FAIR_Bioinfo

Practical session : Literate programming

This session will be divided into three parts:

- General observations on Markdown
- RMarkdown / RStudio
- Jupyter (links to other resources)

Table of Contents

Version : 1.0 (30/08/2020)

Contacts :

- Thomas Denecker
- Celine Hernandez celine.hernandez@i2bc.paris-saclay.fr
- Claire Toffano-Nioche

A. General information

The IFB RStudio server: https://rstudio.cluster.france-bioinformatique.fr/
Counts data used for the analysis : https://zenodo.org/record/3997137

Additional resources

Analyzing RNA-seq data using DESeq2 :
http://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

RMarkdown
In RStudio: https://rmarkdown.rstudio.com/lesson-8.html
And more generally : http://rmarkdown.rstudio.com

Jupyter
General documentation : https://jupyter-notebook.readthedocs.io

Note that the commands you have to enter are inside a blue box. The first '$' character should **not** be copied.

B. Introduction to Markdown

1. What is Markdown?

Markdown is used on the Internet to specify formatting in a lightweight and user-readable manner for :

- Titles
- Emphasis
- Lists
- Image
- Link
- Citation
- Tasks
- Table
- And more…

2. Where is Markdown ?

Everywhere ! As an introduction to the language, we will have a look at two renown websites : Wikipedia and Github, plus how it is used in RStudio (RMarkdown).

Wikipedia : https://en.wikipedia.org/wiki/Help:Cheatsheet
GitHub : https://guides.github.com/features/mastering-markdown/#syntax
RMarkdown : https://rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

Q: How do you format a text in italic for each site ?
Q: How do you write an unordered list ?

The Markdown specification had ambiguities. By consequence, its implementation can differ from one tool to another… There are many Markdown*S*. Always check the specifications with which you are working !

C. RMarkdown (RStudio)

1. Quick introduction to RStudio

“RStudio is an integrated development environment (IDE) for R, a programming language for statistical computing and graphics. It is available in two formats: RStudio Desktop is a regular desktop application while RStudio

Server runs on a remote server and allows accessing RStudio using a web browser." Wikipedia (30/08/2020)

- What about Docker?

This is not an exercise and is provided for your information. You can skip this part and come back to it if needed be.

Docker can be used to install and run RStudio on your computer. You can find a very nice tutorial here:
https://ropenscilabs.github.io/r-docker-tutorial/02-Launching-Docker.html
This tutorial describes how to:
Launch RStudio inside of a Docker container
Connect to the interface
Link a volume to a Docker container

Note that if you want to install new packages and stay reproducible you might want to build a new images on top of this one. To do so, you will need to write a Dockerfile.
https://community.rstudio.com/t/rocker-rstudio-container-update/23106

2. RMarkdown (DEseq2 analysis)

Goal: Transform an R script into an RMarkdown file or R Notebook.

a) Check that you have access to the data

For this practical exercise we will use a table of counts from a RNA-seq gene expression experiment from the Bioproject PRJNA304086. It studies iron metabolism in Ostreococcus tauri. We will compare two experiments: with iron (runs SRR3099585-87) and without iron (SRR3105697-99) at t=9h. You can find more information on the experiment here :
https://www.ncbi.nlm.nih.gov/bioproject/?term=PRJNA304086

The counts dataset can be downloaded from this Zenodo URL: https://zenodo.org/record/3997137
On the IFB RStudio server, it can be accessed from
/shared/data/projects/fair_training2020/Result/counts_matrix.txt

b) Execute the provided R script

As the goal of this practical session is not to learn how to perform such an analysis, the script will be provided at the beginning of the session. If you want to learn more about DESeq2, we recommend to read its Bioconductor vignette:
http://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

Briefly, the provided R script loads the file containing the counts into an appropriate object and starts a differential expression analysis between the two groups. It displays the results both as a table ordered by adjusted p-value and as a plot.

