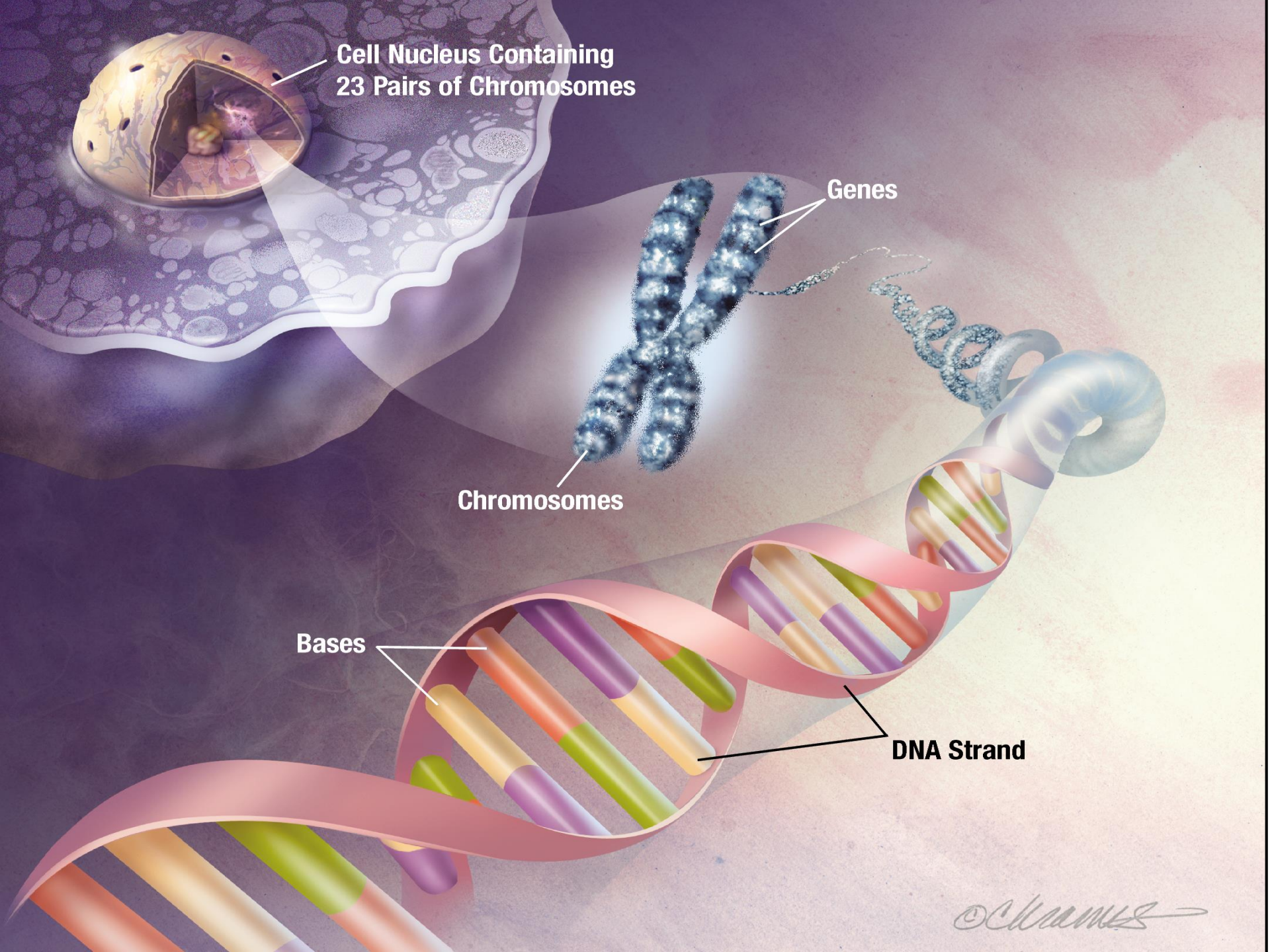


Lecture 1

Introduction to bioinformatics (MVE510)

