



Managing
dependencies



Managing and executing
analysis workflow



Versioning and
collaborating on code
(and some other files)



Connecting code
and reporting

and...



Isolating and exporting
environment

Typical guidelines for keeping a notebook of wet-lab work

Record everything you do in the lab, even if you are following a published procedure.

Use a bound notebook so that tear-out would be visible.

When you finish a page, put a corner-to corner line through any blank parts that could still be used for data entry.

Write a title for each and every new set of entries.

The investigator and supervisor must sign each page.

If you make a mistake, put a line through the mistake and write the new information next to it.

If you're testing a specific hypothesis, write it down beforehand.

All pages must be pre-numbered.

Each page should be numbered and dated consistently.

Properly introduce and summarize each experiment.

Use a ball point pen so that marks will not smear nor will they be erasable.

Use a ball point pen so that marks will not smear nor will they be erasable.

It is critical that you enter all procedures and data directly into your notebook in a timely manner.

Typical guidelines for keeping a notebook of dry-lab work

Literate programming

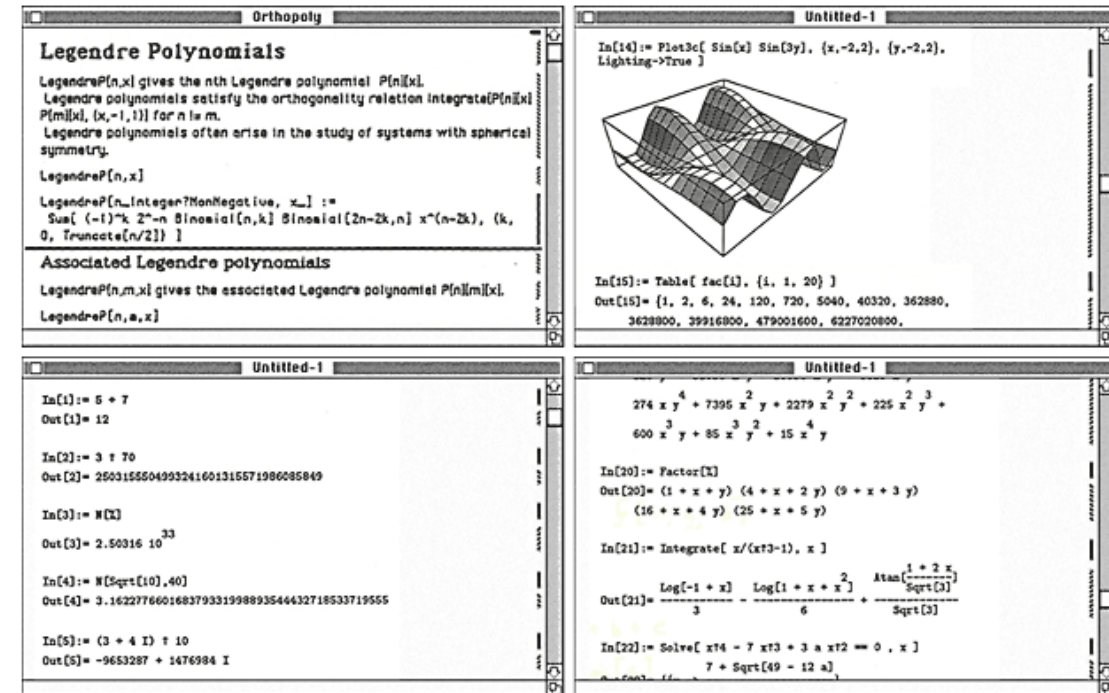
Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

Donald Knuth (1984)

Literate computing

A literate computing environment is one that allows users not only to execute commands interactively, but also to store in a literate document the results of these commands along with figures and free-form text.