In RStudio, you can execute the script by:

1. Selecting the whole code.
2. Clicking on Run (top right hand side corner of the editor pane).

Check that the code can be correctly executed.
The output of each command will appear in the Console pane. The plot at the end of the script will be displayed in the Plots pane.

Note that the script's code can be found at the end of this document for an easy reference.

c) RMarkdown files

- The global skeleton

We will start by creating a new RMarkdown file.
Click on File -> New File -> R Markdown…
Keep the default choice "Document". You can provide a title as well as your name in the fields. Click on "OK".

The structure of a document is always the same.

* Header
Optional section of render options written as key:value pairs (YAML).
Always located at the beginning of the file and between two lines of - - - characters.

* Text
Description of the code, or any text formatted with markdown, mixed with:

* Chunks of executable code
Each chunk begins with the characters ```{r} and ends with ```

RMarkdown will execute the code and append the results to the document. It will use the location of the .Rmd file as the working directory

More details here: https://rmarkdown.rstudio.com/authoring_quick_tour.html

- Knit the default Rmd file

When you click the ‘Knit’ button (top left hand side of the editor pane) a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

In fact, this button refers to the package knitr.
https://yihui.org/knitr/

- Knit from the command line.

RStudio only provides a convenient button to knit an Rmd file. But you can do the same with an R command.

```
$ rmarkdown::render("session_07_Rmarkdown.Rmd")
```

This will generate the equivalent PDF from your Rmd file.

RMarkdown and knitR are R libraries. You don’t need RStudio to use them.

- Adapt the Rmd file

Replace the code in the chunk by the content of the R script. Do not modify the header, but remove most of the text. Knit the document. A PDF file will appear, its content displaying the step by step differential analysis of your RNA-seq data.

Congratulations!
You just successfully created your first reproducible R analysis !

a) R Notebooks

The main difference with RMarkdown files is that Notebooks interactively display the results of code execution.

Same as for the precedent exercise, you can open a new R Notebook and insert the R code inside one of the chunks. This time, click on the Run all button (top right hand side) or in the top right corner of each chunk.

- Improve the Rmd file

Have a look at the cheatsheet and try a few improvements.
https://rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

1/ Separate chunks.

The R code can be divided into a set of paragraphs. Try to move each one of them into a separated chunk, and the comments into the text part. Use different Title levels.

2/ Chunk options

Look at the cheatsheet. For instance, try different levels of verbosity.
* `echo = FALSE` parameter can be added to the code chunk to prevent printing of the R code that generated the plot.
* 'message=FALSE' will not display messages, like when a library is loaded.

Q: Is it a good idea not to display all this information?
Q: Would your answer be the same if you had to send the results of the analysis to a collaborator who doesn't know R?

3/ Insert an inline value

Text can be generated on-the-fly using inline variables put between two $ signs. Formulas can be written using twice two $$ signs.

4/ Change the size of the plot.
This can be changed with the figure.width and figure.height parameters.

Q: What is missing in this file for the analysis to be completely reproducible?
Q: Do you know how to install a specific library version?

- Other engines

Look into the top right hand corner for the "Insert" button.

Q: How many different chunk types can you insert?

Try to insert a chunk from a different engine, for instance Bash. In the chunk, type the unix command 'pwd'. Then Run the chunk.
Q: what path is displayed?

Just after the Bash chunk, insert an R chunk. Inside the chunk, write “getwd()” and run the chunk.
Q: Is the R session’s current working directory the same?

Q: If you change directory in the bash chunk using a ‘cd’ command, does it change the current working directory in a successive R chunk?

Q: If you add a new Bash chunk, in which folder will it be located?

Chunks other than R are working in distinct sessions, making it difficult to share variables.
Q: How would you share a simple variable from an R chunk ? And a more complex variable like a table?

Bonus information you can explore:

- The full list of knitr examples

https://github.com/yihui/knitr-examples

- Add interactive elements

https://rmarkdown.rstudio.com/lesson-14.html

D. Jupyter (Bonus)

1. Quick introduction to Jupyter

A bit of history...

