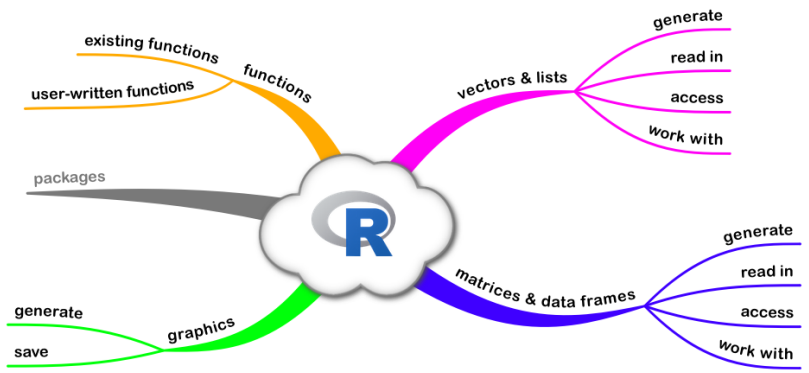
Bioinformatik

Stefanie Hartmann

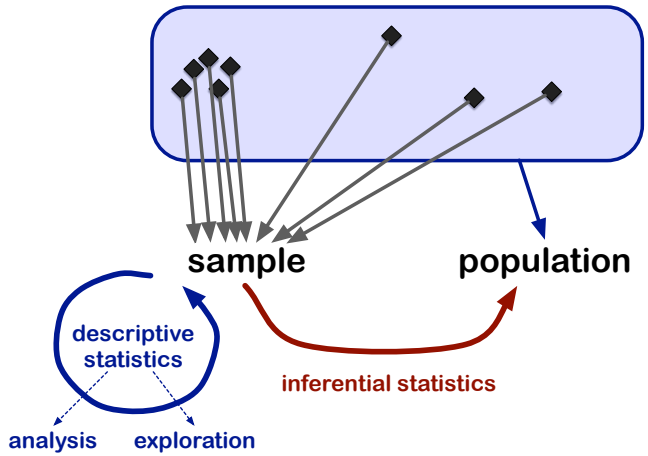
Wintersemester 2019 / 2020, Universität Potsdam

Introduction to R
Jan 17, 2020

Overview



Sample vs. population



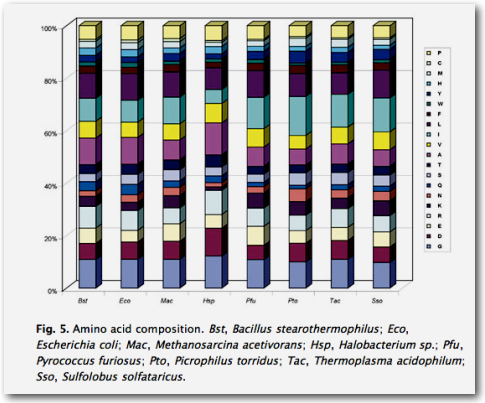
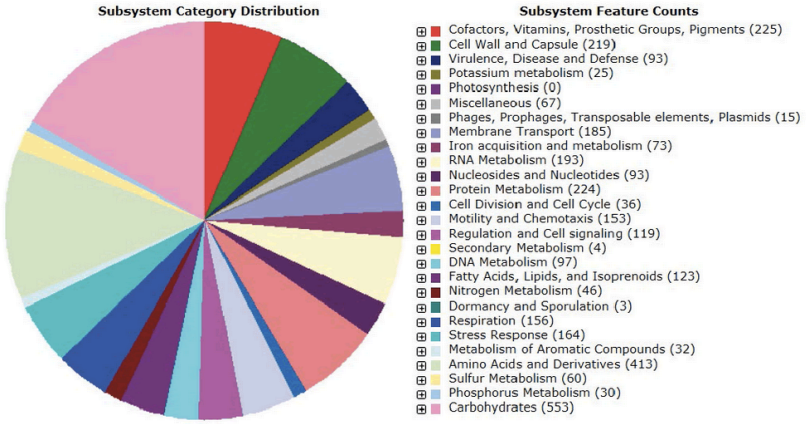
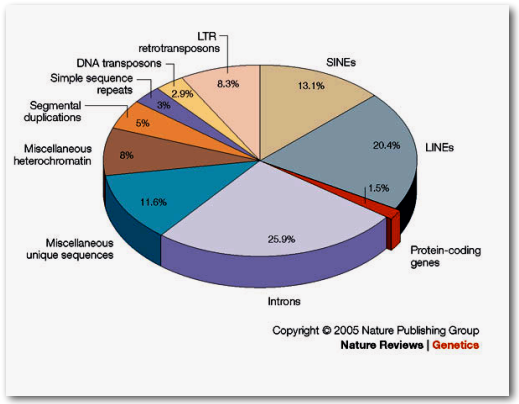


Fig. 5. Amino acid composition. *Bst*, *Bacillus stearothermophilus*; *Eco*, *Escherichia coli*; *Mac*, *Methanosarcina acetivorans*; *Hsp*, *Halobacterium sp.*; *Pfu*, *Pyrococcus furiosus*; *Pto*, *Picrophilus torridus*; *Tac*, *Thermoplasma acidophilum*; *Sso*, *Sulfolobus solfataricus*.

