BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Introduction to ImageJ
Session 5: Macro scripting, automation and data mining with R

Dimitri Vanhecke

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

adolphe merkle institute
excellence in pure and applied nanoscience

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

EPFL
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

UNIVERSITÉ DE GENÈVE

FNSNF
SWISS NATIONAL SCIENCE FOUNDATION

Scripting: What? Why? How?

What is an ImageJ script?

A script is a recipe
= an algorithm that automates a series of (repetitive) ImageJ commands.
In essence: a text file with a sequence of commands.

Why writing scripts/Macros?
Scripts make your life easier

Plugins
- = programming
- Require java programming knowledge
- infinite possibilities
- Library compatible (once a class is written, it can be exchanged/used anywhere)

Macros
- easier: automation of tasks... And documentation,
- not so easy to exchange because of missing libraries
- Much easier and lighter (scripting vs programming)
- Relatively slow (~40x slower than plugins)
- Limited real-time interaction: a macro is a train that rolls, and rolls, and rolls...
- No extendability (no use in other software)

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Learning Imagej macro scripting

Advantages of ImageJ macros over other languages/plugins:
- Easy to learn since commands are mirrors of the GUI functions
- No need to understand java, Python, C++, …

How to write ImageJ macros
- Fiji has a IDE (integrated development environment): editor with syntax highlighter, command auto-completion, ...
- ImageJ has not such features

Both Fiji and ImageJ can *record* a macro = recording the sequence of clicks in ImageJ

Recording a macro

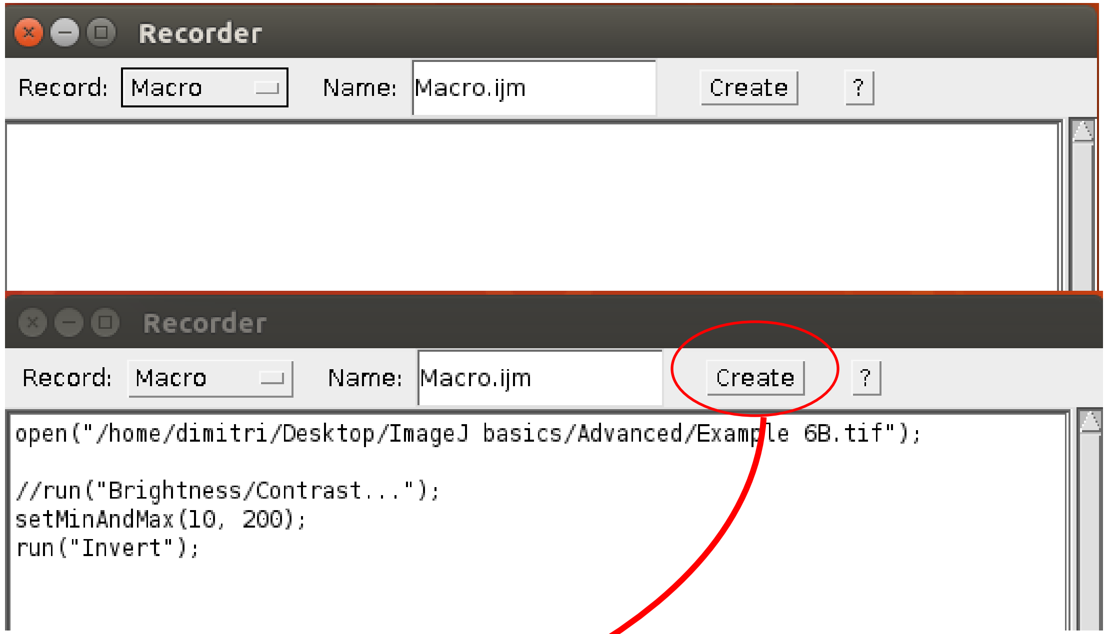
EXERCISE
Use the macro recorder

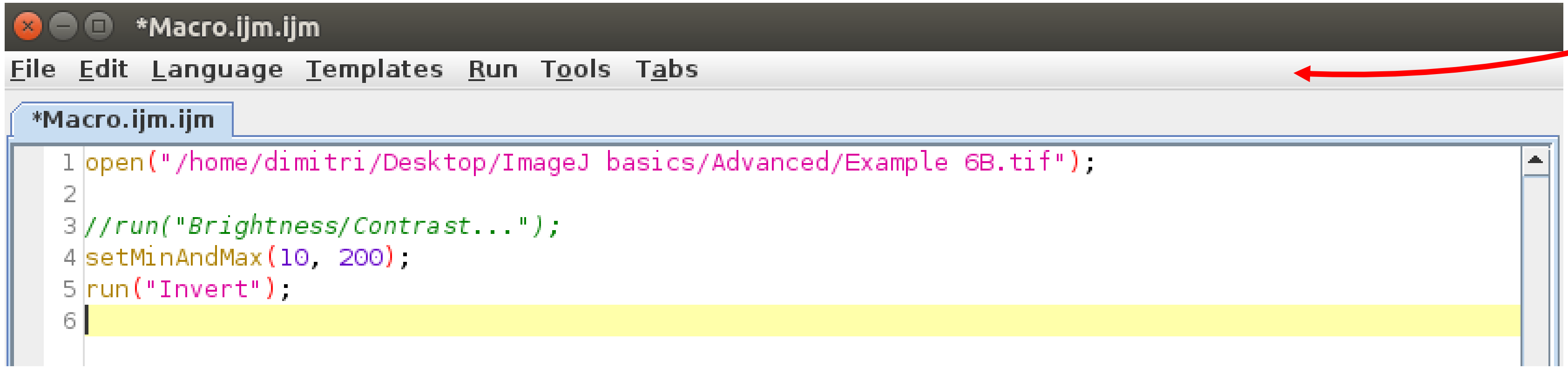
Plugins > macro > record...

Open example 1 (A, lena or B, Fabio)
Adjust brightness and contrast (use 'set' in Contrast & Brightness)
Invert the image (Edit > Invert)

In the Recorder: click «Create»

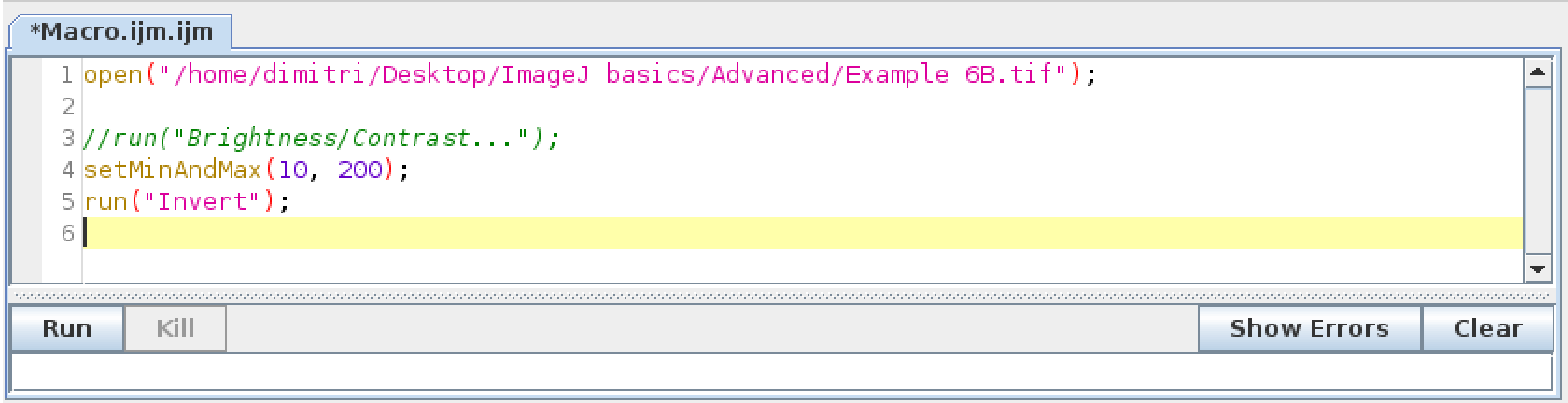
Close all images. In the created macro window: click «run»





Recording a macro

EXERCISE
See how the macro recorder works



Close all images.

