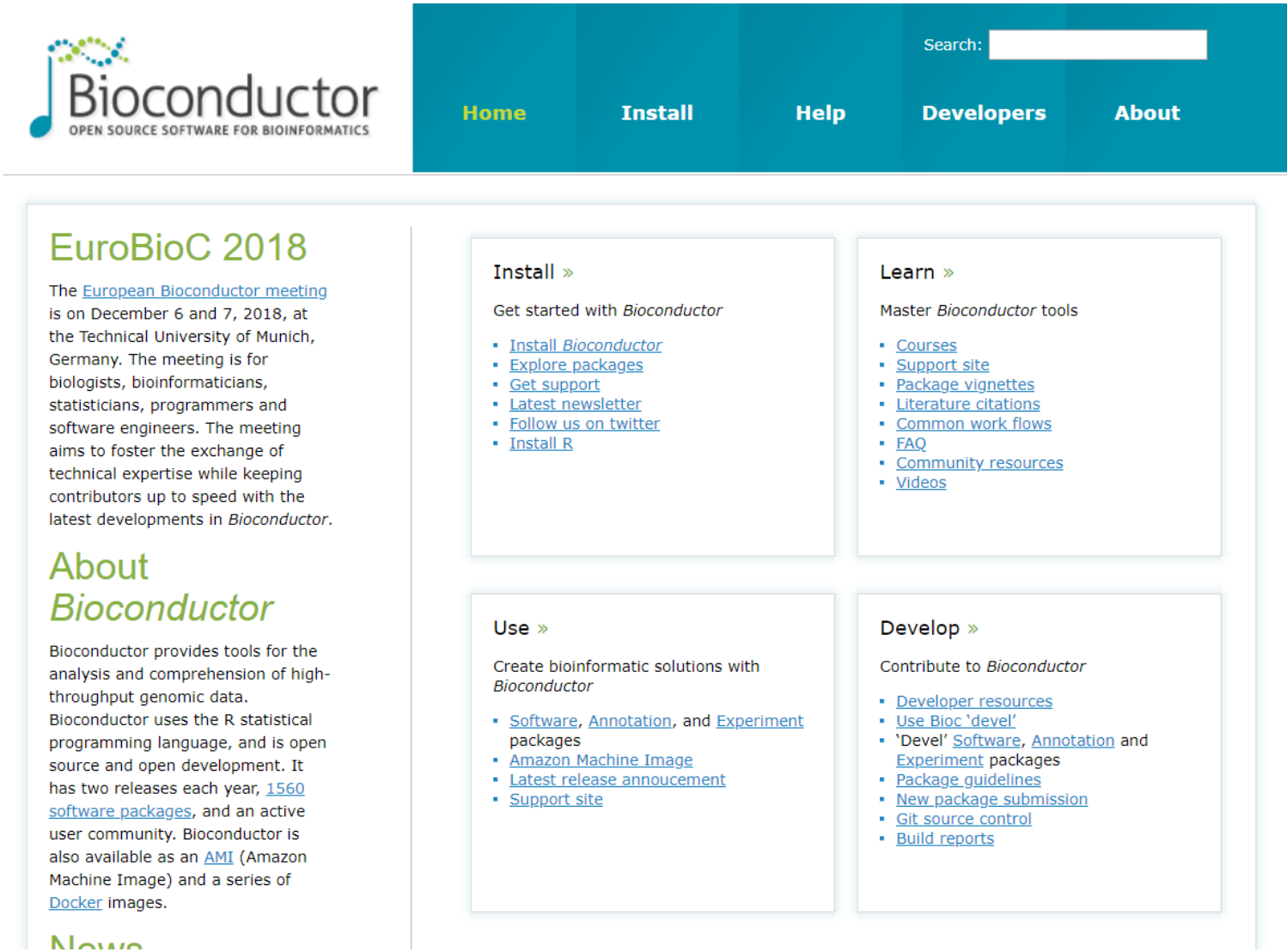
```
t<-read.table("/home/pier/data/input_stat")

dim(t) #number of columns and rows

#I want to calculate the sum of each column
and if it is > 10, print out "column X has
sum greater than 10"

sum_10(t)
```

Advantages:

- Active user community (great support)

R packages for:

- Statistical analysis;
- Plotting;
- Graphs;
- Managing calendar dates;
- Selecting colour palette;
- Machine learning;
- Population genetics;
- …



Outline:

- Why R? What can R do?
- Basic commands and operations
- Data analysis in R
- Github and Gitlab

Variables:

- Alphanumeric symbols, plus “.” and “_” are allowed;
- Variables are case sensitive, so “T” is different from “t”;

Variables:

- Alphanumeric symbols, plus “.” and “_” are allowed;
- Variables are case sensitive, so “T” is different from “t”;

Assignment “<-”:

- t<-3, T<-5, X.1<-9;
- In 2001, the “=“ assignment was introduced for compatibility with other languages;

Variables:

- Alphanumeric symbols, plus “.” and “_” are allowed;
- Variables are case sensitive, so “T” is different from “t”;

Assignment “<-”:

- t<-3, T<-5, X.1<-9;
- In 2001, the “=“ assignment was introduced for compatibility with other languages;

```
> t<-3

> k<-“hello”

> a<-1.435275289
```

Arithmetic operators:

- Addition: +
- Subtraction: -
- Division: /
- Multiplication: *
- Exponentiation: ^

- We can use R as a calculator:

```
> (3+2)^2
[1] 25

> (7-5)/2
[1] 1

> 1*2*3*4
[1] 24
```

Data type:

- Numeric, character and logical

```
> t<-2.356  #numeric variable

> t<-”hello” #character variable

> t<-TRUE; f<-FALSE #logical variables
```

Data structure:

- **Vector**: an ordered collection of data;
- **Matrix**: two-dimensional generalisations of vectors
- **Array**: multi-dimensional generalisations of vectors

c function for concatenating values and vectors to create longer vectors

Data structure:

- **Vector**: an ordered collection of data;
- **Matrix**: two-dimensional generalisations of vectors
- **Array**: multi-dimensional generalisations of vectors

```
> t<-c(3, 5.6748, 67, 5) #numeric vector

> t<-c(1, 1:3, c(5, 8), 13)
[1] 1  1  2  3  5  8 13

> t<-c("hello", "how", "are", "you", "?") #character vector
```

c function for concatenating values and vectors to create longer vectors

Data structure:

- **Vector**: an ordered collection of data;
- **Matrix**: two-dimensional generalisations of vectors
- **Array**: multi-dimensional generalisations of vectors

```
> t<-c(3, 5.6748, 67, 5) #numeric vector

> length(t)
[1] 4
```

c function for concatenating values and vectors to create longer vectors

Data structure:

- **Vector**: an ordered collection of data;
- **Matrix**: two-dimensional generalisations of vectors
- **Array**: multi-dimensional generalisations of vectors

```
> matrix(c(1,2,3,4,5,6), ncol=2, nrow=3)

      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

c function for concatenating values and vectors to create longer vectors

Data structure:

- **Vector**: an ordered collection of data;
- **Matrix**: two-dimensional generalisations of vectors
- **Array**: multi-dimensional generalisations of vectors

```
> t<- matrix(c(1,2,3,4,5,6), ncol=2, nrow=3)

     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

> dim(t)
[1] 3 2
```

c function for concatenating values and vectors to create longer vectors

Data structure:

- **Vector**: an ordered collection of data;
- **Matrix**: two-dimensional generalisations of vectors
- **Array**: multi-dimensional generalisations of vectors

```
> matrix(c(1,2,3,4,5,6), ncol=2, nrow=3, dimnames=list(c("Variable_1",
"Variable_2", "Variable_3"), c("factor_1","factor_2")))


            factor_1 factor_2
Variable_1        1        4
Variable_2        2        5
Varibale_3        3        6
```

c function for concatenating values and vectors to create longer vectors

Data structure:

- **Vector**: an ordered collection of data;
- **Matrix**: two-dimensional generalisations of vectors
- **Array**: multi-dimensional generalisations of vectors

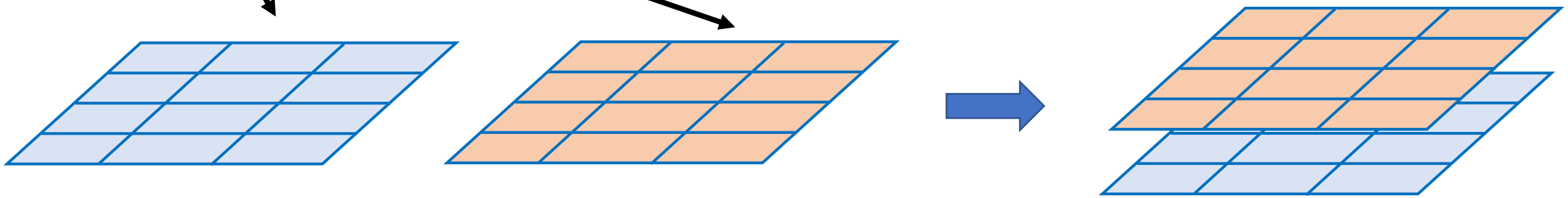
```
> array(1:24,dim = c(4, 3, 2), dimnames = list(c("one", "two", "three",
"four"),c("apple", "orange", "pear"),
    c("land", "sea")))
```

c function for concatenating values and vectors to create longer vectors

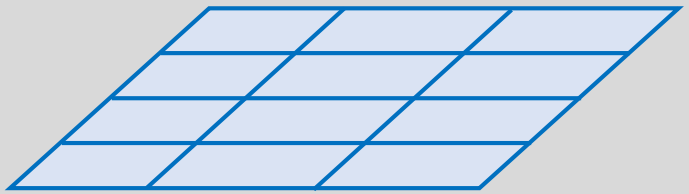
Data structure:

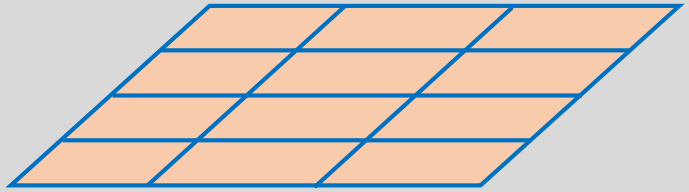
- **Vector**: an ordered collection of data;
- **Matrix**: two-dimensional generalisations of vectors
- **Array**: multi-dimensional generalisations of vectors

```
> array(1:24,dim = c(4, 3, 2), dimnames = list(c("one", "two", "three",
"four"),c("apple", "orange", "pear"),
    c("land", "sea")))
```



```
> array(1:24,dim = c(4, 3, 2), dimnames = list(c("one", "two", "three",
"four"),c("apple", "orange", "pear"),
    c("land", "sea")))

, , land

      apple orange pear
one       1      5    9
two       2      6   10
three     3      7   11
four      4      8   12

, , sea

      apple orange pear
one      13     17   21
two      14     18   22
three    15     19   23
four     16     20   24
```





Data structure:

- **Data frame**: are matrix-like structure but the columns can be of different data types (i.e. numerical and character)

```
> data.frame(weight=c(1,23,4,56,32), gender=c("M","F","F","M","F"))

      weight gender
1      1      M
2     23      F
3      4      F
4     56      M
5     32      F
```

Indexing and selecting:

Indexing and selecting:

- **Vector:**

```
> t<-c(3, 5.6748, 67, 5) #numeric vector

> length(t)
[1] 4

> t[3]
[1] 67

> t[c(2,4)]
[1] 5.6748  5
```

Indexing and selecting:

- **Matrix:**

```
> t<- matrix(c(1,2,3,4,5,6), ncol=2, nrow=3)
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