Millman KJ and Perez F (2014)



Wolfgang Mathematica notebook (1988)

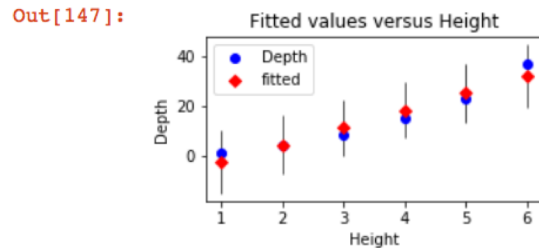
Study of velocity/energy relationship (1722-03-17, W. Gravesande)

If I drop brass balls from various heights and measure penetration depth in a block of clay, can I settle the dispute regarding conservation of energy?

```
In [146]: import statsmodels as sm; import matplotlib.pyplot as plt; import numpy
plt.rcParams["figure.figsize"] = (4,2)
data = {"Depth":[1.1, 4.3, 8.7, 15.5, 23.2, 37.0], "Height":[1, 2, 3, 4, 5, 6]}
```

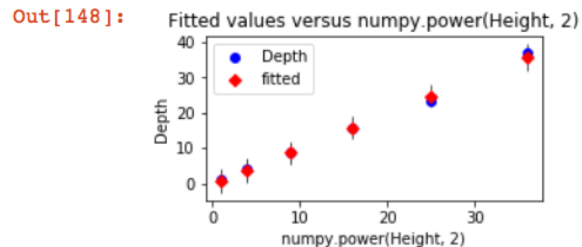
Test Bessler's hypothesis that kinetic energy, and thereby penetration depth, is linear with respect to height.

```
In [147]: res = sm.formula.api.ols(formula = 'Depth ~ Height', data = data).fit()
sm.graphics.api.plot_fit(res, 1)
```



Okayish, but maybe Bernoulli's quadratic model fits better?

```
In [148]: res = smf.ols(formula = 'Depth ~ numpy.power(Height,2)', data = data).fit()
sm.graphics.api.plot_fit(res, 1)
```



Neat!

- The Jupyter Notebook is a web application for *interactive* data science and scientific computing.
- In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.
- The ability to execute code from the browser, with the results of computations attached to the code which generated them.
- Mix and match languages to suit your needs (e.g. scikit-learn + ggplot2).