Autumn, 2020

Bioinformatics



Cell Nucleus Containing
23 Pairs of Chromosomes

Genes

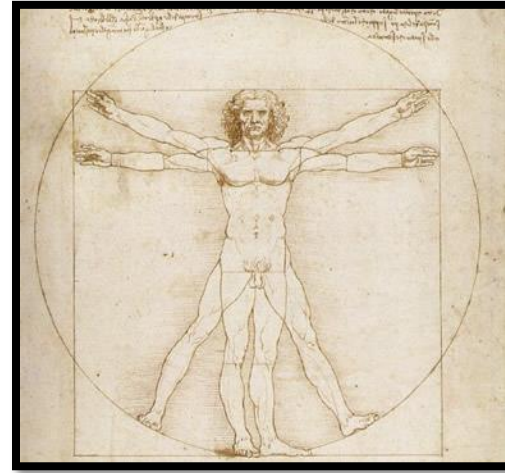
Chromosomes

Bases

DNA Strand

©Chumms

The genome contains vast amount of information



- Genome size (in bases: A,C,G or T)
 - Bacteria: 5 million bases
 - Humans: 3,2 billion bases.
 - Amoeba: 670 billion bases

Example: Genome sequencing

- Sequencing of the human genome results in 3 billion observations
- The genome of half of the Icelandic population is under sequencing. This will result in 500 trillion (5×10^{14}) observations. The aim is to find patterns that explain heritable diseases.

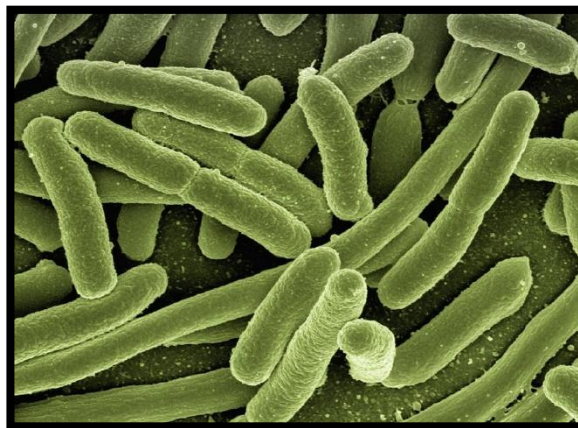


Example: Transcriptome sequencing

- Cancer patients are today analyzed by transcriptome sequencing at several hospitals.
- For each patient, the 50 million nucleotides of the 'exome' (i.e. the part of the genome that corresponds to the genes) is sequenced.
- In Sweden alone, more than 60,000 people are diagnosed with cancer each year.

Example: Sequencing of bacteria

- There are 10^{14} bacterial cells in the human gut carrying more than 3 million genes and 10^{20} nucleotides
- Changes in the type and the genomes of these bacteria is associated with diseases, such as diabetes.



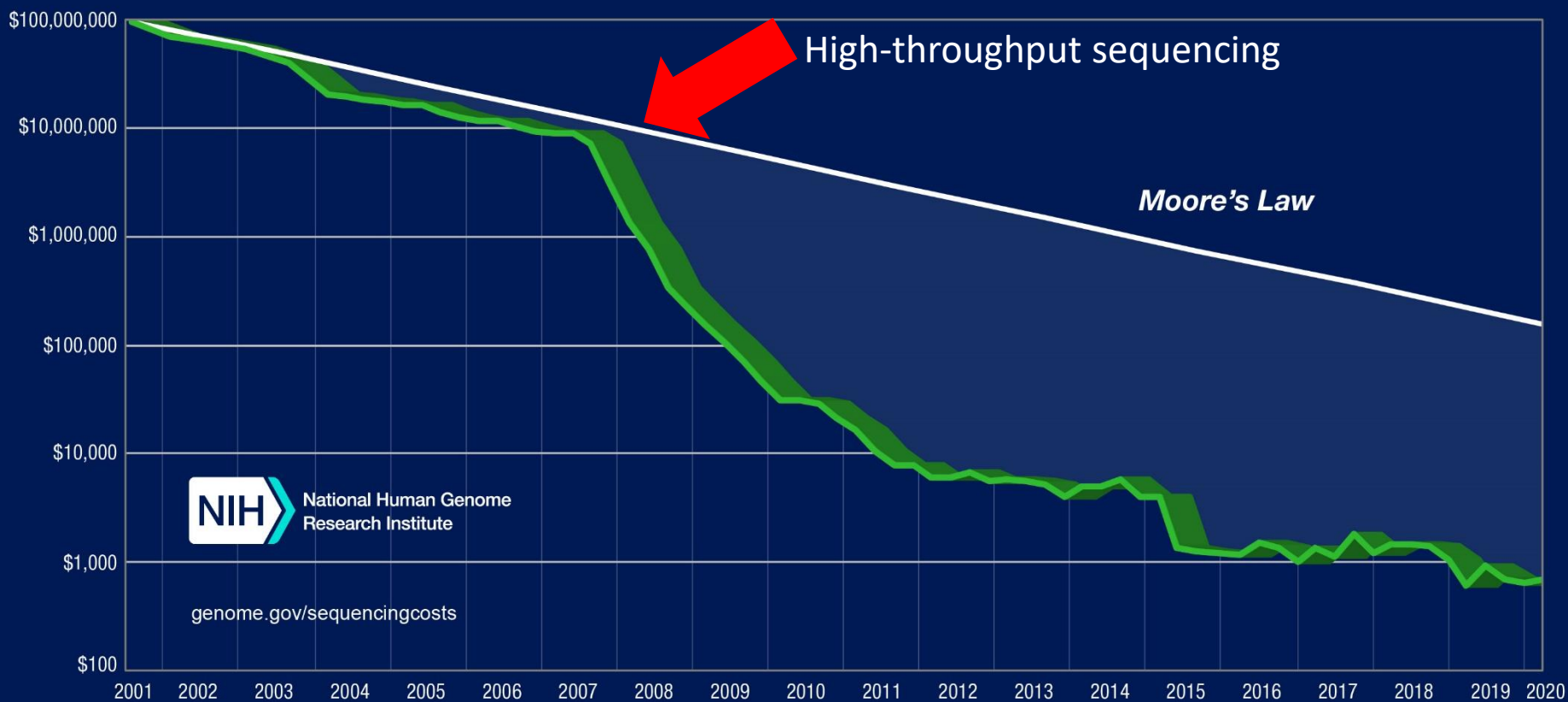
There are also a lot of bacteria in the environment!