Macro.ijm > click run

Automatically opens example 1 (lena or Fabio)
Automatically adjusts brightness and contrast
Inverts the image

The recorder will:

- Turn graphical commands (GUI) to code
- Keep track of what you do to your image (=log)

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

The print command

1. Open a new script window
In Fiji: File > New > Script... A new window opens, named «New_»

2. Set the language
In the Editor: Language > IJ1 macro (note: in the recorder, the macro language was set as default)

3. Write a line of code, e.g. :

```
print("hello world");
```

! Do not ignore lowercase (print is not the same as Print or PRINT)
! watch the quotation marks ‘’ ‘’
! and the semicolon at the end (=end of command)... They are all important

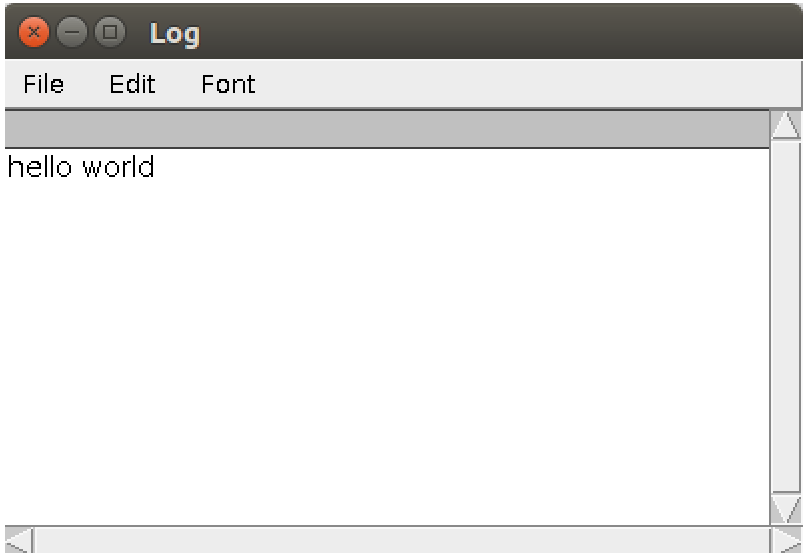
4. **Automatically, the syntax highlighter** will:
- Put «print» in dark yellow: it is recognized as a valid command
- (‘’hello world’’) in pink: it is recognized as text
- If this does not happen, you forgot point 2, setting the language

Now click «run»

The print command

EXERCISE
The “Hello world” script

Result:

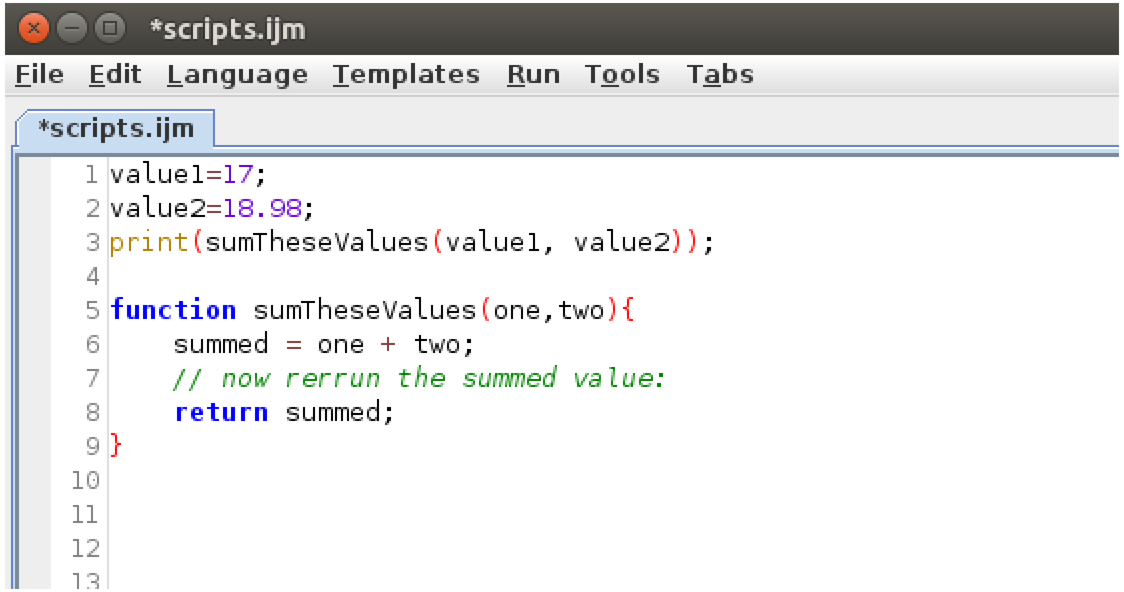


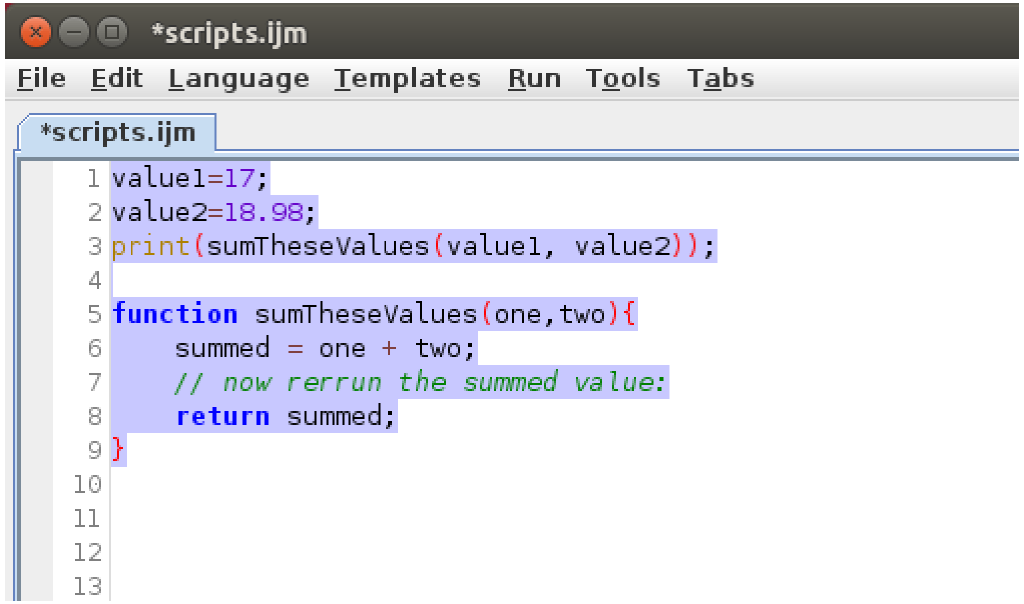
Script.ijm: Line 1-12

Note: you can also use other words than «Hello world» ☺

Running / debugging the script

Find the prepared script.ijm on the website





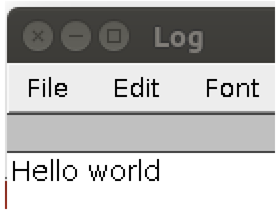
To run the entire script (everything), click «run» (**don't do this now!!**)
To run part of the script, select the lines you want to include and hit CTRL+SHIFT+R (or Run > Run selected code)

Variables and strings

EXERCISE
Combining text variables

In the Editor:
Write:

```
text1 = "hello ";
text2 = "world";
print (text1 + text2);
```



Variables are **typeless** (no need to declare integers, bytes, ...)
Variables can contain
- letters and text (=string)
- a number (integer, double, float)
- a boolean (true/false)
- an array (= a list of variables)

Case sensitive! (the variable AMI is not the same as Ami)

Using variables

The statements in the commands can be fitted with variables. In short, the options must be a text form that is understood by the command.
e.g.

```
scaling = 0.5;
run("Scale...", "x=" +scaling +" y="+scaling + " interpolation=Bilinear create title=Fabio-scaled");
```

Note the difference between text (=string, between “ ”) and variables

- Text must always be between “ ”
- Variables not (try it)
- Plus (+) strings numbers and text together
- The sentence is read from left to right (you can perform calculation on the variables during concatenation)

Alternatively:

```
scaling = 0.5;
options = "x=" +scaling +" y="+scaling + " interpolation=Bilinear create title=Fabio-scaled";
run("Scale...", options);
```

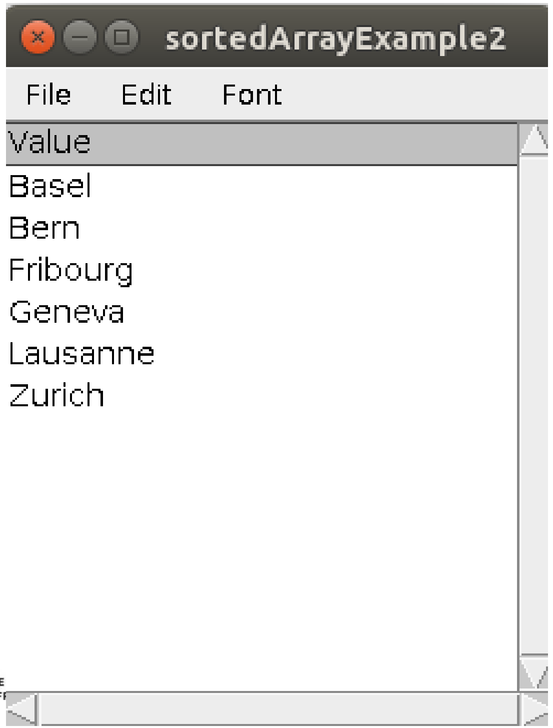
Arrays

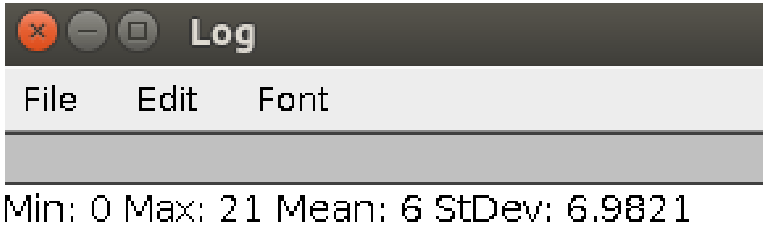
An array is a list of variables. It can contain numbers, strings (text) or both.