> dim(t)
[1] 3 2

> t[1,] #row 1
[1] 1 4

> t[,2] #column 2
[1] 4 5 6
```

Indexing and selecting:

- **Matrix:**

```
> t<- matrix(c(1,2,3,4,5,6), ncol=2, nrow=3)
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

> dim(t)
[1] 3 2

> t[3,2] #specific cell
[1] 6
```

Indexing and selecting:

- **Matrix:**

```
> t <- matrix(c(1,2,3,4,5,6), ncol=2, nrow=3, dimnames=list(c("Variable_1",
"Variable_2", "Variable_3"), c("factor_1","factor_2")))


            factor_1 factor_2
Variable_1        1        4
Variable_2        2        5
Varibale_3        3        6

> t["Variable_3", "factor_2"]
[1] 6
```

Indexing and selecting: Array

```
> t <- array(1:24,dim = c(4, 3, 2), dimnames = list(c("one", "two",
"three", "four"),c("apple", "orange", "pear"),
    c("land", "sea")))

, , land

      apple orange pear
one       1      5    9
two       2      6   10
three     3      7   11
four      4      8   12

, , sea

      apple orange pear
one      13     17   21
two      14     18   22
three    15     19   23
four     16     20   24
```

Indexing and selecting: Array

```
> t <- array(1:24,dim = c(4, 3, 2), dimnames = list(c("one", "two",
"three", "four"),c("apple", "orange", "pear"),
    c("land", "sea")))

, , land

      apple orange pear
one       1      5    9
two       2      6   10
three     3      7   11
four      4      8   12

, , sea

      apple orange pear
one      13     17   21
two      14     18   22
three    15     19   23
four     16     20   24
```

```
> t[3,1,2]
[1] 15

> t["two","orange","land"]
[1] 6
```

Deleting:

```
> t<-10:20 #vector

> t
 [1] 10 11 12 13 14 15 16 17 18 19 20

> t[-2] #remove element in position number 2
 [1] 10 12 13 14 15 16 17 18 19 20

> t1<-t[-2]

> t1
[1] 10 12 13 14 15 16 17 18 19 20
```

Deleting:

```
> t<- matrix(c(1,2,3,4,5,6), ncol=3, nrow=2)

> t
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> t[-1,] #remove row1
[1] 2 4 6

> t[,-2] #remove col2
     [,1] [,2]
[1,]    1    5
[2,]    2    6
```

Commands/operations:

- paste text strings together;
- append element to vectors;
- operations between vectors and matrices (sum, difference);
- “apply” a specific rule/function to all columns/rows of a matrix;
- match specific elements;
- subset a vector or a matrix;
- …and many more!!

Commands/operations:

- paste text strings together;
- append element to vectors;
- operations between vectors and matrices (sum, difference);
- “apply” a specific rule/function to all columns/rows of a matrix;
- match specific elements;
- subset a vector or a matrix;
- …and many more!!

- “which” element satisfies a specific condition…

```
> t<-10:20 #vector
> t
 [1] 10 11 12 13 14 15 16 17 18 19 20

> which(t>15)
[1]  7  8  9 10 11

> t[which(t>15)]
[1] 16 17 18 19 20
```

Loops and conditional execution

Syntax:

```
for (variable in sequence) {
        statements
}
```

Loops and conditional execution

Syntax:

for (variable in sequence) {
statements
}

```
> for (i in 1:10) {
+ print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

```
> for (i in 1:10) {
+ print(paste("hello to our customer number ",
i, sep=""))
+ }
[1] "hello to our customer number 1"
[1] "hello to our customer number 2"
[1] "hello to our customer number 3"
[1] "hello to our customer number 4"
[1] "hello to our customer number 5"
[1] "hello to our customer number 6"
[1] "hello to our customer number 7"
[1] "hello to our customer number 8"
[1] "hello to our customer number 9"
[1] "hello to our customer number 10"
```

Loops and conditional execution

Comparison operators

equal: ==
not equal: !=
greater: >
less than: <
greater or equal: >=
less than or equal: <=

Logical operators

and: &
or: |
not: !

Loops and conditional execution

Comparison operators

equal: ==
not equal: !=
greater: >
less than: <
greater or equal: >=
less than or equal: <=

Logical operators

and: &
or: |
not: !

Syntax:

```
If (condition) {

        statement

} else {

        alternative

}
```

Loops and conditional execution

Syntax:

If (condition) {

statement

} else {

alternative

}

```
> x<--4

> if(x>0){

+ print("Positive number")

+ } else if (x==0) {

+ print("Zero")

+ } else {

+ print("Negative number")

+ }

[1] "Negative number”
```

User-defined functions

```
myFunction <- function(arg1, arg2,..) {
        function_body
}

myFunction(arg1=..., arg2=...)
```

User-defined functions

```
myFunction <- function(arg1, arg2,..) {
        function_body
}

myFunction(arg1=..., arg2=...)
```

```
> myvar<-function(x) {
+ y<-sum((x-mean(x))^2)/(length(x)-1)
+ return(y)
+ }

> a<-rnorm(6)

> a
[1] -0.9379583  0.6599282  0.6204624
0.4395611  1.0989696  2.4148308

> var(a)
[1] 1.171392

> myvar(a)
[1] 1.171392
```