1 gram of soil

- 100 million bacteria
- DNA: >100 terabases (10^{14})

Cost per Human Genome



Bioinformatics

Bioinformatics is the analysis and interpretation of molecular data

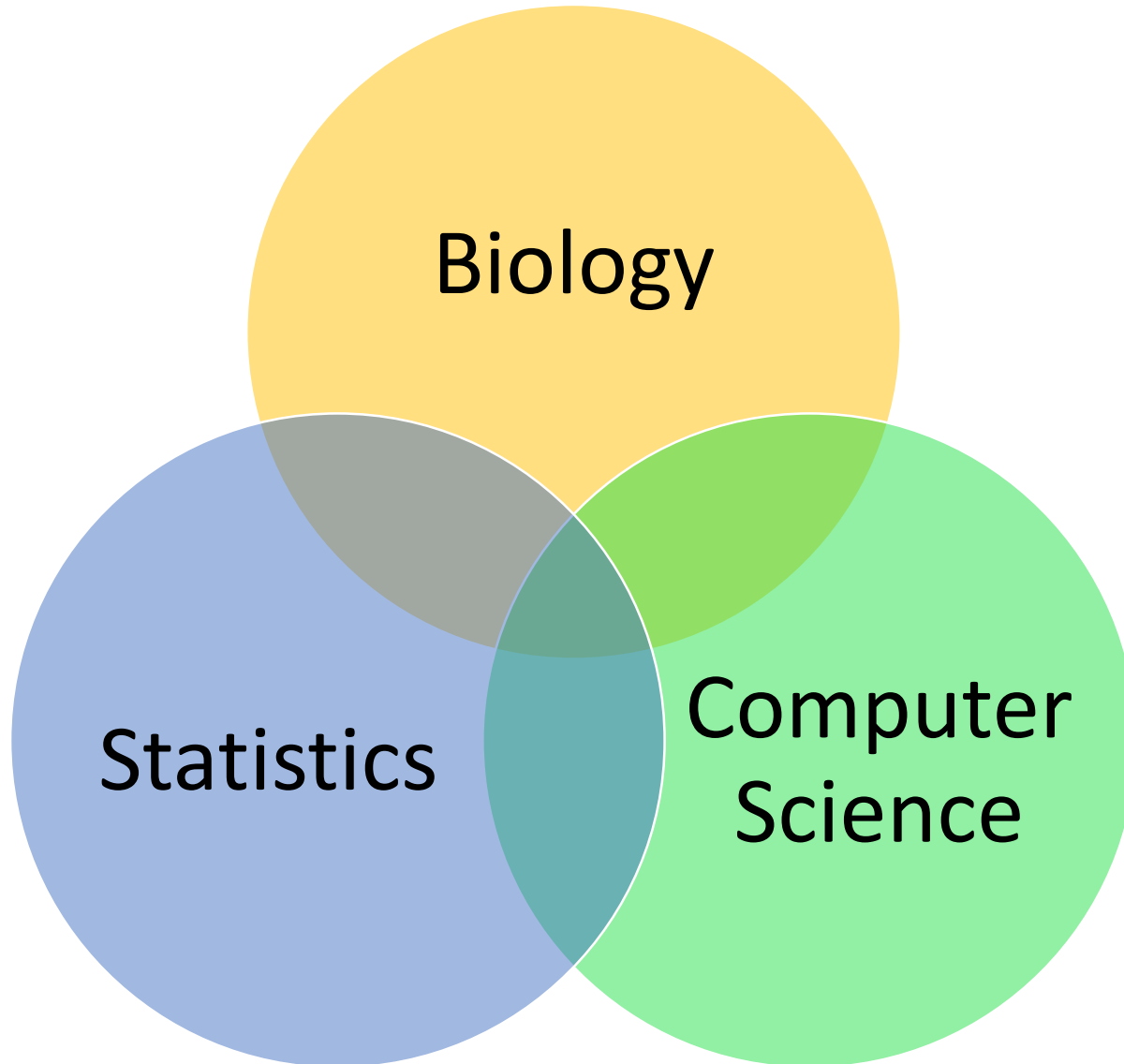
- How should data be generated?
- How should data be analyzed and interpreted?
- What biological conclusions can I draw from the data?