Runs as a local web server →

Load/save/manage notebooks →

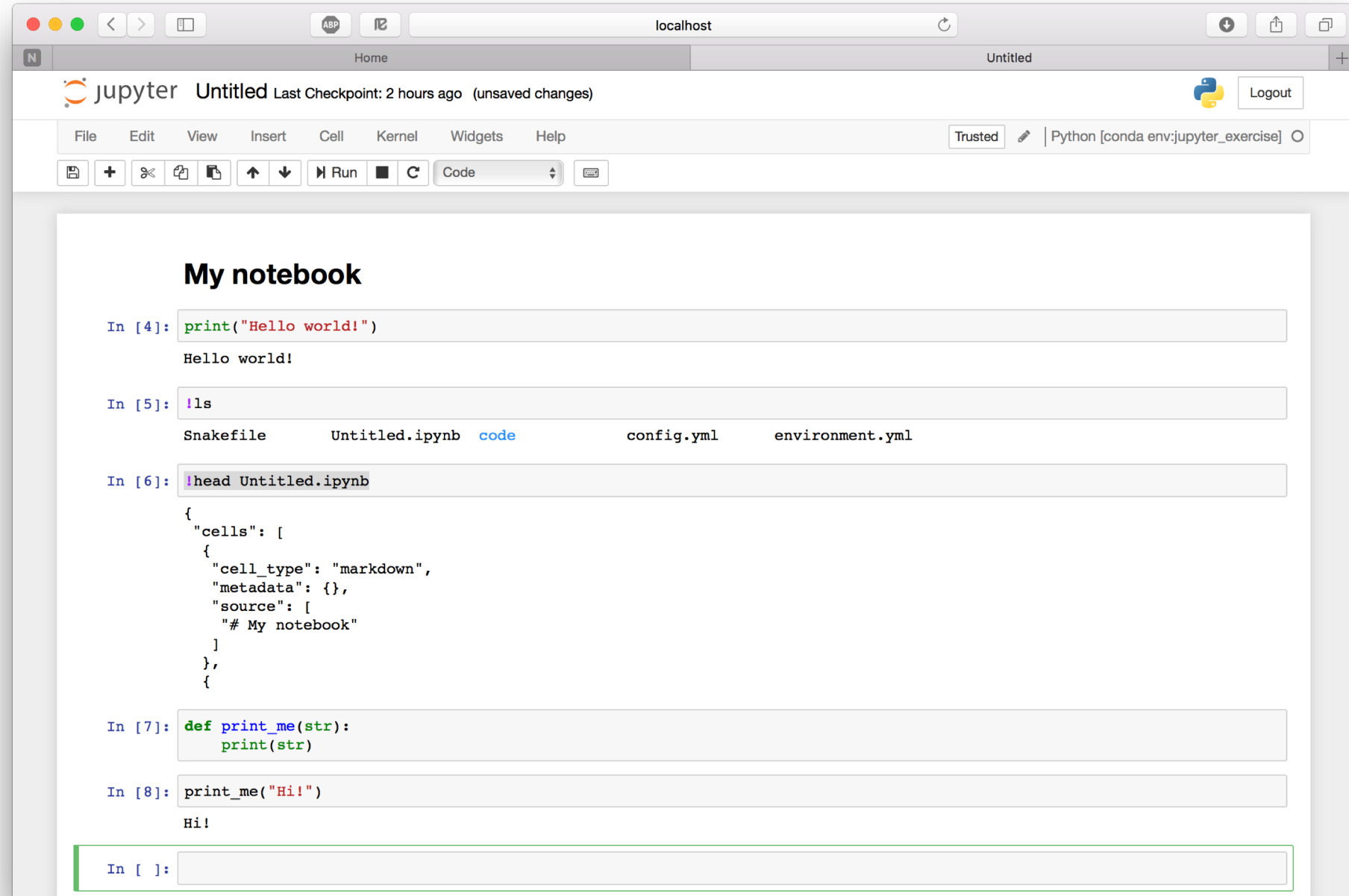
Markdown cell with a header →

Code cell with some Python code →

Run shell command to list files →

The notebook itself is a JSON file →

You can define and call functions →



```
In [4]: print("Hello world!")
Hello world!

In [5]: !ls
Snakefile      Untitled.ipynb  code            config.yml      environment.yml

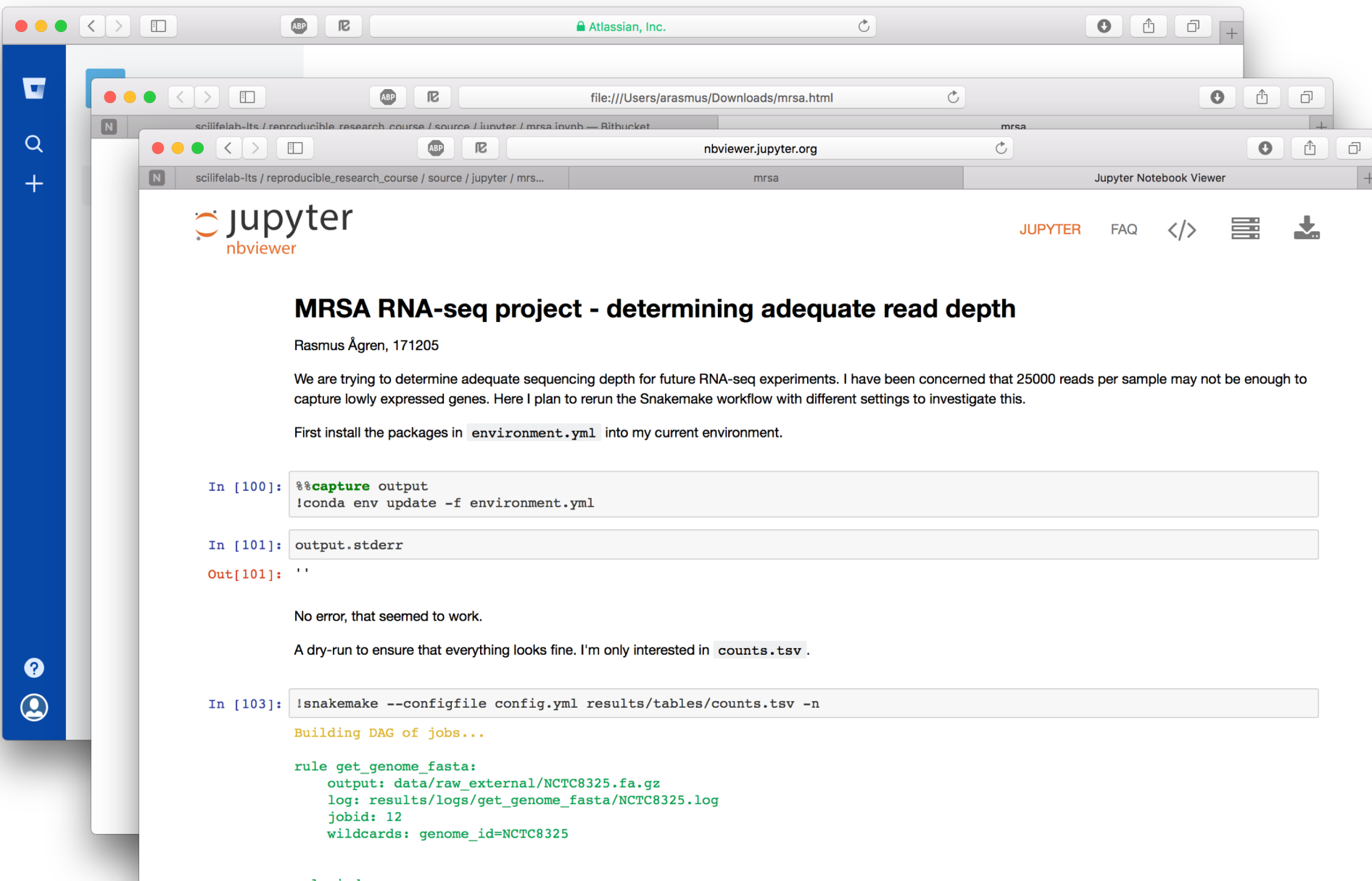
In [6]: !head Untitled.ipynb
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {},
      "source": [
        "# My notebook"
      ]
    },
    {
      "cell_type": "code",
      "metadata": {},
      "source": [
        "In [4]: print('Hello world!')\n",
        "Hello world!\n",
        "In [5]: !ls\n",
        "Snakefile      Untitled.ipynb  code            config.yml      environment.yml\n",
        "In [6]: !head Untitled.ipynb\n",
        "{\n",
        "  \"cells\": [\n",
        "    {\n",
        "      \"cell_type\": \"markdown\", \n",
        "      \"metadata\": {}, \n",
        "      \"source\": [\n",
        "        \"# My notebook\"\n",
        "      ]\n",
        "    },\n",
        "    {\n",
        "      \"cell_type\": \"code\", \n",
        "      \"metadata\": {}, \n",
        "      \"source\": [\n",
        "        \"In [7]: def print_me(str):\\n\",
        "        \"        print(str)\\n\",
        "        \"In [8]: print_me('Hi!')\\n\",
        "        \"        \\n\",
        "        \"In [ ]: \"\n",
        "      ]\n",
        "    }\n",
        "  ],\n",
        "  \"metadata\": {\n",