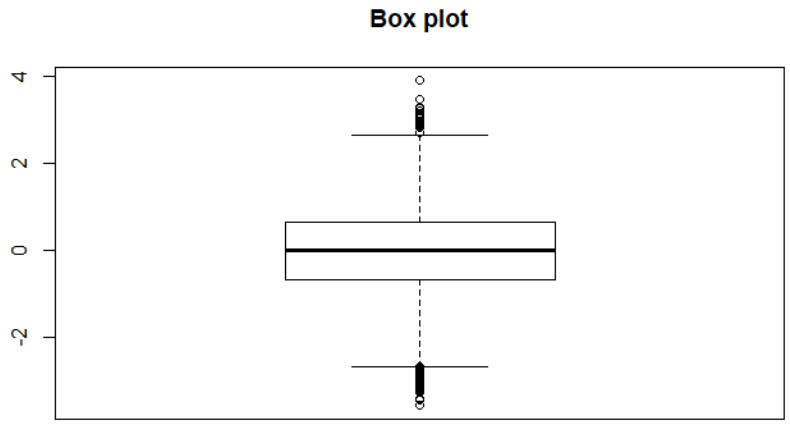
Outline:

- Why R? What can R do?
- Basic commands and operations
- Data analysis in R
- Github and Gitlab

- Descriptive Statistics
- Statistical Modeling
  - Regressions: Linear and Logistic;
  - Time Series;
  - …
- Multivariate Functions
- Bayesian statistics
- Machine learning
- Inbuilt Packages, contributed packages

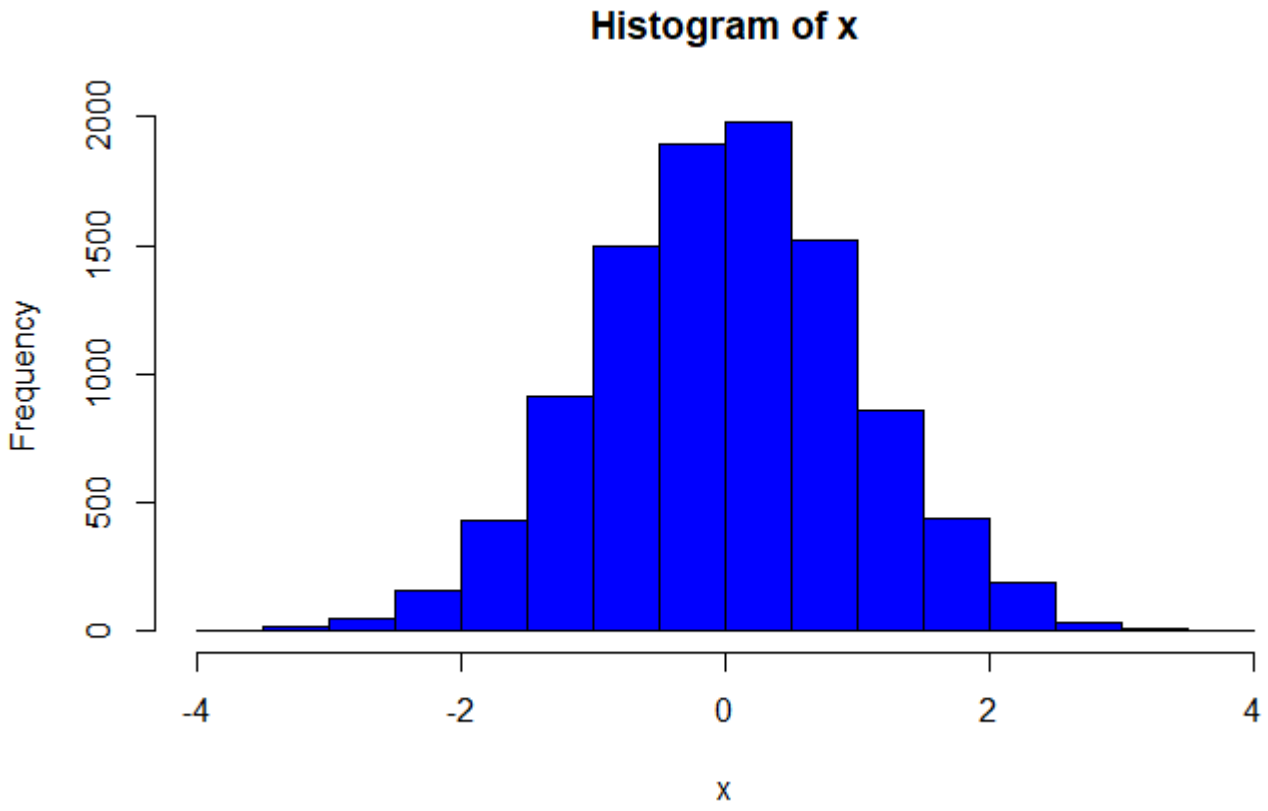
Basic statistical analysis

```
#generate random number form normal distribution
> x<-rnorm(10000,0,1)

#get a summary of the distribution
> summary(x)
 Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-3.55700 -0.66420  0.00822  0.00131  0.66440  3.90500

> boxplot(x, main="Box plot")
```