Important aims of this course

This course is about understanding complex biological data with focus on DNA sequencing.

- How can biological questions be addressed and solved using high-throughput molecular data?
- What are the advantages and disadvantages of different technologies for high-throughput DNA sequencing?
- How should high-throughput DNA sequencing data be analyzed and visualized?
- What are the challenges and limitations of the interpretation of high-throughput data?

Bioinformatics is interdisciplinary



This course is problem-oriented!

This means that we will

- Focus on biological problems and use high-throughput data to solve them
- Learn methods and algorithms that are necessary to solve the problems
- Emphasis on biological interpretation. What can we actually say about the results?

Course organization

- Lectures over Zoom
 - 14, 2x45 minute lectures. Two extra 5 min breaks.
 - Two lectures per week (Tuesday and Fridays)
- Computer exercises of Zoom
 - 4 compulsory exercises
 - Scheduled exercises on Tuesdays (2x45 min) and Wednesday (4x45 min).
 - Two assistant available for any forms of questions
 - Use these sessions to ask questions!

Course organization – lectures

- Next generation DNA sequencing
- Quality assessment of DNA sequence data
- DNA sequence alignment and mapping
- Supervised analysis of high-dimensional data: linear models
- Unsupervised analysis of high-dimensional data: clustering and principal component analysis
- Applications to
 - Genome sequencing
 - Transcriptome sequencing (RNA-seq)
 - Metagenomics

Course organization – exercises

Four computer exercises

The exercises requires you to do bioinformatics and analyze real data. The exercises are the core of the course.

1. Introduction to R
2. Genome (re)sequencing with applications to clinical microbiology
3. Transcriptome sequencing (RNA-seq) with application to human medicine
4. Metagenomics with application to environmental sciences

Guest lectures....

To be decided!

Last year we had



Diagnostics

Course organization - examination

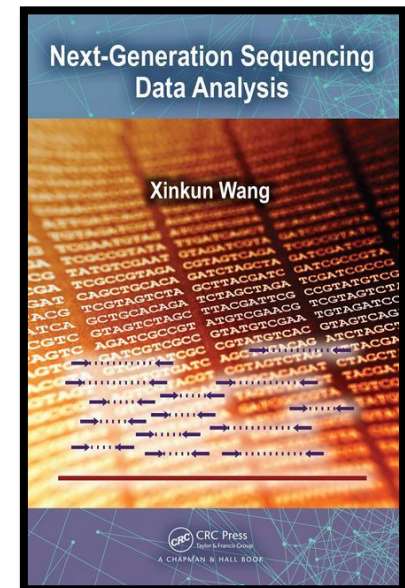
The course is examined in two ways

1. Written exam, 14 January 2021
2. Approved reports from computer exercises.
Exercise 1 is examined during the computer exercise. Exercise 2-4 requires a report that is handed in through the course homepage.

Course literature

- Lecture notes
- Selected research papers
- Xinkun Wang, *Next-Generation Sequencing Data Analysis*, CRC Press, ISBN:9781482217889. Available at Cremona. Summarizes the course.