Arrays have powerful methods:

- Find min, max, mean, mode, median, & maxima/minima (not just max/min!) in a numerical array
- Sorting (alphabetically)
- Find fourier amplitudes
- Rank positions
- Get Vertex positions (assuming the array describes positions on a closed contour)

```
arrayExample = newArray(0,1,1,2,3,5,8,13,21);
Array.getStatistics(arrayExample, min, max, mean, stdDev)
print("Min: " + min + " Max: " + max + " Mean: " + mean +" StDev: " + stdDev);
arrayExample2 = newArray("Bern", "Fribourg", "Lausanne", "Geneva", "Zurich", "Basel");
sortedArrayExample2 = Array.sort(arrayExample2);
Array.show(sortedArrayExample2);
```





Script.ijm: Line 27-37

UNI FR UNIVERSITÉ DE UNIVERSITÄT

Build-in Functions:

There are many build-in functions: **help > Macro functions ...**

http://rsb.info.nih.gov/ij/developer/macro/functions.html

Built-in Macro Functions

[A][B][C][D][E][F][G][H][I][J][K][L][M]
[N][O][P][Q][R][S][T][U][V][W][X][Y][Z]

Print List

A [Top]

abs(n)
Returns the absolute value of *n*.

acos(n)
Returns the inverse cosine (in radians) of *n*.

Array Functions
These functions operate on arrays. Refer to the ArrayFunctions macro for examples.

Array.concat(array1,array2) - Returns a new array created by joining two or more arrays or values (examples). Requires 1.46c.
Array.copy(array) - Returns a copy of *array*.
Array.fill(array, value) - Assigns the specified numeric value to each element of *array*.
Array.findMaxima(array, tolerance) - Returns an array holding the peak positions (sorted with descending strength). Tolerance is the minimum amplitude difference to needed to separate two peaks. There is an optional 'excludeOnEdges' argument that defaults to 'true'. Examples. Requires 1.48c.

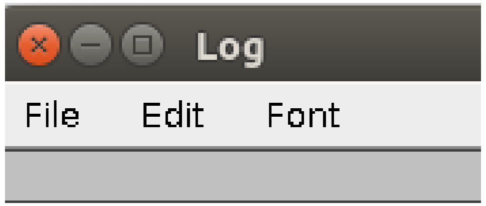
BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Functions

You can also make functions yourself:

```
value1 = 17;
value2 = 18.98;
print(sumTheseValues(value1, value2));


function sumTheseValues(one,two){
    summed = one + two;
    return summed;
}
```



35.980000000000004

1. ‘sumTheseValues’ is called
2. The two variables Value1 and Value2 are send to the function (accepted as variable «one» and «two»).
3. Within the function (defined by {}), some lines can be written
4. The result is returned to the main code

Script.ijm: Line 39-49

Comments

When you read your code again later (or much later), you want to understand what your code does, and why. For this, you can add comments, i.e. text which is ignored by ImageJ when it executes the macro.

Use // in front of the line. ; at the end is not needed.
The entire line will be ignored by the interpreter.
e.g. **Line 46**

For multiline comments (eg **line 43-90**):
Use '/*'. Close the section off by '*/'

```
value1 = 17;
value2 = 18.98;
// this will call a self-made function
print(sumTheseValues(value1, value2));


function sumTheseValues(one,two){
    summed = one + two;
    // now rerrun the summed value:
    return summed;
}
```

```
/*
The Adolphe Merkle Institute staff follow an ethical charter.
They respect ethical standards in their behavior at the workplace and act responsibly in planning and
executing their research and teaching.

At AMI, we:

    respect human rights, protect personal information, prevent any kind of harassment and
    discrimination based on nationality, gender, race, religion, sexual orientation, age
    or any other reason, and act whenever needed to prevent such behavior;

    comply with international rules, applicable laws and regulations, school rules, the spirit
    of such rules, locally accepted societal standards and foster an environment of mutual
    trust and respect;

    teach with integrity, fairness, dignity and honesty and maintain a professional relationship
    with students;

    plan, conduct, document and communicate research and interpret results according to the highest
    international standards;

    limit testing on animals, wherever possible and be committed to applying high standards of
    animal welfare and to using animals responsibly;

    not accept funding from sources that might lead to influencing of research results or their
    interpretation or lead to a conflict of interest;

    not conduct research that has the goal of producing results that can be used to harm humans,
    animals or the environment or can be anticipated of being misused for the latter;

    use resources in a sustainable way and contribute with their research to the same,
    wherever possible.

*/
```

Something useful…

EXERCISE

Try to make a Sobel filter yourself.
Use the recorder!

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

Something useful…

EXERCISE

Try to make a Sobel filter yourself.

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

```
pathtoImage = "C:/Users/vanheckd/Desktop/Scripts/Example 8.tif";
sobelFilter(pathtoImage);

function sobelFilter(path) {
    open(path);
    rename("Original");
    run("Duplicate...", "title=horizontal");
    run("Convolve...", "text1=[-1 -2 -1\n0 0 0\n1 2 1\n] normalize");
    run("16-bit");
    run("Square");
    selectWindow("Original");
    run("Duplicate...", "title=vertical");
    run("Convolve...", "text1=[-1 0 1\n-2 0 2\n-1 0 1\n] normalize");
    run("16-bit");
    run("Square");
    imageCalculator("Add create", "horizontal","vertical");
    rename("Sobel_Filtered");
    run("Square Root");
    run("Enhance Contrast", "saturated=0.35");
    setMinAndMax(0, 255);
    run("8-bit");
    selectWindow("vertical");
    close();
    selectWindow("horizontal");
    close();
}
```

Lines 93-120

Conditional code

To execute a part of the code if and only if a certain condition is met:

```
value1 = 17;
value2 = 18.98;
print(checkIfSameValues(value1, value2));

function checkIfSameValues(value1,value2){
    //Check if both values are the same
    reply = "nope, they are different";
    if(value1==value2){
        reply = "yep, both are the same";
    }
    return reply;
}
```


```
print("\\Clear");
value1 = 19;
value2 = 19;
print(checkIfSameValues(value1,value2));

function checkIfSameValues(value1,value2){
    //Check if both values are the same
    if(value1 == value2){
        reply = "yep, both are the same";
    } else {
        reply = "nope, they are different";
    }
    return reply;
}
```

Lines 124-139

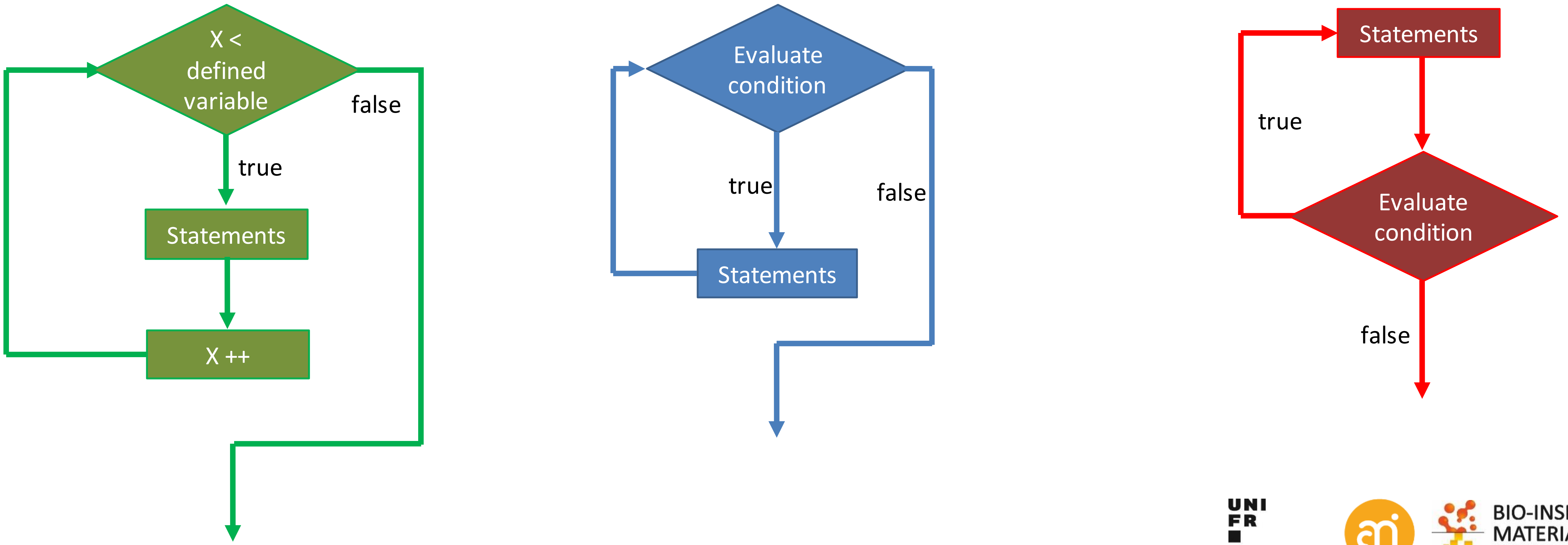
Note the ==, it is not = (that would be an assingment, which does not make sense here). Operators:

<, <=	less than, less than or equal
>, >=	greater than, greater than or equal
==, !=	equal, not equal
&&	boolean AND
||	boolean OR

Loops

To repeat instructions several times, loops are used.

for - runs a block of code a specified number of times
while - repeatedly runs a block of code while (as long as) a condition is true
do...while - runs a block of code once then repeats while a condition is true



FOR Loops

for

This loop is a good choice when the number of repetitions is known, or can be supplied aforehand by the user.

```
for (initialization; condition; increment) {
        statement(s);
}
```

```
for(x = 0; x <= 10; x++){
    print(x);
}
```

Output:

```
Log
File   Edit   Font
0
1
2
3
4
5
6
7
8
9
10
```

```
for(x = 0; x <= 100; x+=10){
    print(x);
}
```

Output:

```
Log
File   Edit   Font
0
10
20
30
40
50
60
70
80
90
100
```

Lines 186-195

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

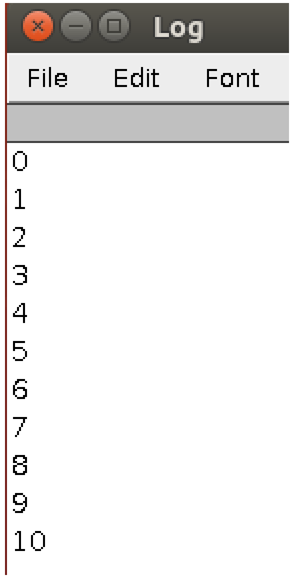
BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

WHILE Loops

the loop **must repeat until a certain "condition" is met**. If the "condition" is FALSE at the beginning of the loop, the loop is never executed.

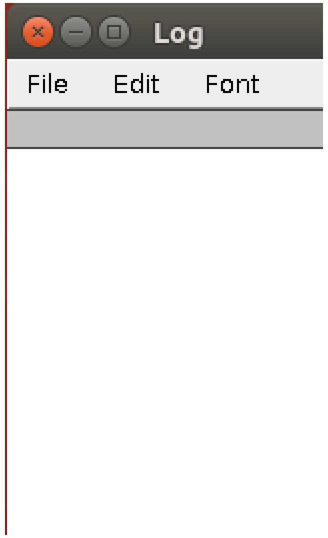
```
while (condition) {
            statement(s);
}
```

```
x = 0;
while (x <= 10){
    print(x);
    x = x + 1;
}
```



Lines 198-204

```
x = 11;
while (x <= 10){
    print(x);
    x = x + 1;
}
```



UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

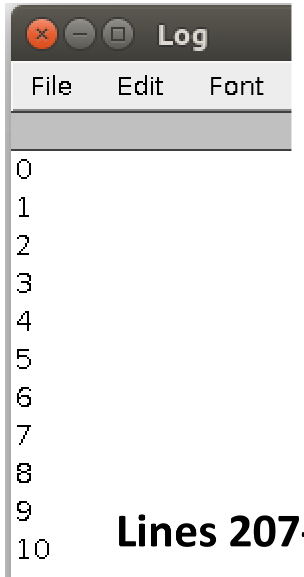
DO Loops

Same concept as the while loop except that the do-while **will always execute the body of the loop at least one time**.
Do-while is an exit-condition loop: the condition is checked at the end of the loop.

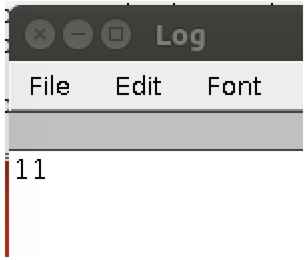
This looping process is a good choice when you are asking a question, whose answer will determine if the loop is repeated.

```
do {
        statement(s);
} while (condition);
```

```
x = 0;
do {
    print(x);
    x = x + 1;
} while (x <= 10)
```




```
x = 11;
do {
    print(x);
    x = x + 1;
} while (x <= 10);
```



Lines 207-214

}

Point operations

EXERCISE
Open an grayscale image and invert the image by a script (do not use the internal invert function).

You now know already enough to write a first simple script.

- Iterate through all pixels in the image to invert the 8 bit image. Obviously, you are not allowed to use the run(«invert») command.
- Suggestions:
- Use a loop
- getPixel, setPixel
- getDimensions

Point operations

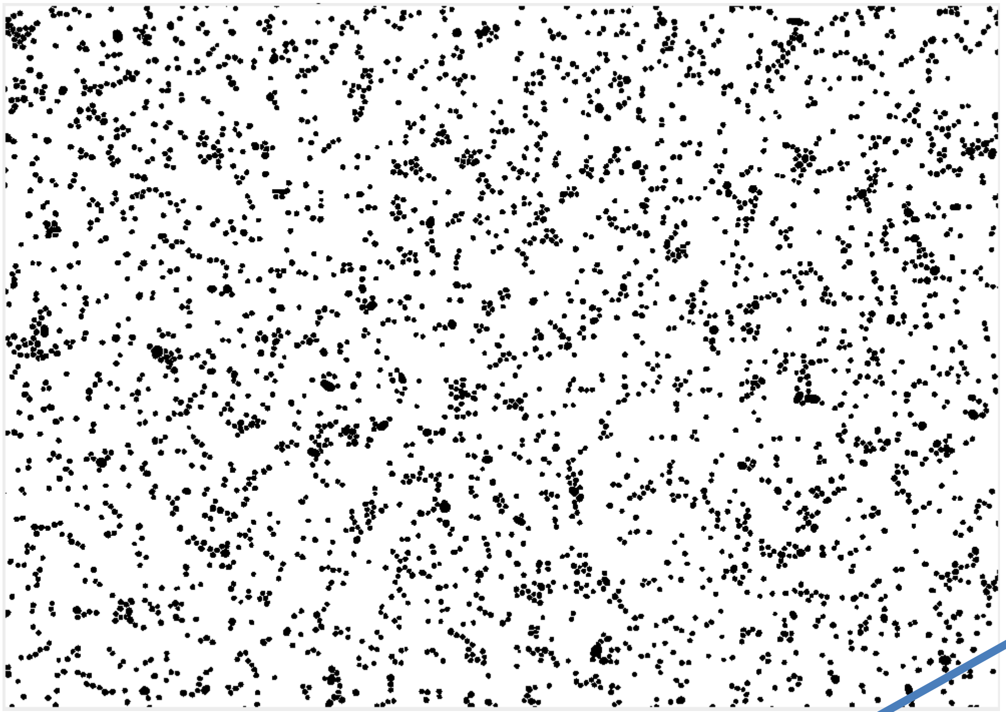
EXERCISE
Open an grayscale image and invert the image by a script (do not use the internal invert function).

You now know already enough to write a first simple script. Iterate through all pixels in the image to invert the 8 bit image. Combine the following:

```
// assuming an 8-bit grayscale image
path =  "C:/Users/vanheckd/Desktop/Scripts/Example 8.tif";
open(path);
getDimensions(width, height, channels, slices, frames);
for (x=0;x <= width; x++){
    for(y = 0; y <= height; y++){
        //setPixel: x position, y position, new intensity for that pixel
        // getPixel: get the intensity of the pixel in position x, y
        setPixel(x, y, 255 - getPixel(x,y));
    }
}
```

Lines 216-228

Result table functions



getResult («Column name», row number)
Will get a value from the result table (assumed there is one).

E.g. After running the «Measure particle« routine:

```
print(getResult("X", 3006));
```