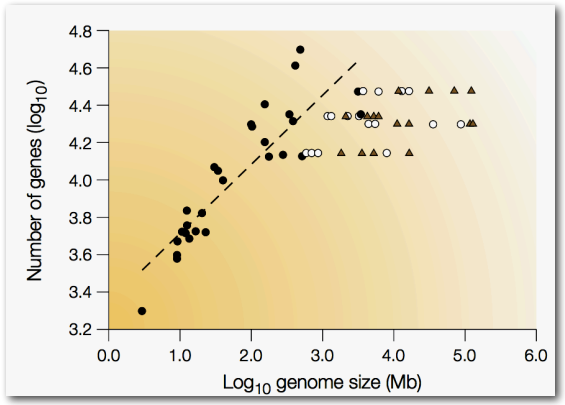
Amino acid composition, bacterial genes



Functional categories of genes,
heavy metal resistant bacterium *Enterobacter cloacae*

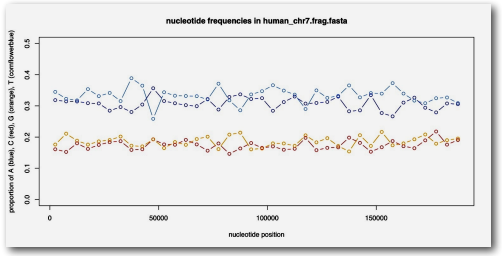


The main components of eukaryotic genomes

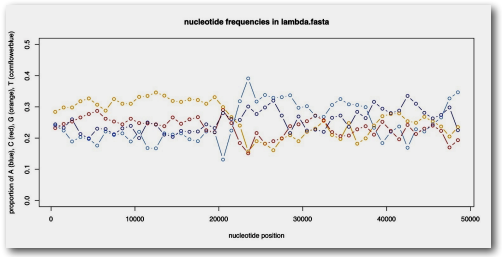


Genome size vs gene number

GC content: human chromosome 7



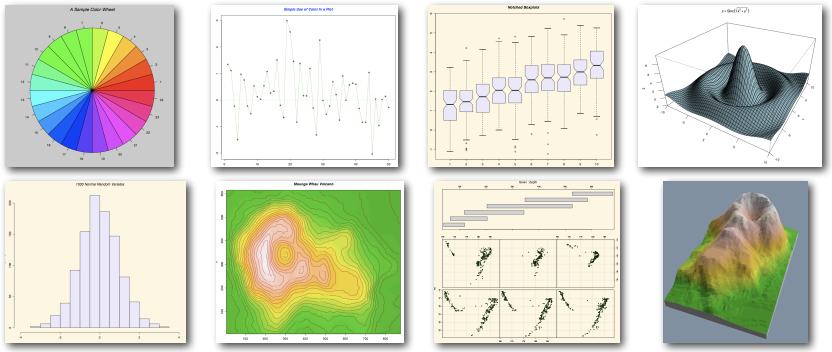
GC content: bacteriophage lambda



What is R?

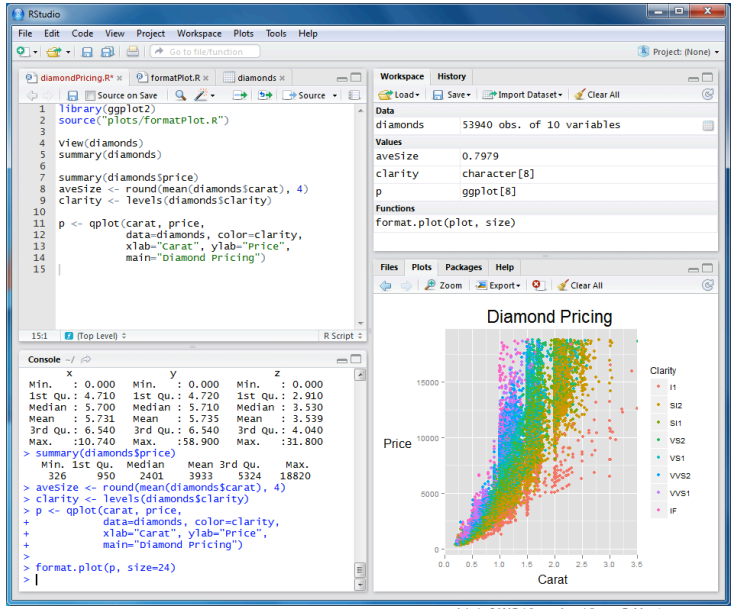
- R is a language and software package with
  - built-in statistical analysis functions
  - excellent graphing capabilities

demo(graphics)



What is R?