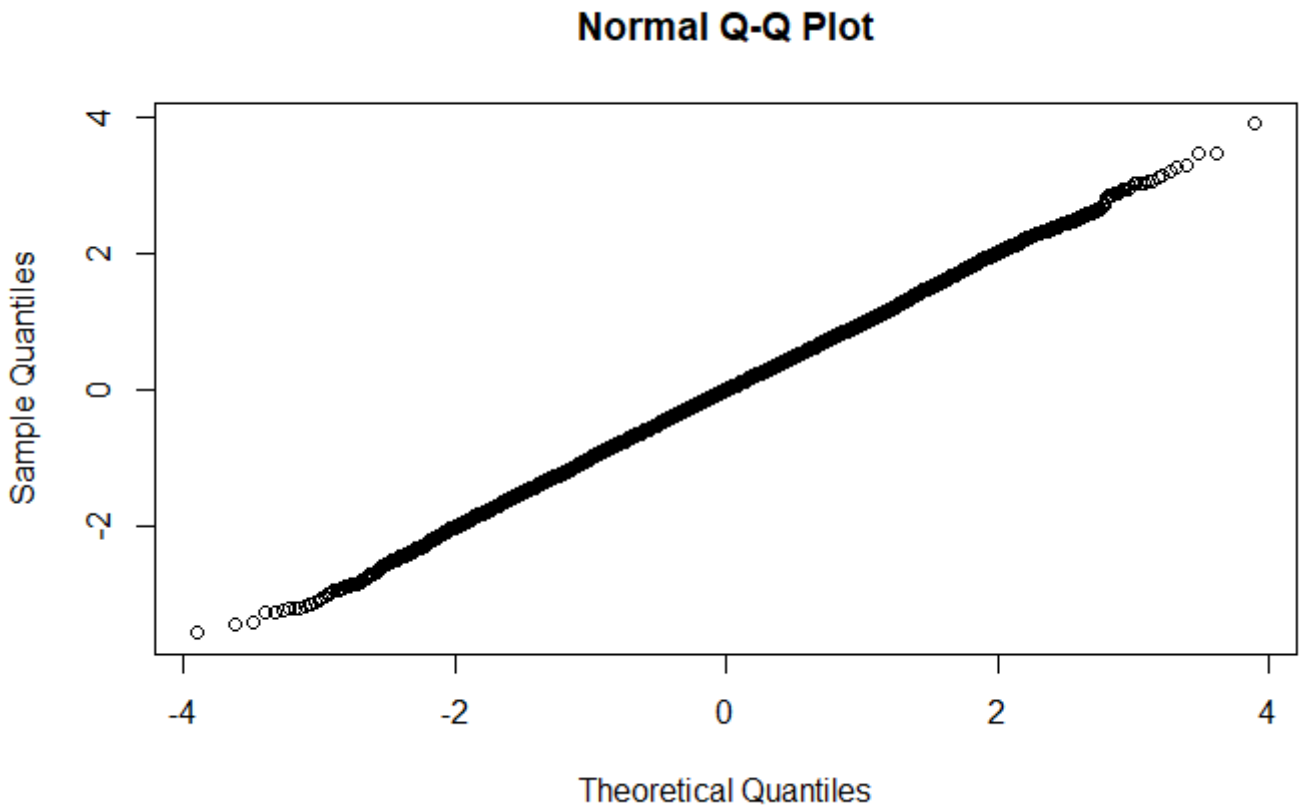
Basic statistical analysis

```
> hist(x, col="blue")
```



Basic statistical analysis

```
> qqnorm(x)
```



Parallel boxplots

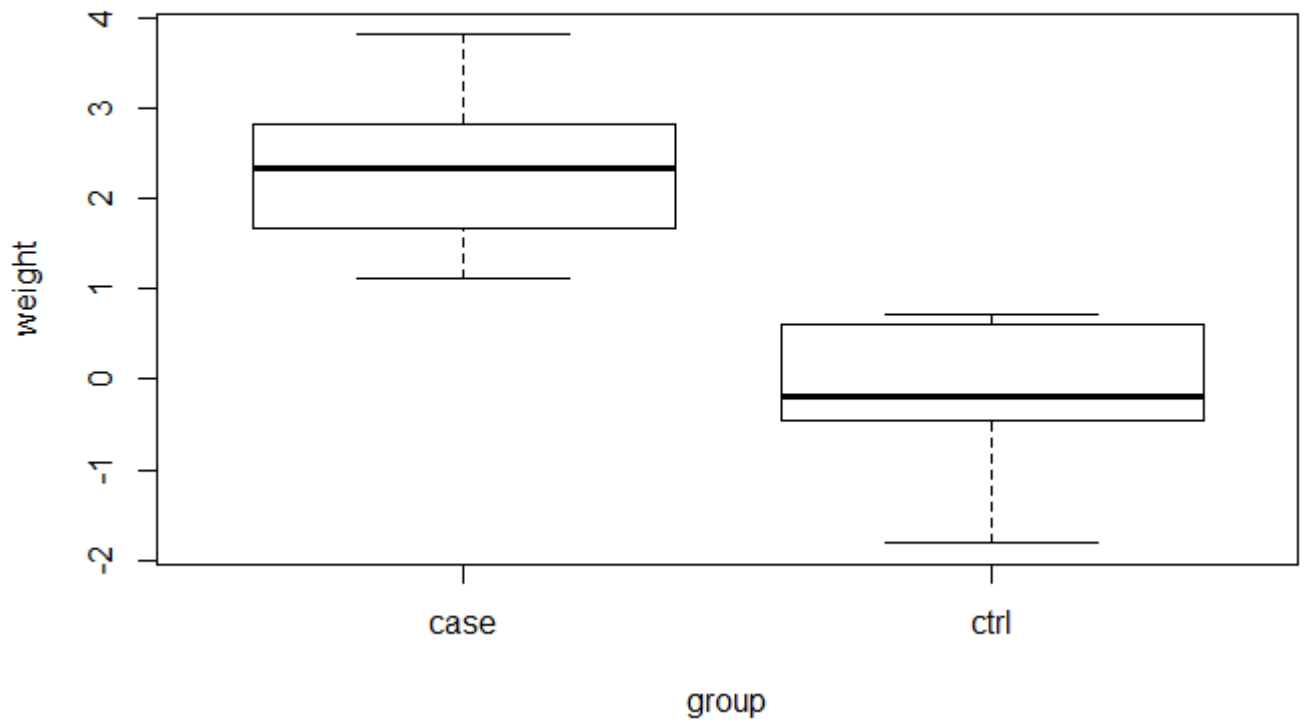
```
> set.seed(12345)

> weight<-
round(c(rnorm(10,0,1),rnorm(10,2,
1)),3)

> group<-rep(c("ctrl","case"),
each=10)

> mydata<-data.frame(weight,
group)

> plot(weight~group, mydata)
```