Result: 1716.88697

	Label	Area	Mean	StdDev	Mode	Min	Max	X	Y	XM	YM	Perim.	BX	BY	Width	Height	Major	Minor	Angle	Circ.	Feret
3006	Example 11 - AuNP.tif	150.25960	255	0	255	255	255	1719.88697	1833.06592	1719.88697	1833.06592	44.58258	1712.64756	1825.58877	14.11765	14.11765	14.16350	13.50771	64.04100	0.95000	15.07766
3007	Example 11 - AuNP.tif	152.59525	255	0	255	255	255	1993.31691	1832.58457	1993.31691	1832.58457	45.31355	1985.29470	1825.58877	15.00000	14.11765	15.25882	12.73299	148.53376	0.93389	15.90685
3008	Example 11 - AuNP.tif	217.99321	255	0	255	255	255	2303.19080	1834.02785	2303.19080	1834.02785	53.62021	2295.00068	1825.58877	16.76471	16.76471	16.98644	16.33995	138.95039	0.95279	17.86629
3009	Example 11 - AuNP.tif	214.87902	255	0	255	255	255	1110.99777	1834.71921	1110.99777	1834.71921	53.40611	1102.94150	1826.47113	15.88236	16.76471	17.96937	15.22549	120.07982	0.94672	19.08822
3010	Example 11 - AuNP.tif	249.13509	255	0	255	255	255	1217.02701	1835.84613	1217.02701	1835.84613	58.39746	1208.82388	1826.47113	16.76471	19.41177	19.51386	16.25555	100.20070	0.91803	20.65536
3011	Example 11 - AuNP.tif	193.85825	255	0	255	255	255	1953.53176	1836.73513	1953.53176	1836.73513	52.06960	1945.58881	1828.23583	15.88236	16.76471	16.44064	15.01328	46.21663	0.89852	17.33549
3012	Example 11 - AuNP.tif	150.25960	255	0	255	255	255	1024.78924	1837.07079	1024.78924	1837.07079	43.85162	1018.23559	1830.00054	13.23530	14.11765	14.30625	13.37293	109.42868	0.98193	15.07766
3013	Example 11 - AuNP.tif	132.35302	255	0	255	255	255	1369.28241	1836.50919	1369.28241	1836.50919	40.83907	1363.23570	1830.00054	12.35294	13.23530	13.40305	12.57305	101.20546	0.99722	14.25485
3014	Example 11 - AuNP.tif	288.06245	255	0	255	255	255	1388.81440	1838.49021	1388.81440	1838.49021	63.38880	1377.35335	1830.00054	22.05883	16.76471	21.86038	16.77795	176.10052	0.90089	22.68523
3015	Example 11 - AuNP.tif	299.74066	255	0	255	255	255	1275.03361	1840.27136	1275.03361	1840.27136	64.42254	1264.41214	1830.88289	21.17648	18.52942	22.03309	17.32129	148.14138	0.90757	23.00896
3016	Example 11 - AuNP.tif	152.59525	255	0	255	255	255	1928.45495	1837.66711	1928.45495	1837.66711	45.09945	1920.88292	1830.88289	15.00000	13.23530	14.67268	13.24163	174.83065	0.94278	15.63534
3017	Example 11 - AuNP.tif	214.10047	255	0	255	255	255	321.85197	1840.35616	321.85197	1840.35616	53.62021	314.11774	1831.76524	15.88236	17.64706	18.17711	14.99695	61.12156	0.93577	18.94493
3018	Example 11 - AuNP.tif	147.92396	255	0	255	255	255	1010.21082	1839.10271	1010.21082	1839.10271	44.36849	1003.23559	1832.64760	14.11765	13.23530	14.39809	13.08109	162.66021	0.94428	15.18058
3019	Example 11 - AuNP.tif	173.61602	255	0	255	255	255	1756.18952	1839.66488	1756.18952	1839.66488	47.59512	1748.82404	1832.64760	15.00000	15.00000	15.16042	14.58104	24.96286	0.96311	15.90685
3020	Example 11 - AuNP.tif	164.27345	255	0	255	255	255	1407.35545	1840.49469	1407.35545	1840.49469	45.61632	1400.29453	1833.52995	14.11765	14.11765	14.73250	14.19715	167.35822	0.99206	15.78401
3021	Example 11 - AuNP.tif	156.48798	255	0	255	255	255	1619.99389	1840.10809	1619.99389	1840.10809	45.61632	1612.05930	1833.52995	15.00000	13.23530	14.78792	13.47361	169.77752	0.94504	15.63534
3022	Example 11 - AuNP.tif	189.18696	255	0	255	255	255	2498.36856	1841.66902	2498.36856	1841.66902	50.60767	2490.88309	1833.52995	15.00000	16.76471	16.86883	14.27961	125.39319	0.92826	17.86629

A propos: arrays

EXERCISE
Try to find the mean and the median value of the particle measurement of Figure 11

Median: use an array.

- Store all values (Feret) in one array
- Sort the array
- Pick out the middle value (=median)

Mean: sum all values and devide by the total number of values.

- Sum all values (Feret) in one variable
- Divide by the length of the table: nResults

A propos: arrays

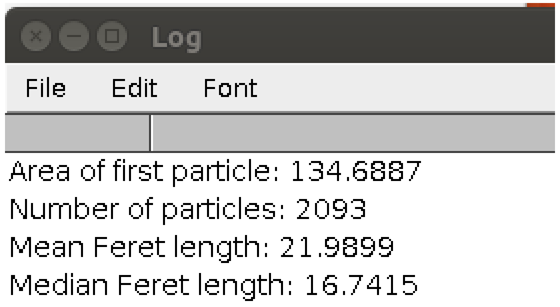
EXERCISE
Try to find the mean and the median value of the particle measurement of Figure 11

Median: use an array.
- Store all values (Feret) in one array
- Sort the array
- Pick out the middle value (=median)

Mean: sum all values and devide by the total number of values.
- Sum all values (Feret) in one variable
- Divide by the length of the table: nResults

```
open("/home/dimitri/Desktop/ImageJ basics/Advanced/Example 11 - AuNP.tif");
setAutoThreshold("Default");
//run("Threshold...");
run("Convert to Mask");

run("Set Measurements...", "area fit feret's display redirect=None");
run("Analyze Particles...", "  show=Nothing display exclude clear");
print("Area of first particle: " + getResult("Area", 0));
print("Number of particles: " + nResults);
values = newArray;
for (e=0;e<nResults;e++){
    summed = summed + getResult("Feret", e);
    values = Array.concat(values, getResult("Feret", e));
}
Array.sort(values);
print("Mean Feret length: " +summed / nResults);
print("Median Feret length: " + values[floor(nResults/2)+1]);
```



Lines 218-236

Remark

Commands, variables, arrays, loops and conditionals are not exclusive to FIJI/ImageJ!!

- All programming languages have them
- The concept is usually the same, but the syntax may differ

Python

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Java

```
String[] fruits = {“apple", “banana", “cherry"};
for (String i : fruits) {
 System.out.println(i);
}
```

ImageJ macro

```
fruits = newArray("apple", "banana", "cherry“);
for(x = 0; x < fruits.length; x++){
  print(fruits[x]);
}
```

English

```
Thanks
for
The
Programming
lesson
```

French

```
Merci
pour
la
leçon
de programmation
```

German

```
Danke
für
die
Programmier-
stunde
```

UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

A very short intro into R/Rstudio

Why? And Why R… I am fine with Excel!

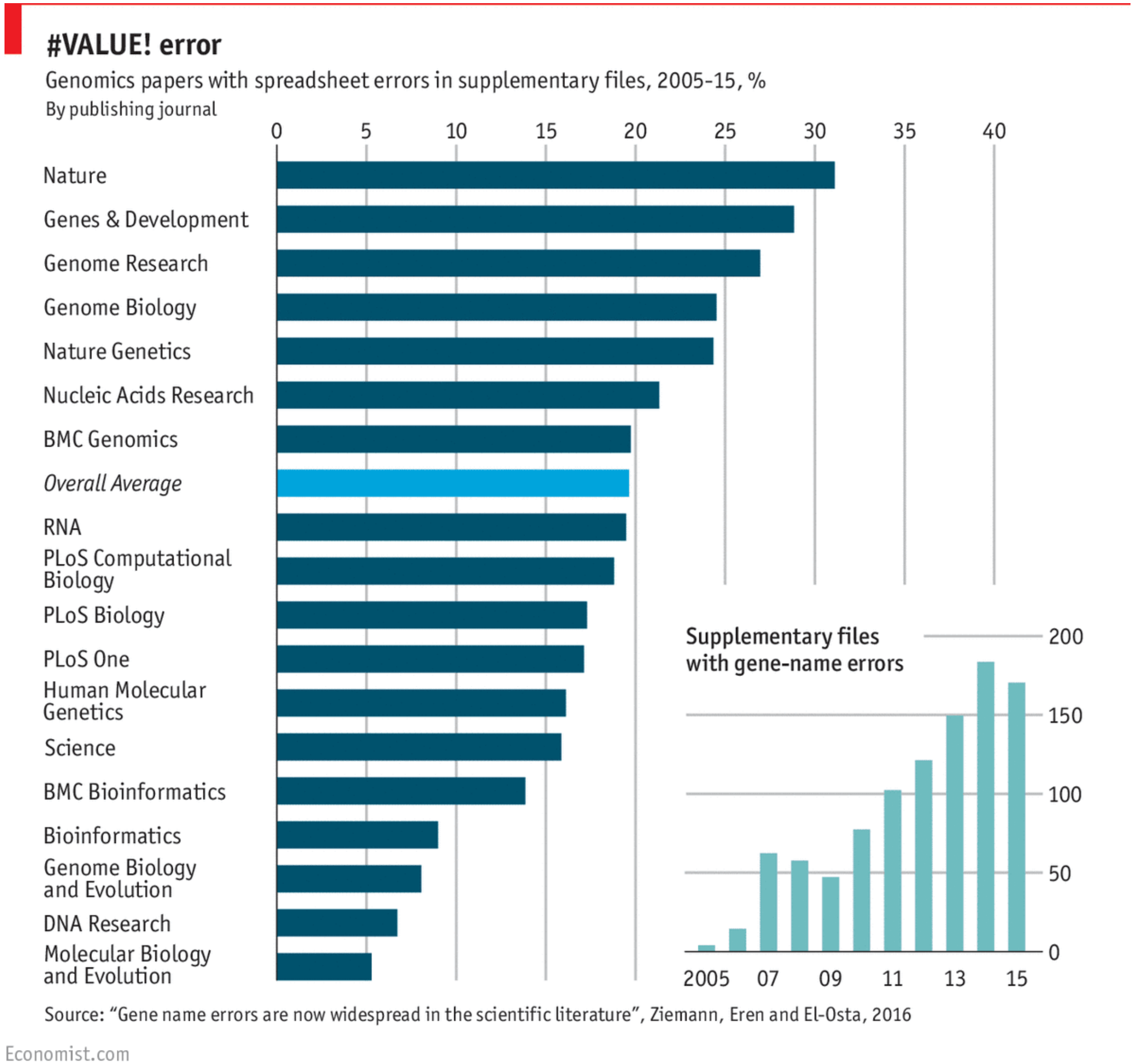
Natural science: Data > Information > Knowledge