- R is a language and software package with
  - built-in statistical analysis functions
  - excellent graphing capabilities
- functions can be added (written) by the user
- additional modules (packages) are available for a variety of specific purposes
- R is available for free at http://www.r-project.org
  - links to books, pdf manuals, and tutorials are also available at this url
  - http://www.math.csi.cuny.edu/Statistics/R/simpleR/printable/simpleR.pdf
  - http://cran.r-project.org/doc/contrib/refcard.pdf



Starting and quitting R

to start, type: R

```
R version 3.3.2 (2016-10-31)
Copyright (C) 2016 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Starting and quitting R

to start, type: R

- interaction through the command line interface
- the ">" is the R prompt
- commands can be executed directly

```
> 2+6
[1] 8
```

- the "+" is the R prompt when continuation is expected

```
> 1*2*3*4*
+ 5*6
[1] 720
```

to quit, type: q()

Data: entered directly or saved in variables

use R like an overgrown calculator

```
> 2+6
[1] 8
```

assign values to variables (objects) and use these

```
> x = 2
> y = 6
> x+y
[1] 8
```

```
> x <- 2
> y <- 6
```

variables are case sensitive, should not begin with numbers or symbols, and should not contain blank spaces

Existing variables

```
> ls()
[1] "x" "y"

> rm(x)

> ls()
[1] "y"
```


Variables: one-dimensional objects

vector

```
> a = 7
> b = c(1, 2, 3, 4, 5)
> c = c("one", "two", "three")
```

list

```
> d = c(1, "two", 3, "four", 5)
```

if the object already exists, its previous value is erased!

Generating vectors

1. read from an external file

2. type into the R interface

```
> a = c(2, 3, 4, 5, 6, 7)          > b = (2:7)
[1] 2 3 4 5 6 7

> c = c(rep(1:4,2))
[1] 1 2 3 4 1 2 3 4

> d = c(rep(1:4, each=2, times=3))
[1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```


Using vectors

```
> x = 2
> y = 6

> x+y              [1] 8
> x*y              [1] 12
> x+3              [1] 5
> factorial(y)     [1] 720
```


Using vectors

```
> x = 2      > x = c(1,3,5)
> y = 6      > y = c(2,4,6)

> x+y              [1] 8        [1]  3  7 11
> x*y              [1] 12       [1]  2 12 30
> x+3              [1] 5        [1]  4 6 8
> factorial(y)     [1] 720      [1]  2  24 720
```


Accessing vectors

```
> y = c(2, 4, 8, 16, 32, 64)
        1  2  3   4   5   6
```

- **vector content**
- **vector indices**
- **general information about the vector**

Vector content

```
> y                 [1] 2 4 8 16 32 64
> y[2]                  1 2 3  4  5  6
> y[-2]
> y[1:3]
> y[c(1,3,5)]
> y[y>10]
> y[y<8 | y>16]
> max(y)
```


Vector content vs. indices

```
> y[y>5]              [1] 8 16 32 64      (value)
> which(y>5)          [1] 3 4 5 6         (index)

> max(y)              [1] 64              (value)
> which(y==max(y))    [1] 6               (index)

> y[y<10]             [1] 2  4  8         (value)
> y<10                [1]  TRUE  TRUE  TRUE
                          FALSE FALSE FALSE (index)
```


About the vector

```
> length(y) [1] 6

> class(y)  [1] "numeric"
```


Two-dimensional variables

1	2	45
1	5	67
2	1	46

matrix

4	“small”	6
8	“med”	3
2	“large”	7

data frame

- **generating matrices and data frames**
- **reading in**
- **accessing**
- **working with**

Two-dimensional variables

	[,1]	[,2]	[,3]
[1,]	1	2	45
[2,]	1	5	67
[3,]	2	1	46

```
y[row,column]
```


Two-dimensional variables

	[,1]	[,2]	[,3]
[1,]	1	2	45
[2,]	1	5	67
[3,]	2	1	46

```
y[3,2]
y[,2]
y[3,]

y[,2]*10

sum(y)
sum(y[,2])
```


Built-in functions

log10(x)

- **gives logs to the base of 10**
- **expects one argument**

```
> log10(5)
[1] 0.69897
```

log(x,n)

- **gives logs to the base n**
- **expects two arguments**

```
> log(81,3)
[1] 4
```

```
> help(log)
> ?log
```


Mathematical functions

log(x) **log to the base e of x**

```
> log(5)
[1] 1.609438
```

$e^? = 5$

```
> number = 5
> log(number)
[1] 1.609438

> numbers = c(2, 5, 10)
> log(numbers)
[1] 0.6931472 1.6094379 2.3025851
```

Mathematical functions

`log(x)`	log to the base e of x
`exp(x)`	e^x
`log(x,n)`	log to base n of x
`log10(x)`	log to base 10 of x
`sqrt(x)`	square root of x
`factorial(x)`	x!
`choose(n,x)`	binomial coefficients n!/(x! (n-x)!)
`cos(x)`	cosine of x (in radians)
...	