        "    \"kernelspec\": {\n",
        "      \"display_name\": \"Python 3\", \n",
        "      \"language\": \"python\", \n",
        "      \"name\": \"python3\"\n",
        "    },\n",
        "    \"language_info\": {\n",
        "      \"name\": \"python\", \n",
        "      \"version\": \"3.7.4\"\n",
        "    }\n",
        "  }\n",
        "}
```

Sharing is caring

Put the notebook on GitHub/Bitbucket and it will be rendered there..

.. or export to one of many different formats, including HTML and PDF ..

.. or paste a link to any Jupyter notebook at nbviewer.jupyter.org and they will render it for you.



The screenshot shows a web browser window displaying a Jupyter Notebook Viewer. The browser's address bar shows the URL `nbviewer.jupyter.org`. The notebook's title is "MRSA RNA-seq project - determining adequate read depth" by Rasmus Ågren, 171205. The notebook content includes a text block explaining the project's goal and a code block with three IPython input/output pairs.

MRSA RNA-seq project - determining adequate read depth
 Rasmus Ågren, 171205

We are trying to determine adequate sequencing depth for future RNA-seq experiments. I have been concerned that 25000 reads per sample may not be enough to capture lowly expressed genes. Here I plan to rerun the Snakemake workflow with different settings to investigate this.

First install the packages in `environment.yml` into my current environment.

```
In [100]: %%capture output
          !conda env update -f environment.yml
```

```
In [101]: output.stderr
```

```
Out[101]: ''
```

No error, that seemed to work.