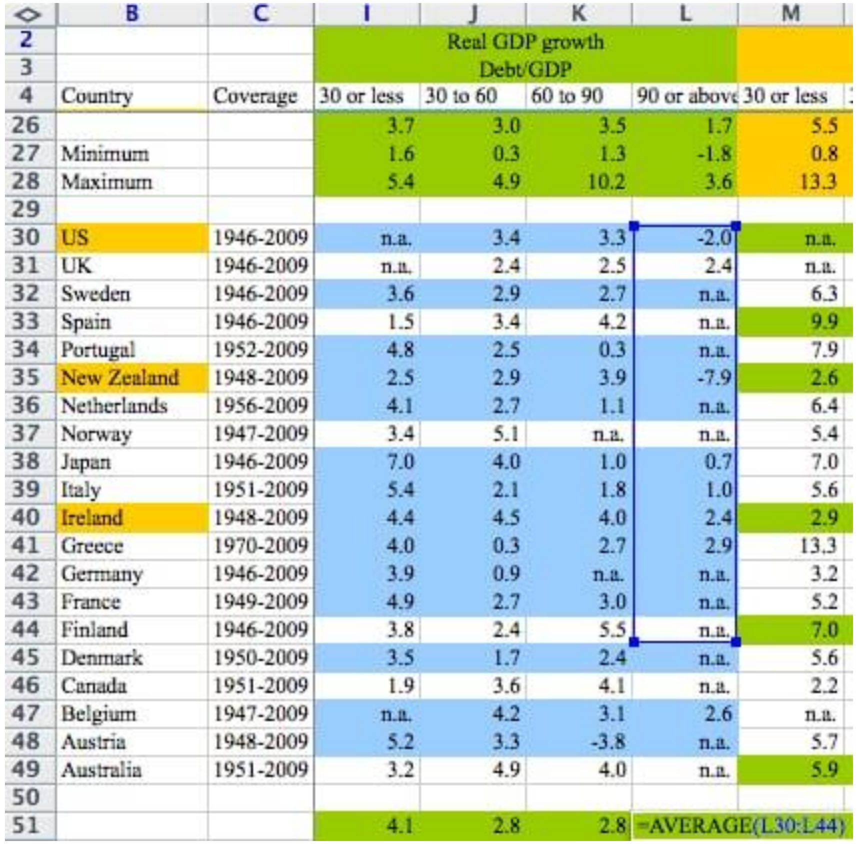
There is increasingly more emphasis on the first step: from data to information. Aka Data analysis

R:

- Is a great resource for data analysis, data visualization, data science and machine learning
- provides many statistical techniques (such as statistical tests, classification, clustering and data reduction)
- is easy to draw graphs in R, like pie charts, histograms, box plot, scatter plot, etc++
- works on different platforms (Windows, Mac, Linux)
- Is open-source and free
- has a large community support
- has many packages (libraries of functions) that can be used to solve different problems

A very short intro into R/Rstudio





◇	B	C	I	J	K	L	M
2			Real GDP growth				
3			Debt/GDP				
4	Country	Coverage	30 or less	30 to 60	60 to 90	90 or above	30 or less
26			3.7	3.0	3.5	1.7	5.5
27	Minimum		1.6	0.3	1.3	-1.8	0.8
28	Maximum		5.4	4.9	10.2	3.6	13.3
29							
30	US	1946-2009	n.a.	3.4	3.3	-2.0	n.a.
31	UK	1946-2009	n.a.	2.4	2.5	2.4	n.a.
32	Sweden	1946-2009	3.6	2.9	2.7	n.a.	6.3
33	Spain	1946-2009	1.5	3.4	4.2	n.a.	9.9
34	Portugal	1952-2009	4.8	2.5	0.3	n.a.	7.9
35	New Zealand	1948-2009	2.5	2.9	3.9	-7.9	2.6
36	Netherlands	1956-2009	4.1	2.7	1.1	n.a.	6.4
37	Norway	1947-2009	3.4	5.1	n.a.	n.a.	5.4
38	Japan	1946-2009	7.0	4.0	1.0	0.7	7.0
39	Italy	1951-2009	5.4	2.1	1.8	1.0	5.6
40	Ireland	1948-2009	4.4	4.5	4.0	2.4	2.9
41	Greece	1970-2009	4.0	0.3	2.7	2.9	13.3
42	Germany	1946-2009	3.9	0.9	n.a.	n.a.	3.2
43	France	1949-2009	4.9	2.7	3.0	n.a.	5.2
44	Finland	1946-2009	3.8	2.4	5.5	n.a.	7.0
45	Denmark	1950-2009	3.5	1.7	2.4	n.a.	5.6
46	Canada	1951-2009	1.9	3.6	4.1	n.a.	2.2
47	Belgium	1947-2009	n.a.	4.2	3.1	2.6	n.a.
48	Austria	1948-2009	5.2	3.3	-3.8	n.a.	5.7
49	Australia	1951-2009	3.2	4.9	4.0	n.a.	5.9
50							
51			4.1	2.8	2.8	=AVERAGE(L30:L44)	

The Reinhart-Rogoff error
(leading to wrong austerity conclusions)

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

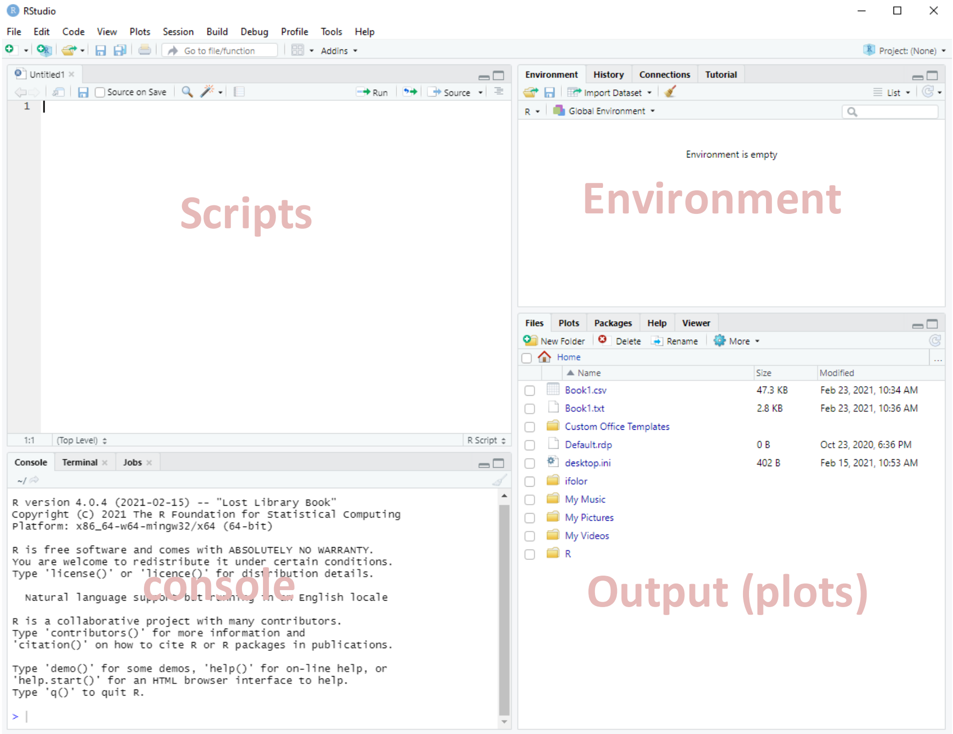
BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

A very short intro into R/Rstudio

Install

Install R: https://cran.rstudio.com (base package)

Install Rstudio: https://rstudio.com



UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Assigning variables syntax

Vectors
Similar to arrays in FIJI: A 1-D list of arguments

```
Vec <- vector(mode = "logical", length = 3)
```

FALSE FALSE FALSE

Matrices
A 2D list of arguments. All columns in a matrix must have the same length & mode (numeric, logic, …)

```
Mat <- matrix(1:9, ncol= 3)
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Arrays
A nD list of arguments
Arrays can have more than two dimensions.

```
Arr <- array(1:4, c(3,2,2))
```

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

, , 2

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

Data Frames
A data frame is more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.). Columns can also get headers

```
d <- data.frame(letters[1:3], 1:3)
colnames(d) <- c("letters", "numbers")
```

```
  letters numbers
1       a       1
2       b       2
3       c       3
```

Lists
An ordered collection of objects (components). A list allows you to gather a variety of (possibly unrelated) objects under one name.

```
List <- list(Vec, Mat)
```

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

[[2]]
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Assigning variables syntax

Generic

```
VariableName <- c("something", "something else")
```

1. Variable name
2. <- (smaller then, hyphen)
3. c: **c**ombines arguments
4. Round bracktets open
5. Your arguments, number, texts, variables, …
6. Round brackets closed

Want to know?