Parallel boxplots

```
> set.seed(12345)

> weight<-
round(c(rnorm(10,0,1),rnorm(10,2,
1)),3)

> group<-rep(c("ctrl","case"),
each=10)

> mydata<-data.frame(weight,
group)

> plot(weight~group, mydata)
```

```
 weight group
1   0.586  ctrl
2   0.709  ctrl
3  -0.109  ctrl
4  -0.453  ctrl
5   0.606  ctrl
6  -1.818  ctrl
7   0.630  ctrl
8  -0.276  ctrl
9  -0.284  ctrl
10 -0.919  ctrl
11  1.884  case
12  3.817  case
13  2.371  case
14  2.520  case
15  1.249  case
16  2.817  case
17  1.114  case
18  1.668  case
19  3.121  case
20  2.299  case
```

Parallel boxplots




```
 weight group
1   0.586  ctrl
2   0.709  ctrl
3  -0.109  ctrl
4  -0.453  ctrl
5   0.606  ctrl
6  -1.818  ctrl
7   0.630  ctrl
8  -0.276  ctrl
9  -0.284  ctrl
10 -0.919  ctrl
11  1.884  case
12  3.817  case
13  2.371  case
14  2.520  case
15  1.249  case
16  2.817  case
17  1.114  case
18  1.668  case
19  3.121  case
20  2.299  case
```

T-test

```
> t.test(weight~group, mydata)

      Welch Two Sample t-test

data:  weight by group
t = 6.5335, df = 17.979, p-value = 3.873e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.640945 3.196655
sample estimates:
mean in group case mean in group ctrl
            2.2860            -0.1328
```

A t-test is a linear regression…

```
> summary(lm(weight~group, mydata))

Call:
lm(formula = weight ~ group, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.6852 -0.4560  0.0184  0.7238  1.5310 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.2860     0.2618   8.733 6.89e-08 ***
groupctrl    -2.4188     0.3702  -6.534 3.85e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8278 on 18 degrees of freedom
Multiple R-squared:  0.7034,	Adjusted R-squared:  0.6869 
F-statistic: 42.69 on 1 and 18 DF,  p-value: 3.85e-06
```

A t-test is a linear regression…

```
> summary(lm(weight~group, mydata))

Call:
lm(formula = weight ~ group, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.6852 -0.4560  0.0184  0.7238  1.5310 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.2860     0.2618   8.733 6.89e-08 ***
groupctrl    -2.4188     0.3702  -6.534 3.85e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8278 on 18 degrees of freedom
Multiple R-squared:  0.7034,	Adjusted R-squared:  0.6869 
F-statistic: 42.69 on 1 and 18 DF,  p-value: 3.85e-06
```

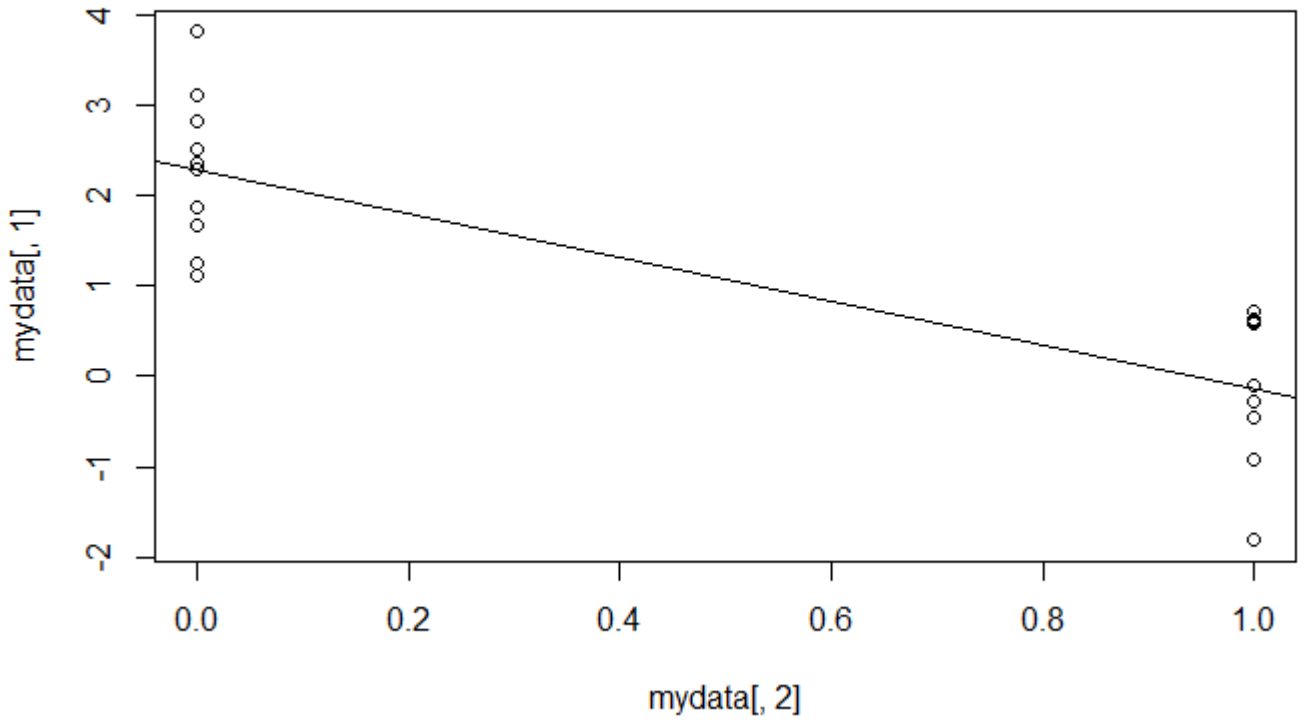
A t-test is a linear regression…

```
> summary(lm(weight~group, mydata))

Call:
lm(formula = weight ~ group, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.6852 -0.4560  0.0184  0.7238  1.5310 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.2860     0.2618   8.733 6.89e-08 ***
groupctrl    -2.4188     0.3702  -6.534 3.85e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8278 on 18 degrees of freedom
Multiple R-squared:  0.7034,	Adjusted R-squared:  0.6869 
F-statistic: 42.69 on 1 and 18 DF,  p-value: 3.85e-06
```