- 2011 : IPython (interactive Python shell) with notebook functionalities
- 2014 : Spin-off project called Project Jupyter
- a non-profit, open-source project maintained by a strong Community
- ”Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license”
- A reference to the three core programming languages supported byJupyter (Julia, Python and R)

See: https://jupyter.org/

Noticeable differences with R Notebooks

- Engines cannot be mixed. Each Jupyter notebook is associated to one kernel while one R Notebook can run chunks from different languages.
- Specialized blocks (no inline code inside a Markdown block).

2. Install Jupyter

With Conda

```
$ conda install -c conda-forge notebook
```

Congratulations, you have installed Jupyter Notebook! To run the notebook, run the following command at the Terminal (Mac/Linux) or Command Prompt (Windows):

```
$ jupyter notebook
```

You can find more information here:
https://jupyter.org/install

With Docker

The Jupyter Community maintains a set of Docker images with various contents. To pull a Docker image containing a basic version of Jupyter, you can use this command:

```
$ docker pull jupyter/minimal-notebook:latest
```

Be careful as this image is quite big (988.4MB).
To start a container you need to provide port numbers :

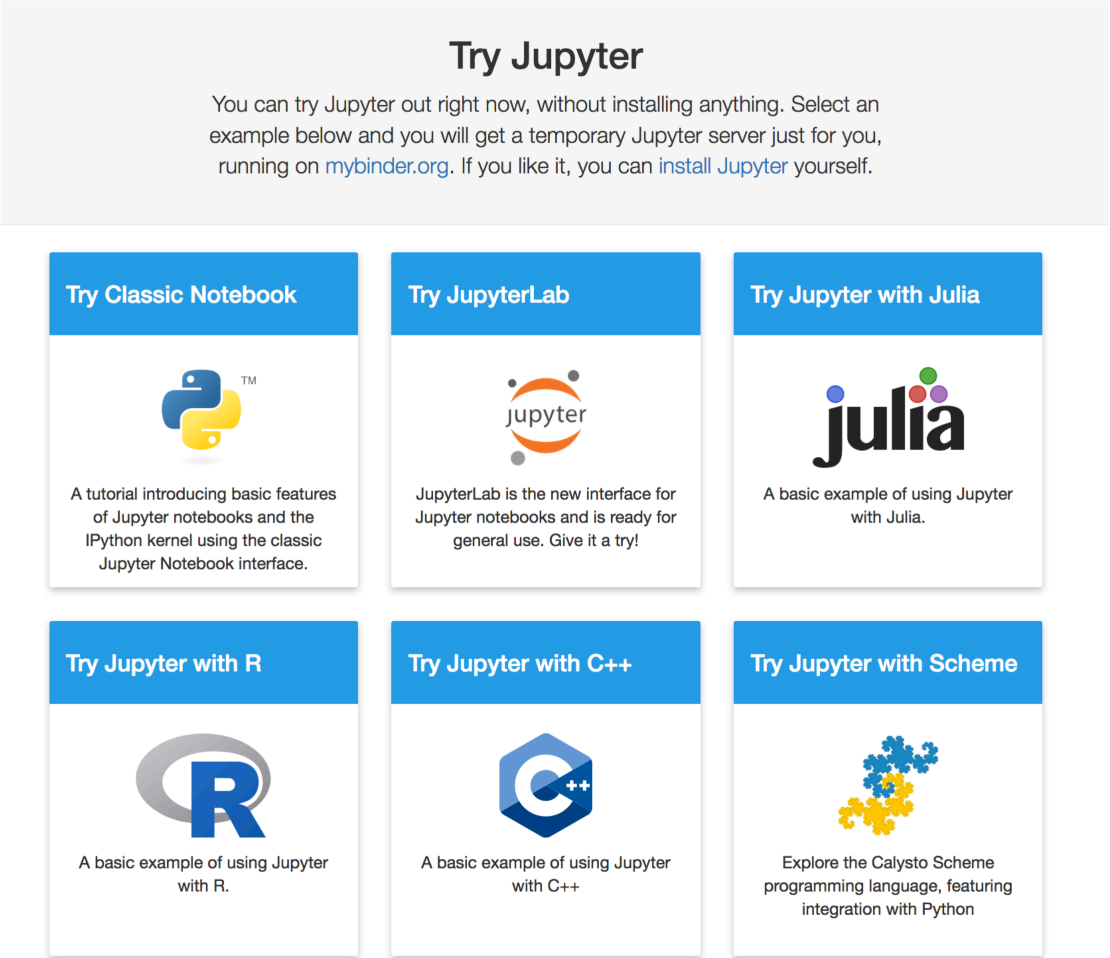
```
$ docker run -p 8888:8888 jupyter/minimal-notebook:latest
```