Vector functions

```
> numbers = c(1, 10, 5, 30)
> mean(numbers)                > m = mean(numbers)
[1] 11.5

> sort(numbers)
[1]  1  5 10 30                (values)

> rank(numbers)
[1] 1 3 2 4                    (indices)
```


Vector functions

`length(x)`	number of elements in x
`max(x)`	maximum value in x
`min(x)`	minimum value in x
`sum(x)`	total of all the values in x
`mean(x)`	arithmetic average of all values in x
`median(x)`	median value in x
`var(x)`	variance of x
`sort(x)`	a sorted version of x
`quantile(x)`	vector containing the min., lower quartile, median, upper quartile, max. of x

Other built-in functions

string functions
- strsplit(x, split); toupper(x)

statistical probability functions
- dnorm(x); dunif(x, min=0, max=1)

statistical analysis functions
- chisq.test(matrix)

graphics functions
- plot(x,y); boxplot(z)

Reading data into R

through the keyboard: using the concatenate or matrix function

```
> x = c(3, 6, 8, 1, 6, 8)
> x = matrix(c(1, 2, 3, 4, 5, 6),  byrow=T, nrow=2)
```

through the keyboard or from the clipboard: using the scan function

```
> y = scan()
1: 4
2: 5
```

from an external file using the read.table function

- **reads in a file in table format, creates a dataframe from it**

```
> z = read.table("/home/bioinf/genomes.txt")
```


Reading data into R

```
> z = read.table("/home/bioinf/genomes.txt", header=T)
```

```
> z[,3]
```

header

rows

organism	type	genomeSizeMB	geneNumber
Arabidopsis	plant	125	25,000
yeast	unicellular	12	6,200
human	animal	3,000	24,000
rice	plant	466	60,000
worm	animal	100	21,000

Reading data into R

```
> z = read.table("/home/bioinf/genomes.txt", header=T)
> attach(z)

> z[,3]
> genomeSizeMB
> z$genomeSizeMB
```

header

rows

organism	type	genomeSizeMB	geneNumber
Arabidopsis	plant	125	25,000
yeast	unicellular	12	6,200
human	animal	3,000	24,000
rice	plant	466	60,000
worm	animal	100	21,000

Using row names

```
> z = read.table("/home/bioinf/genomes.txt", header=T, row.names=1)
```

row names

header

rows

organism	type	genomeSizeMB	geneNumber
Arabidopsis	plant	125	25,000
yeast	unicellular	12	6,200
human	animal	3,000	24,000
rice	plant	466	60,000
worm	animal	100	21,000

Generating graphics

- different plots for different data/purposes!
  - plots for a single variable
  - plots with two variables
  - multivariate plots
  - special plots
- general introduction today
- more graphics next week!

xv	ys
90.77212	51.75918
16.11536	28.95312
31.12350	35.50002
39.79581	32.69104
48.82297	40.50366
78.17519	56.58430
29.97319	31.87137
61.66741	46.30729
55.47211	44.09255
8.93557	30.02324
73.36877	46.05974
60.94679	42.96545
30.12837	41.13832
63.49312	41.89534
69.42570	45.94758
33.65231	30.78103
60.49952	45.73121
89.56325	47.64866
37.79412	36.71042
55.99477	41.87292
0.14132	26.84904
71.25819	49.54433
1.22322	27.91121
81.51914	45.52460
3.87410	23.16908
73.32387	49.90860
45.66414	40.22269
67.81447	45.13829
...	...

Saving graphics

an output format in R is called a “device”: we have to decide on a device before writing into it

1. save an image that has already been generated: copy the content of that window into a file using `dev.copy` command.

```
> plot(mydata)
> dev.copy(device = pdf, file = "plot_of_mydata.pdf")
> dev.off()
```

2. first create the device and then generate the graphics, writing these directly into the device.

```
> pdf(file = "plot_of_mydata.pdf")
> plot(mydata)
> dev.off()
```


Today’s exercise

- starting R
- data formats
  - vectors
  - matrices
- functions
  - optional: writing your own function
- generating graphics
- saving graphics

➡ please finish this exercise before the next lecture

Terms and concepts

- assigning and using variables
  - concatenate function
  - read.table function
  - accessing data in vectors and matrices
  - accessing content vs. indices