```
is.array(variable)
is.matrix(variable)
is.vector(variable)
is.data.frame(variable)
Is.list(variable)
```

Want to convert/to be sure?

```
Now_arr <- as.array(variable)
Now_Mat <- as.matrix(variable)
Now_vect <- as.vector(variable)
Now_df <- as.data.frame(variable)
Now_list <- as.list(variable)
```

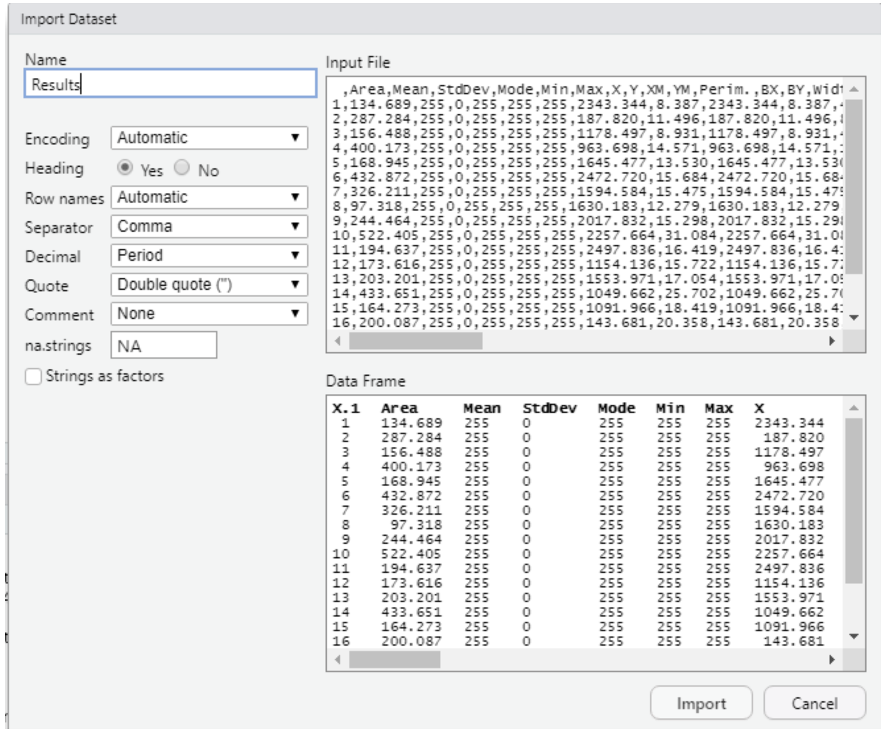
Things to do

1. Import your data (CSV, XLS, …)
2. Look at the data
3. Organize your data / extract parts of it
4. Do statistics on it
5. Plot (parts of) the data + make the plots look good
6. Correlate your data, model it
7. Output data, plots, ect…

Importing csv, xls, … in R

In the environment: Import dataset > text base

1. Use heading if available
2. Press import: the table will be shown
3. In the console, you see the commands you (graphically) used




```
Results <- read.csv("C:/Users/vanheckd/Desktop/scripting/Results.csv")
```

Also possible:

- XLSX, TXT, HTML, and other Common Files into R
- JSON, XML Files
- SAS, SPSS, and Other Datasets into R
- Stata Files
- Systat Files
- Minitab Files
- RDA or RData Files
- Directly read Databases (mySQL, …) from the internet
- Import through Webscraping

UNI FR UNIVERSITÉ DE FRIBOURG UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS NATIONAL CENTER OF COMPETENCE IN RESEARCH

Look at your data (assuming number data)

Get a summary:

Summary(Results)

```
> summary(Results)
      X.1              Area              Mean          StdDev       Mode           X          
 Min.   :   1.0   Min.   :  0.093   Min.   :255   Min.   :0   Min.   :255   Min.   :  4.642  
 1st Qu.: 661.8   1st Qu.: 18.977   1st Qu.:255   1st Qu.:0   1st Qu.:255   1st Qu.:211.731  
 Median :1322.5   Median : 21.582   Median :255   Median :0   Median :255   Median :442.298  
 Mean   :1322.5   Mean   : 24.230   Mean   :255   Mean   :0   Mean   :255   Mean   :443.428  
 3rd Qu.:1983.2   3rd Qu.: 25.210   3rd Qu.:255   3rd Qu.:0   3rd Qu.:255   3rd Qu.:666.869  
 Max.   :2644.0   Max.   :147.724   Max.   :255   Max.   :0   Max.   :255   Max.   :902.484  

       Y              Perim.             BX               BY              Width             Height       
 Min.   :  2.773   Min.   : 0.863   Min.   :  1.83   Min.   :  0.61   Min.   : 0.305   Min.   : 0.305  
 1st Qu.:164.053   1st Qu.:15.947   1st Qu.:208.93   1st Qu.:161.34   1st Qu.: 4.880   1st Qu.: 4.880  
 Median :325.842   Median :17.062   Median :439.81   Median :322.84   Median : 5.185   Median : 5.185  
 Mean   :322.285   Mean   :18.188   Mean   :440.64   Mean   :319.49   Mean   : 5.582   Mean   : 5.589  
 3rd Qu.:479.729   3rd Qu.:18.535   3rd Qu.:663.83   3rd Qu.:477.02   3rd Qu.: 5.795   3rd Qu.: 5.795  
 Max.   :637.084   Max.   :74.644   Max.   :899.75   Max.   :634.10   Max.   :16.165   Max.   :20.130  

     Major             Minor             Angle            Circ.            Feret            IntDen       
 Min.   : 0.344   Min.   : 0.344   Min.   :  0.00   Min.   :0.278   Min.   : 0.431   Min.   :  23.72  
 1st Qu.: 5.136   1st Qu.: 4.687   1st Qu.: 40.37   1st Qu.:0.913   1st Qu.: 5.456   1st Qu.: 4839.16  
 Median : 5.487   Median : 5.017   Median : 83.69   Median :0.931   Median : 5.859   Median : 5503.36  
 Mean   : 5.879   Mean   : 5.108   Mean   : 87.32   Mean   :0.916   Mean   : 6.203   Mean   : 6178.64  
 3rd Qu.: 5.970   3rd Qu.: 5.373   3rd Qu.:134.92   3rd Qu.:0.948   3rd Qu.: 6.288   3rd Qu.: 6428.50  
```

See all data:

Results$Feret

```
Results$Feret
  [1]  5.335  5.076  5.076  5.698  7.446  5.624  6.288  5.327  7.649  5.327  5.498  5.827  5.335  6.176
 [15]  5.498  8.212  6.463  5.335  6.008  6.288  5.456  5.335  5.430  5.859  6.434 17.706  4.822  5.827
 [29]  5.867  5.673  8.480  6.484  5.405  5.498  5.430  5.922  5.498  5.498  5.327  5.498  4.927  6.016
 [43]  6.717  6.598  6.176  5.430  6.288  5.498  5.922  6.434  5.498  5.891  5.755  5.076  6.854  4.822
 [57]  5.456  6.077 10.653  6.527  6.176  5.859  5.607  5.049  4.822  6.347 10.933  5.673  6.820  6.138
 [71]  5.624  5.498  6.280  5.755  8.496  5.922  5.867  5.327  6.100  5.498  5.755  5.673  5.607  6.347
 [85]  4.890  5.755  6.854  5.185  5.698  5.787  5.827  5.698  5.335  6.054  6.962  5.212  6.016  5.405
 [99]  6.881  5.755  5.755 13.910  6.016  5.498  5.498  6.288  5.891  5.524  6.077  8.058  5.049  6.077
[113]  5.002  6.100  6.100  5.922  7.041  6.854  6.527  5.673  5.327  6.016  5.498  5.867  6.176  5.891
[127]  6.077  5.755  6.369  6.772  7.022  6.138  5.076  6.280  6.288  6.176  6.008  6.176  6.288  6.054
[141]  8.297  5.755  5.335  6.689  5.335  6.968  7.533  5.922  5.247  5.624  5.076  4.503  6.434  6.221
[155]  8.325  5.498  6.434  5.673  5.787  8.803  5.859  5.498  5.787  5.405 12.280  5.498  5.247  5.624
[169]  6.100  5.730  5.247  5.113  8.723  6.176  5.624  5.430  6.689  5.335  6.054  5.405  5.076  5.247
[183]  5.566  5.787  5.456  5.922  5.698  6.470  6.008  8.104  5.185  5.607  6.016  5.599  5.498 16.006
[197] 15.317 13.196  5.498  6.288  5.827  5.673  5.624  5.265  5.498  5.859  5.787  5.456  6.751  5.673
[211]  6.717  5.698  5.335  5.430  5.030  9.878  7.383  6.347  5.755  5.335  5.335  5.076  5.755  5.891
[225]  5.498  5.498  5.859  5.335  5.673  6.347  5.396  5.456  5.867  7.120  7.776  5.922  6.176  8.589
[239]  5.698  5.076  5.456  5.673  5.498  5.922  7.383  5.247  5.498  6.470  5.247  6.570  5.827  5.498
[253]  6.176 17.148  5.755  5.327  5.335  5.335  6.288  6.176  5.498  5.002  6.077  4.745  5.867  6.484
[267]  5.859  6.016  5.624  5.599  5.891  5.498  5.247  5.624  5.624  4.927  5.456 11.674  5.566  6.549
[281]  5.787  5.002  5.673  7.918  6.288  5.787  6.434  5.430  5.327  6.751  6.138  5.599  5.922  5.755
```

Or Environment > Data > doubleclick “Results”

Get one column: $

Summary(Results$Feret)