The server will be exposed on host port 8888. The server logs appear in the terminal and include a URL to the notebook server.
Don’t forget to also map a local folder.

The DockerHub page : https://hub.docker.com/u/jupyter
You can find more information here:
https://jupyter-docker-stacks.readthedocs.io

3. Try Jupyter online

Go to : https://jupyter.org/try



Try Jupyter

You can try Jupyter out right now, without installing anything. Select an example below and you will get a temporary Jupyter server just for you, running on mybinder.org. If you like it, you can install Jupyter yourself.

Try Classic Notebook

A tutorial introducing basic features of Jupyter notebooks and the IPython kernel using the classic Jupyter Notebook interface.

Try JupyterLab

JupyterLab is the new interface for Jupyter notebooks and is ready for general use. Give it a try!

Try Jupyter with Julia

A basic example of using Jupyter with Julia.

Try Jupyter with R

A basic example of using Jupyter with R.

Try Jupyter with C++

A basic example of using Jupyter with C++

Try Jupyter with Scheme

Explore the Calysto Scheme programming language, featuring integration with Python

And more…

Select one of the proposed tutorials or example, more specifically:

- “Try Classic Notebook” is a tutorial that will guide you through the main functionalities of the interface.
- “Try Jupyter with R” is an example of what Jupyter looks like with an R kernel. You can see a set of code blocks using ggplot2 and interactively modify them.
- “Try JupyterLab” for a tour of its new functionalities.

This is the end of the practical session. We hope you enjoyed it.
Don’t hesitate to ask any questions and feel free to contact us any time after the session (email addresses on the front page).

E. Annex: R script to perform the DESeq2 analysis.

Provided as a reference.

```
# As a first step in the analysis, we need to load a count matrix.
# This matrix will be used by DESeq2 to perform its differential analysis,
# i.e. to find genes that are differentially expressed between conditions.
# On the IFB RStudio server, data is available here:
# "/shared/data/projects/fair_training2020/Result/counts_matrix.txt"
counts <- read.table("counts.txt", sep="\t", row.names="Geneid",
header=TRUE)
counts <- counts[,1:5]

# A first look at the count matrix. We have 6 samples.
head(counts)

# Then we need to provide the experimental design so that DESeq2
# knows which sample groups to compare.
sampleInfo <- data.frame(condition = c("iron", "iron", "noIron", "noIron",
"noIron"),
                         row.names = names(counts))
sampleInfo

# Next step: load the data into an object useable by DESeq2.
library("DESeq2")
dds <- DESeqDataSetFromMatrix(countData = counts,
                              colData = sampleInfo,
                              design = ~ condition)
dds

# Differential expression analysis
# Perform the analysis by providing the data to DESeq
dds <- DESeq(dds)
# Extract the results
res <- results(dds)
# Display the results
# Re-order by adjusted p-value
resOrdered <- res[order(res$padj),]
resOrdered

# Display a summary of the results
summary(resOrdered)

# Display a MA plot of the dataset.
# Statistically significant genes are displayed in red.
plotMA(res)

# Keep versions of the packages used for the analysis
sessionInfo()
```