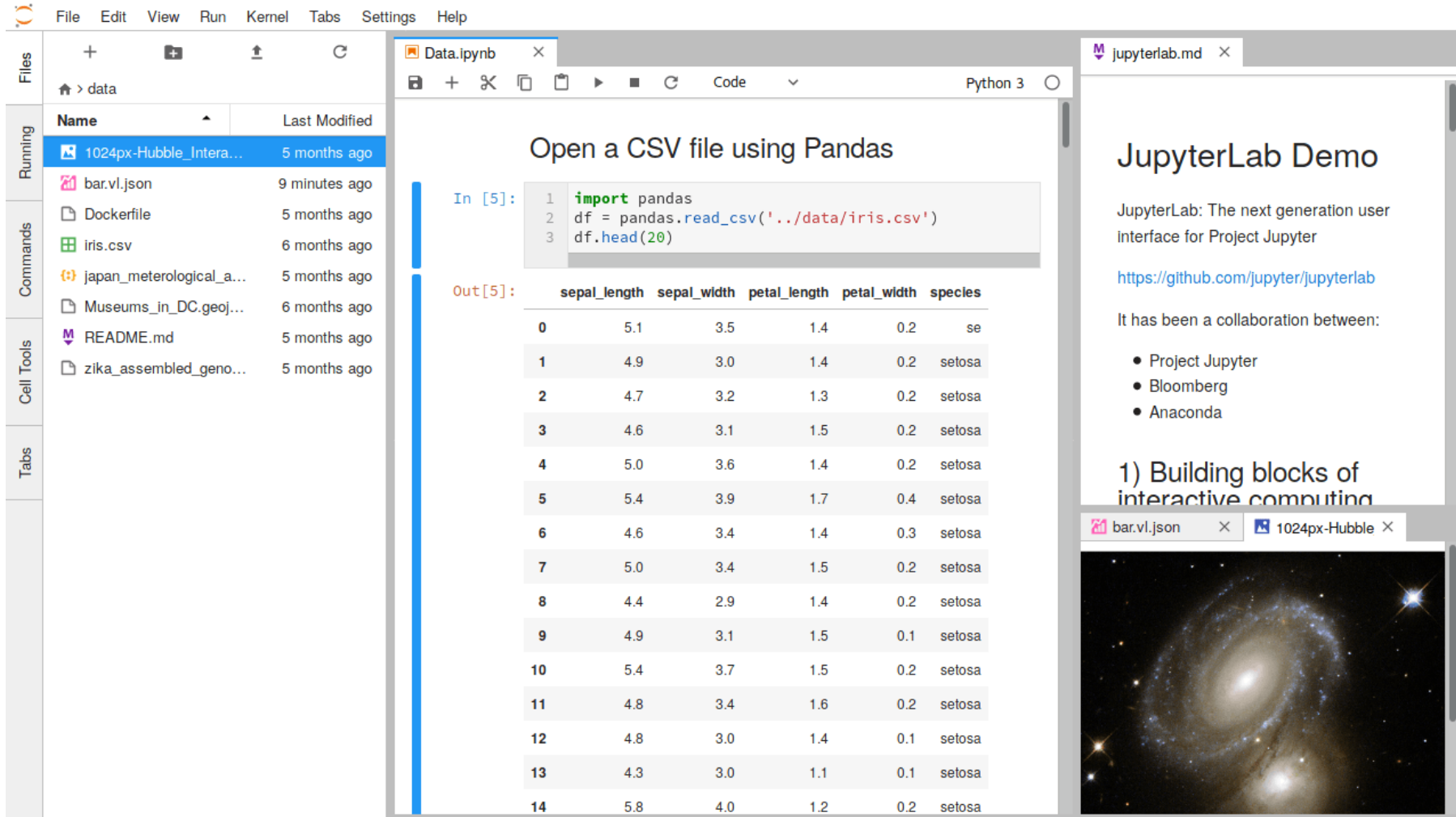
A dry-run to ensure that everything looks fine. I'm only interested in `counts.tsv`.

```
In [103]: !snakemake --configfile config.yml results/tables/counts.tsv -n
```

Building DAG of jobs...

```
rule get_genome_fasta:
  output: data/raw_external/NCTC8325.fa.gz
  log: results/logs/get_genome_fasta/NCTC8325.log
  jobid: 12
  wildcards: genome_id=NCTC8325
```