See the course homepage for full information.



Course organization - teachers

Erik Kristiansson

erik.kristiansson@chalmers.se

Examiner, course administrator and lecturer

Juan Salvador Inda Diaz

inda@chalmers.se

Assistant

David Lund

dlund@chalmers.se

Assistant

Course organization – important links

The course homepage is available at Canvas

<https://chalmers.instructure.com/courses/10879>

The course homepage contains

- All necessary reading material
- The compulsory exercises
- Zoom-links to all online sessions
- Recordings of all lectures

The syllabus is available at (please read it!):

https://www.student.chalmers.se/sp/course?course_id=30425

Course evaluation

The following persons has been randomly selected to be student representatives

Leo Benson

Edwin Eliasson

Louise Stauber Näslund

Hannah Steinhausen

Chattarin Wangwittaya

Extra meeting already after two weeks.

Introduction to



What is R?

- R is a statistical programming language for data analysis. All exercises in this course will be done in R.
- R contains many methods for statistical analysis, data interpretation and visualization.
- R is open source and continuously developed by the research community.
- R is free available at <https://www.r-project.org/>

Why do you need to learn R?

- R is the standard environment for analysis of biological and high-dimensional data.
- R has hundreds of different 'R packages' specialized for molecular biology containing the state-of-the-art methods.
- Learning more languages improve your overall programming skills.
- In this course you learn R and are expected to independently work with R to solve biological problems.

Is there a difference between R and MATLAB?

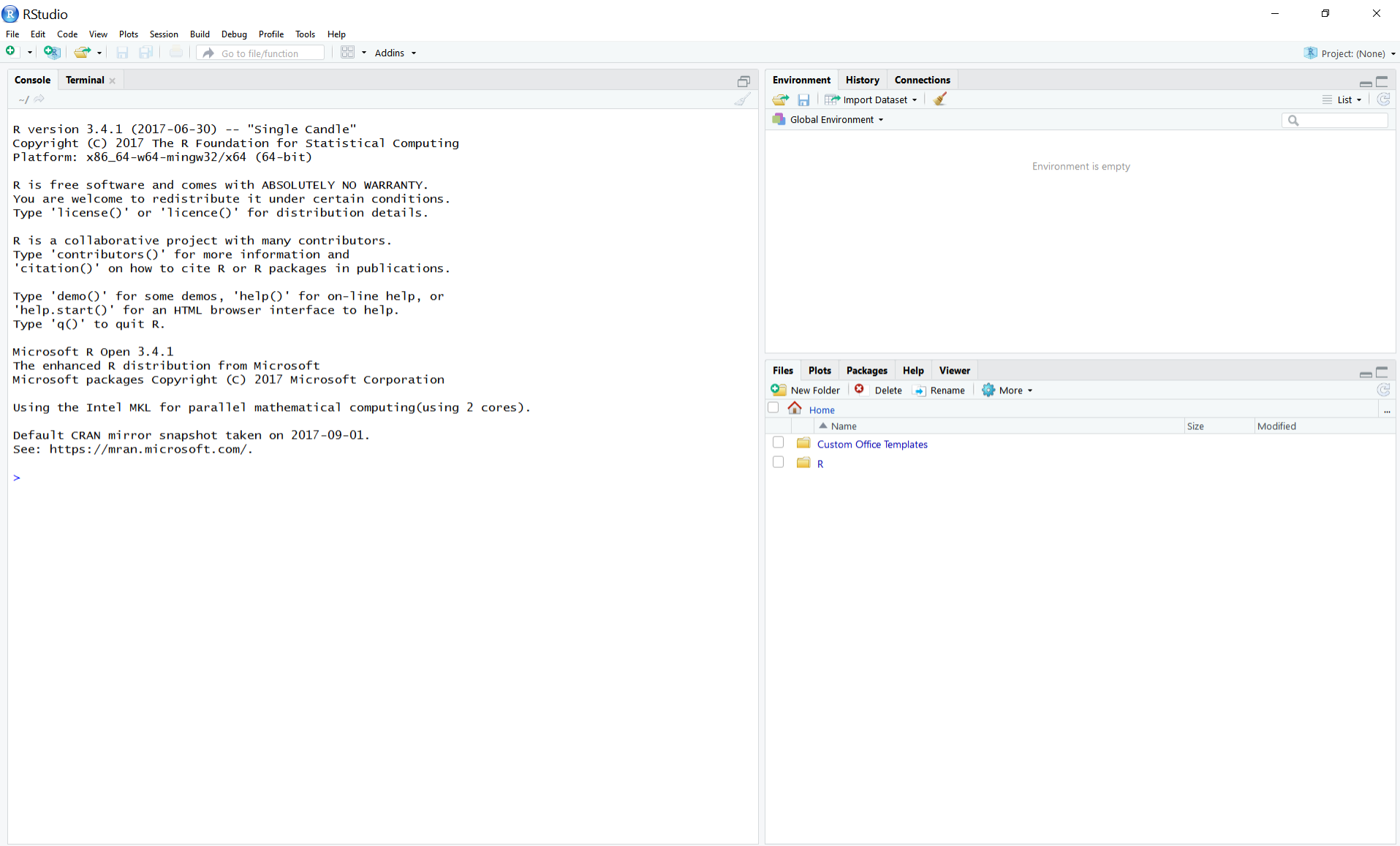
Yes!