```
summary(Results$Feret)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.431   5.456   5.859   6.203   6.288  21.552 
```

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

BIO-INSPIRED MATERIALS
NATIONAL CENTER OF COMPETENCE IN RESEARCH

Organize your data (assuming number data)

Select first n values:

```
newVar <- Results$Feret[firstValue:lastValue]
```

```
particles=1000
feret <- Results$Feret[0:particles]
minferet <- Results$MinFeret[0:particles]
NewResults <- data.frame(feret, minferet)
```

```
length(NewResults$feret)
```

1000

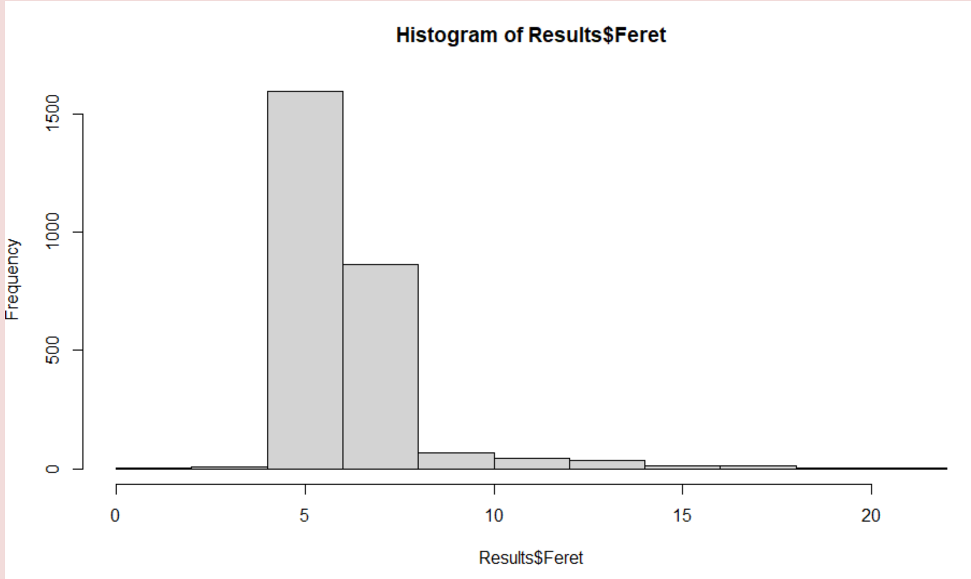
Filter for values:

```
newVar <- Results[Results$Feret < 25,]
```

```
FilteredResults <- Results[Results$Feret < 25,]
Filteredferet <- FilteredResults$Feret[0:particles]
Filteredminferet <- FilteredResults$MinFeret[0:particles]
NewResultsFiltered <- data.frame(Filteredferet, Filteredminferet)
colnames(NewResultsFiltered) <- c("Feret", "MinFeret")
```

Plot a histogram:

```
hist(NewResults$Feret)
```



Do statistics

Many statistical test in the base package, with many more to download:

```
#Normality test
shapiro.test(feret)
shapiro.test(minferet)
length(minferet)
```

- Two-sample differences tests (e.g. t-test).
- Non-parametric tests (e.g. U-test).
- Matched pairs tests (e.g. Wilcoxon).
- Association tests (e.g. Chi squared).

...

Common R packages:

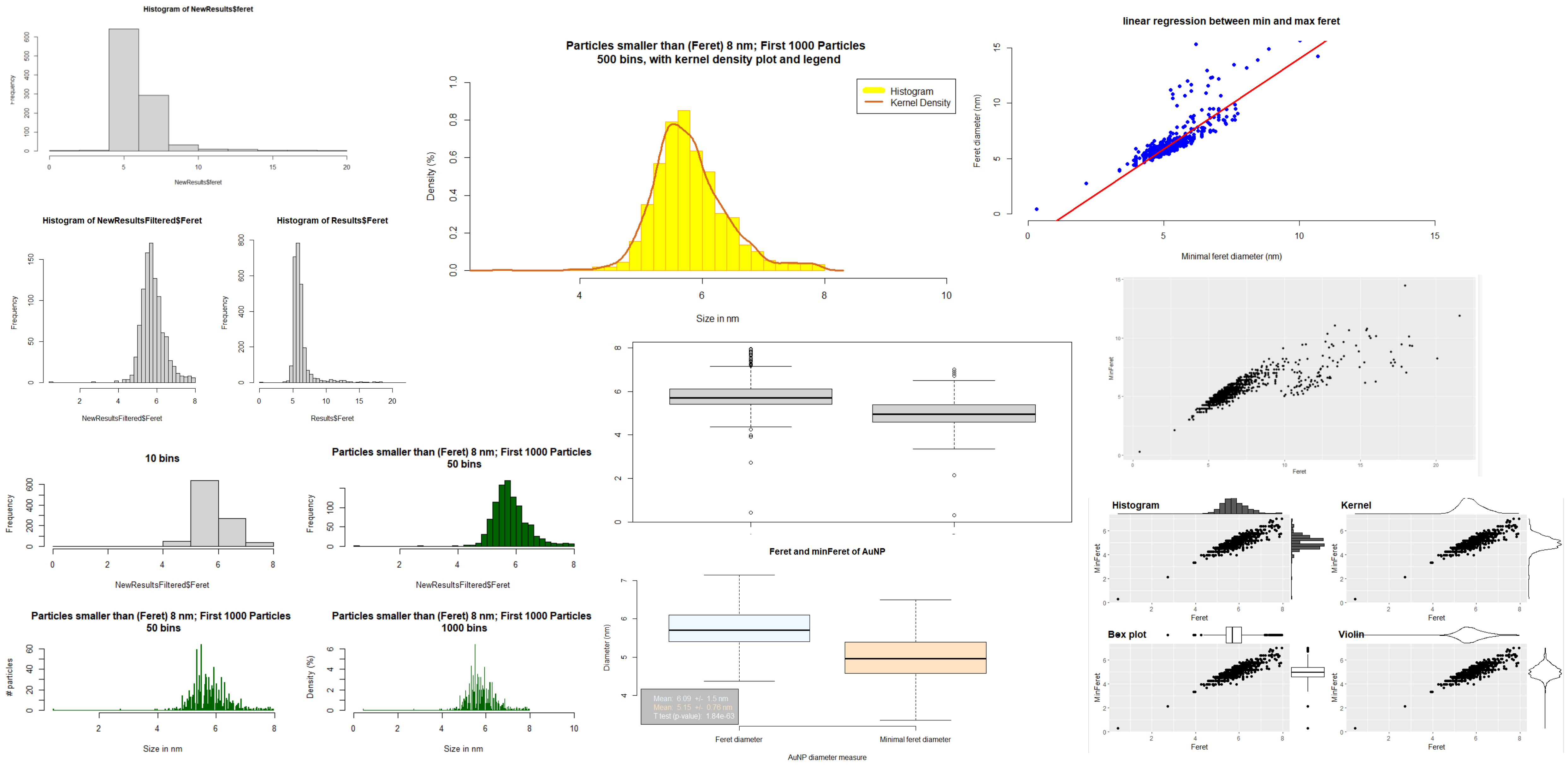
ggplot2 (Grammar of Graphics)	popular data visualization library.
data.table	fast handling a vast amount of data (up to 100GB)
dplyr	data manipulation
tidyr	helps to create tidy data (see also: Tidyverse)

Installing and loading packages:

```
packages.install("ggplot2")
```

```
library(ggplot2)
```

Plot data


Histogram of NewResults$feret
Frequency
NewResults$feret
Particles smaller than (Feret) 8 nm; First 1000 Particles
500 bins, with kernel density plot and legend
Histogram
Kernel Density
Density (%)
Size in nm
linear regression between min and max feret
Feret diameter (nm)
Minimal feret diameter (nm)
Histogram of NewResultsFiltered$Feret
Frequency
NewResultsFiltered$Feret
Histogram of Results$Feret
Frequency
Results$Feret
MinFeret
Feret
10 bins
Frequency
NewResultsFiltered$Feret
Particles smaller than (Feret) 8 nm; First 1000 Particles
50 bins
Frequency
NewResultsFiltered$Feret
Feret and minFeret of AuNP
Diameter (nm)
Mean: 6.09 +/- 1.5 nm
Mean: 5.15 +/- 0.76 nm
T test (p-value): 1.84e-63
Feret diameter
Minimal feret diameter
AuNP diameter measure
Histogram
Kernel
Box plot
Violin
MinFeret
Feret
Particles smaller than (Feret) 8 nm; First 1000 Particles
50 bins
particles
Size in nm
Particles smaller than (Feret) 8 nm; First 1000 Particles
1000 bins
Density (%)
Size in nm


That's all Folks!