JupyterLab



The screenshot displays the JupyterLab environment. On the left is a file browser showing a directory named 'data' containing files like '1024px-Hubble_Inter...', 'bar.vl.json', 'Dockerfile', 'iris.csv', 'japan_meterological_a...', 'Museums_in_DC.geoj...', 'README.md', and 'zika_assembled_geno...'. The main area is a code editor titled 'Data.ipynb' with a Python 3 kernel. It contains a code cell with the following code:

```
1 import pandas
2 df = pandas.read_csv('../data/iris.csv')
3 df.head(20)
```

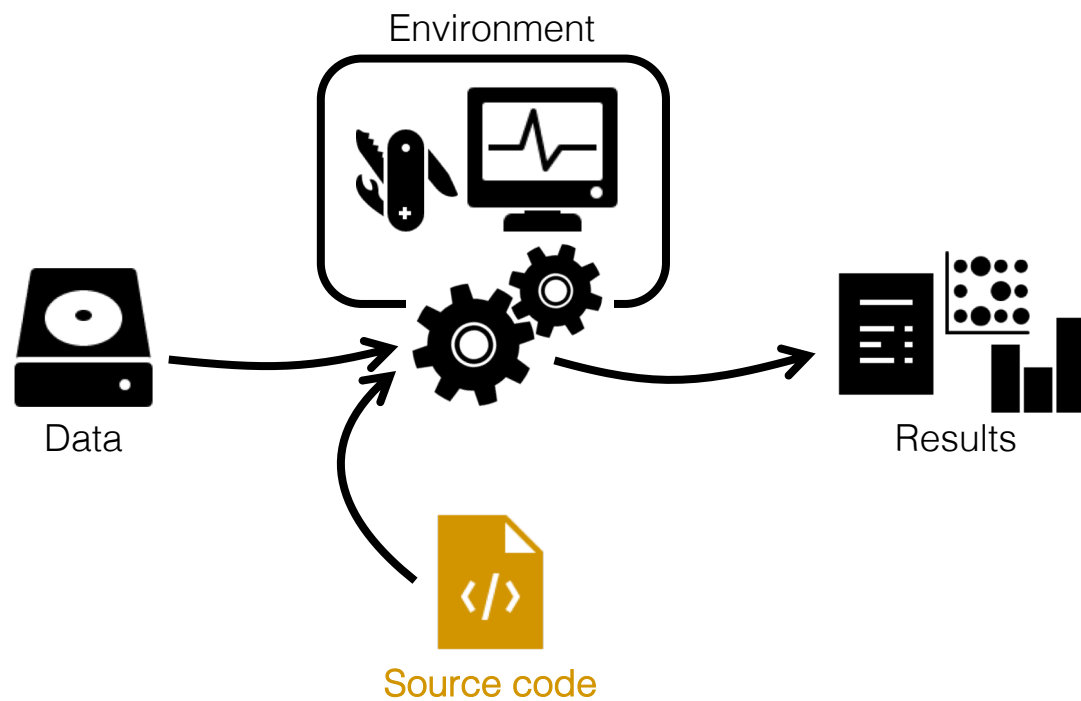
The output of the code cell is a table showing the first 15 rows of the iris dataset:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	se
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3.0	1.4	0.1	setosa
13	4.3	3.0	1.1	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa

On the right side of the interface is a sidebar titled 'JupyterLab Demo'. It contains the text: 'JupyterLab: The next generation user interface for Project Jupyter', a link to <https://github.com/jupyter/jupyterlab>, and a list of collaborators: Project Jupyter, Bloomberg, and Anaconda. Below this is a section titled '1) Building blocks of interactive computing' which includes a thumbnail image of a spiral galaxy.

JupyterLab is a full-fledged IDE, similar to e.g. Rstudio.

```
conda install -c conda-forge jupyterlab
```

```
project
|- doc/
|
|- data/
|   |- raw_external/
|   |- raw_internal/
|   |- meta/
|
|- code/
|- notebooks/
|
|- intermediate/
|- scratch/
|- logs/
|
|- results/
|   |- figures/
|   |- tables/
|   |- reports/
|
|- Snakefile
|- config.yml
|- environment.yml
|- Dockerfile
```