- R uses a different syntax and different functions and commands
- R is slightly less user friendly but offers a much larger set of tools for data analysis, interpretation and visualization

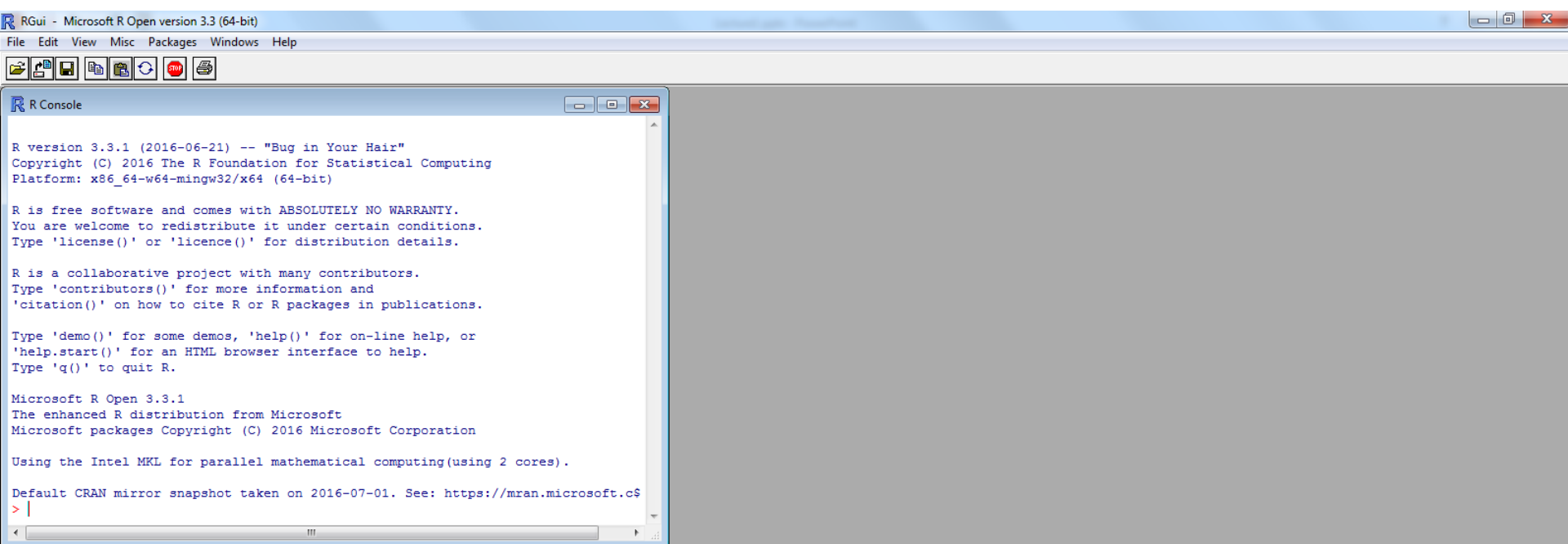
But....

- The concept is the same! You write programs in scripts, that are run in an 'interpreter'.
- Many of the commands will be similar to MATLAB.

The interface of Rstudio



The interface of R



R has a nice help function

- R contains help information about every command!
- The help is informative but sometime a bit overwhelming. Always take your time when reading the R help.

The help function is activated using either 'help' or '?', that is

```
> help(sum)
```

or

```
> ?sum
```

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

```
## Default S3 method:
```

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

x

An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.

trim

the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.

na.rm

a logical value indicating whether NA values should be stripped before the computation proceeds.