A t-test is a linear regression…

```
> summary(lm(weight~group, mydata))

Call:
lm(formula = weight ~ group, data = mydata)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.6852 -0.4560  0.0184  0.7238  1.5310 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.2860     0.2618   8.733 6.89e-08 ***
groupctrl    -2.4188     0.3702  -6.534 3.85e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8278 on 18 degrees of freedom
Multiple R-squared:  0.7034,	Adjusted R-squared:  0.6869 
F-statistic: 42.69 on 1 and 18 DF,  p-value: 3.85e-06
```

Parallel boxplots




```
 weight group
1   0.586  ctrl
2   0.709  ctrl
3  -0.109  ctrl
4  -0.453  ctrl
5   0.606  ctrl
6  -1.818  ctrl
7   0.630  ctrl
8  -0.276  ctrl
9  -0.284  ctrl
10 -0.919  ctrl
11  1.884  case
12  3.817  case
13  2.371  case
14  2.520  case
15  1.249  case
16  2.817  case
17  1.114  case
18  1.668  case
19  3.121  case
20  2.299  case
```

- Logistic regression;
- Bayesian analysis (Bayer factor, library(BayesFactor);
- ANOVA;
- Approximate Bayesian Computation (ABC) framework;
- ….

Outline:

- Why R? What can R do?
- Basic commands and operations
- Data analysis in R
- Github and Gitlab

- Github: “GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.”

- Github: “GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.”

Coding

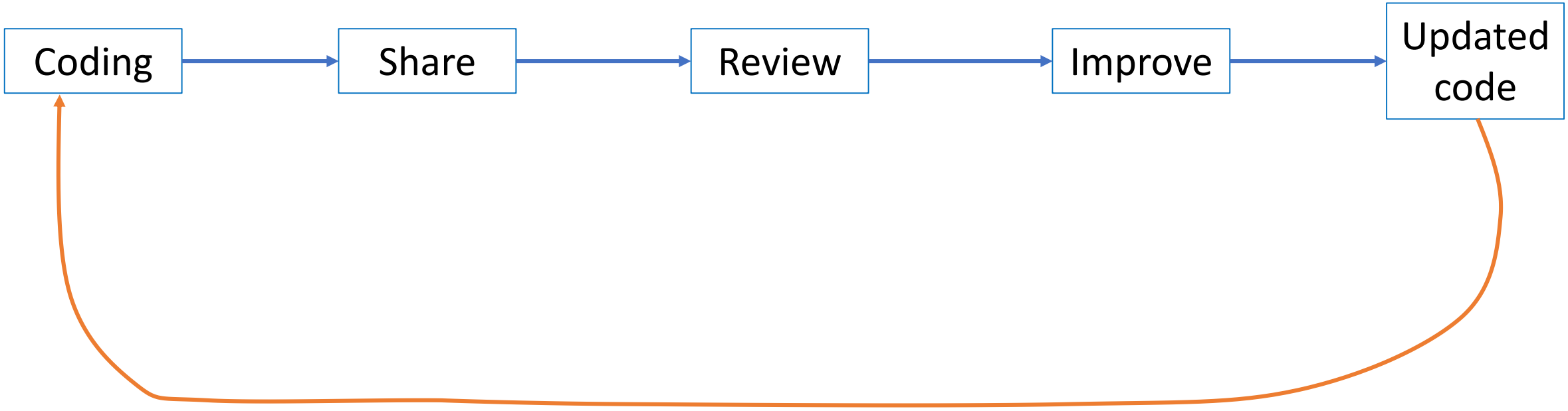
- Github: “GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.”

- Github: “GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.”

- Github: “GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.”

- Github: “GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.”

- Github: “GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.”



Github and Gitlab

Search or jump to... /

Pull requests Issues Marketplace Explore

David Reich Lab

Report abuse

Repositories 8 People 0 Projects 0

Find a repository... Type: All Language: All

ADNA-Tools

This project is a collection of tools for bioinformatic processing of ancient DNA data at the Reich Lab at Harvard Medical School.

Java 1 Updated 5 days ago

AdmixTools

Tools test whether admixture occurred and more

C 58 21 Updated on Sep 27

EIG

Eigen tools by Nick Patterson and Alkes Price lab

Top languages

C Python Java Matlab

People 0

This organization has no public members. You must be a member to see who's a part of this organization.

Github and Gitlab

Search or jump to... / Pull requests Issues Marketplace Explore

Thibaut Jombart
thibautjombart

Lecturer in genetic analysis & outbreak response at Imperial College London. R geek. Muay Thai enthusiast. Vocalist of the grind/hardcore band The Brood.