...

further arguments passed to or from other methods.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

Introduction to R: scripts

- R scripts are files with multiple commands that are executed at once. Load a script into R using the **source** command
- R scripts can be written in any text editor or using the build in editor (available under File->New Script)

```
> source("myscript.R")
```

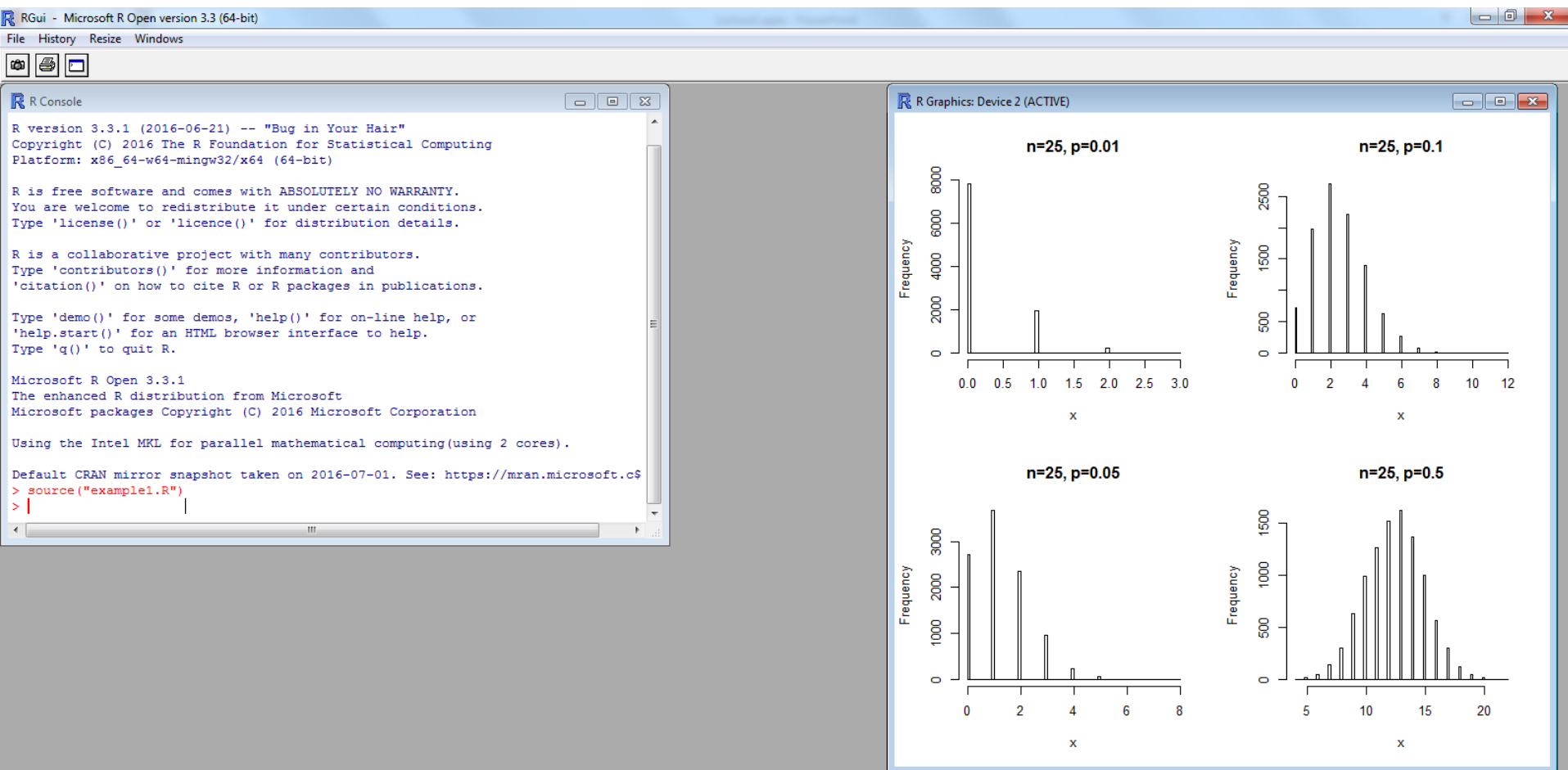
- Note that you need to point R to the directory where you have your scripts.

Example of an R script

Content of 'example1.R'

```
# Vector of probabilities
p=c(0.01, 0.05, 0.1, 0.5)
# Opens a plot window and set the layout to a 2x2 grid
layout(matrix(1:4, nrow=2, ncol=2))
# Loop over the 4 probabilities
for(i in 1:4){
  # Sample 10000 observation from a binomial with parameters
  # n=25 and p according to the probability vector
  x=rbinom(10000, 25, p[i])
  # Create a histogram with no title
  hist(x, breaks=100, main="")
  # Set the title
  title(paste("n=25, p=", p[i], sep=""))
}
```

Example of an R script



Vector and matrices in R

- R is, similarly to MATLAB, very efficient in working with vectors and matrices.

```
> x=c(1,2,3,4,5,6,7,8,9,10,11,12)
```

```
> x
```

```
 1  2  3  4  5  6  7  8  9 10 11 12
```

```
> x[4:7]
```

```
[1] 4 5 6 7
```

```
> 2*x
```

```
 2  4  6  8 10 12 14 16 18 20 22 24
```

```
> exp(x)
```

```
2.718282e+00 7.389056e+00 2.008554e+01 5.459815e+01
```

```
1.484132e+02 4.034288e+02 1.096633e+03 2.980958e+03
```

```
8.103084e+03 2.202647e+04 5.987414e+04 1.627548e+05
```

Vector and matrices in R

- R is, similarly to MATLAB, very efficient in working with vectors and matrices.

```
> y=matrix(x, nrow=3, ncol=4, byrow=T)
```

```
> y
```

| | [,1] | [,2] | [,3] | [,4] |
|------|------|------|------|------|
| [1,] | 1 | 2 | 3 | 4 |
| [2,] | 5 | 6 | 7 | 8 |
| [3,] | 9 | 10 | 11 | 12 |

```
> y[1,]
```

```
1 2 3 4
```

```
> y[,2]
```

```
2 6 10
```


Using plots and devices

- R has a powerful, but slightly complex, system for making plots.
- An R-plot are created using the following steps:
 1. Open a plotting device
 2. Plot the figure
 3. Close the plotting device
- Commonly used devices are **pdf** and **bitmap**.

Using plots and devices

Example 1: Creating plotting in a pdf-file.

```
pdf("myplot.pdf", width=12, height=8)
x=rbinom(10000, 25, 0.5)
hist(x, breaks=100, main="")
dev.off()
```

Example 2: Plotting in a window in R

```
windows()
x=rbinom(10000, 25, 0.5)
hist(x, breaks=100, main="")
# Call dev.off() if you want to close the window.
```

Using plots and devices

- When you use **pdf** and **bitmap**, you need to close the devices before the file is viewable.
- **graphics.off** closes all open devices. This is very useful if your script stops due to an error during the plotting of a figure.
- A good strategy is to first design your figures and then use **pdf** or **bitmap** to plot the ‘final’ figure to a pdf/png-file.
- Note that **pdf** and **bitmap** has many arguments where you can set the type of image, its resolution, etc.

Writing functions

- In R it is very easy to write your own functions

```
myfun=function(x) {  
  average.value=sum(x)/length(x);  
  return(average.value)  
}
```

```
> x=c(1,2,3,4,5,6,7,8,9,10,11,12)  
> x  
 1  2  3  4  5  6  7  8  9 10 11 12  
> myfun(x)  
6.5
```

- Use scripts to define functions!

For-loops in R

- Similarly to other programming languages, R can use for-loops

```
for (i in 1:100) {  
  print(i)  
}
```

1

2

3

- Note that 1:100, which creates a vector with the numbers between 1 and 100 can be exchanged for any vector.

Removing stuff

- The command **ls** lists all defined variables, functions and objects

```
> ls()  
"myfun" "x"      "y"
```

- Old variables can sometimes introduce bugs. To remove a variable you can use the **rm** command.

```
> rm(x)  
> ls()  
"myfun" "y"
```

```
> rm(list=ls()) # This clears everything!  
> ls()  
character(0)
```

Tips for learning R

- Hands on experience is very important! You will not learn programming by reading texts.
- Read the R help!
- If that does not help, use Google to search for a solution!
- If that does not work, ask for help. We have two very competent assistants.
- Be persistent and a bit stubborn – once you are ‘over the top’ things will become much easier.

Introduction to computer exercise 1

- The aim of this exercise is to introduce you to R
- Covers many of the basic elements of R
 - Fundamentals
 - Vector and matrix manipulations
 - Writing scripts
 - Plotting figures
 - Writing functions
- Examination at the end of the lecture
- Use this opportunity to learn R – you will need it later in the course.