Follow

Block or report user

Imperial College London
London, UK
thibautjombart@gmail.com

Overview | Repositories 31 | Stars 10 | Followers 114 | Following 9

Popular repositories

adegenet
adegenet: a R package for the multivariate analysis of genetic markers
R 61 27

treescape
Exploring tree diversity
HTML 14 4

treespace
Explore spaces of phylogenetic trees
R 8 3

OutbreakTools
Basic Tools for the Analysis of Disease Outbreaks
R 4 1

apex
Phylogenetic Methods for Multiple Gene Data
R 4 2

adephylo
exploratory analyses for the phylogenetic comparative method
R 3

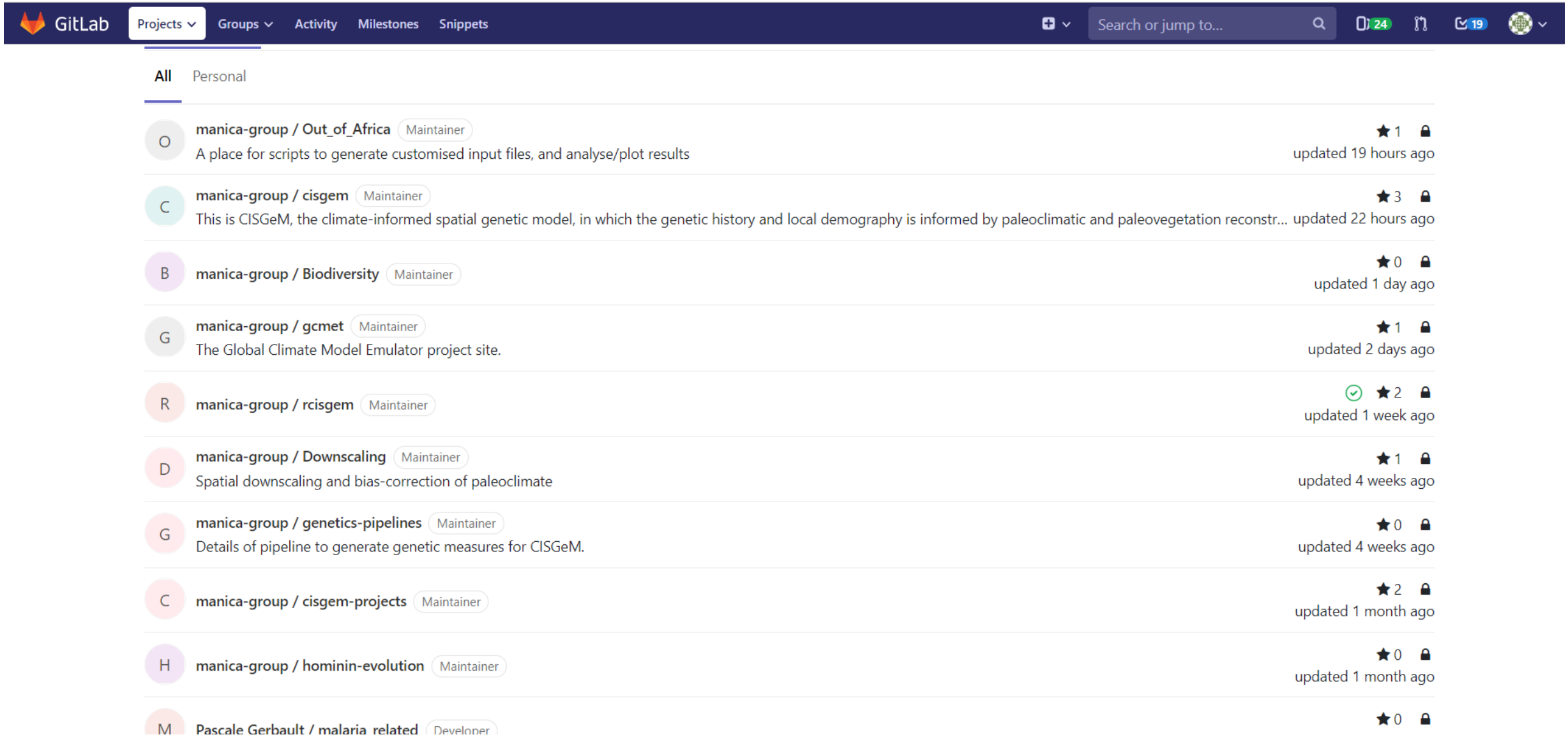
1,198 contributions in the last year

Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct

- Gitlab: Github for companies, university group, enterprises

    - More interaction between users;
    - You can create different projects;
    - You can submit issues and assign them to different lab members;
    - You can use as lab notebook;
    - You can set up mile stones;
    - ….

Github and Gitlab



Github and Gitlab

GitLab | Projects | Groups | Activity | Milestones | Snippets | Search or jump to... | 24 | 19

C cisgem

- Project
- Repository
- Issues 16
  - List
  - Board
  - Labels
  - Milestones
- Merge Requests 0
- CI / CD
- Operations
- Registry
- Wiki
- Snippets

Collapse sidebar

Search or filter results... | Created date

Pi and D outputs in statistics
#59 · opened 1 month ago by Robert Beyer
0 updated 1 month ago

Hunter-gatherer carrying capacity
#58 · opened 3 months ago by Robert Beyer
4 updated 4 days ago

cell distance is hardcoaded, it shoudl be a paramater from the environment file
#57 · opened 4 months ago by Andrea Manica New feature
1 updated 4 months ago

Add MD5 checksum for input files to output streams
#52 · opened 4 months ago by Mario Krapp Version 1.0 New feature
1 updated 3 months ago

rename variable names for consistency with input variables
#50 · opened 4 months ago by Mario Krapp
2 updated 4 months ago

Add Continuous Integration
#42 · opened 5 months ago by Mario Krapp
1 updated 5 months ago

Add parameter values + input file names to output files (demography_output.nc, genealogy_trees.txt)
#40 · opened 5 months ago by Robert Beyer
2 updated 3 months ago

adding extra information to genetics nc file
#38 · opened 5 months ago by Eppie Jones New feature
0 updated 5 months ago

Find optimal number for genealogy trees to decrease overall runtime of MC sweep
#37 · opened 5 months ago by Mario Krapp New feature
0 updated 5 months ago

- Github and gitlab…why???

- It helps to promote reproducible science;
- More transparent and clear specifically for data analysis;
- Working together (with other users) helps to improve and grow faster;

Conclusions:

- R is a flexible language for data analysis, summary, visualisation and modelling;
- R community is vast, lots of support and developers (R package);
- Sharing code, science reproducibility and help the scientific community to grow faster;