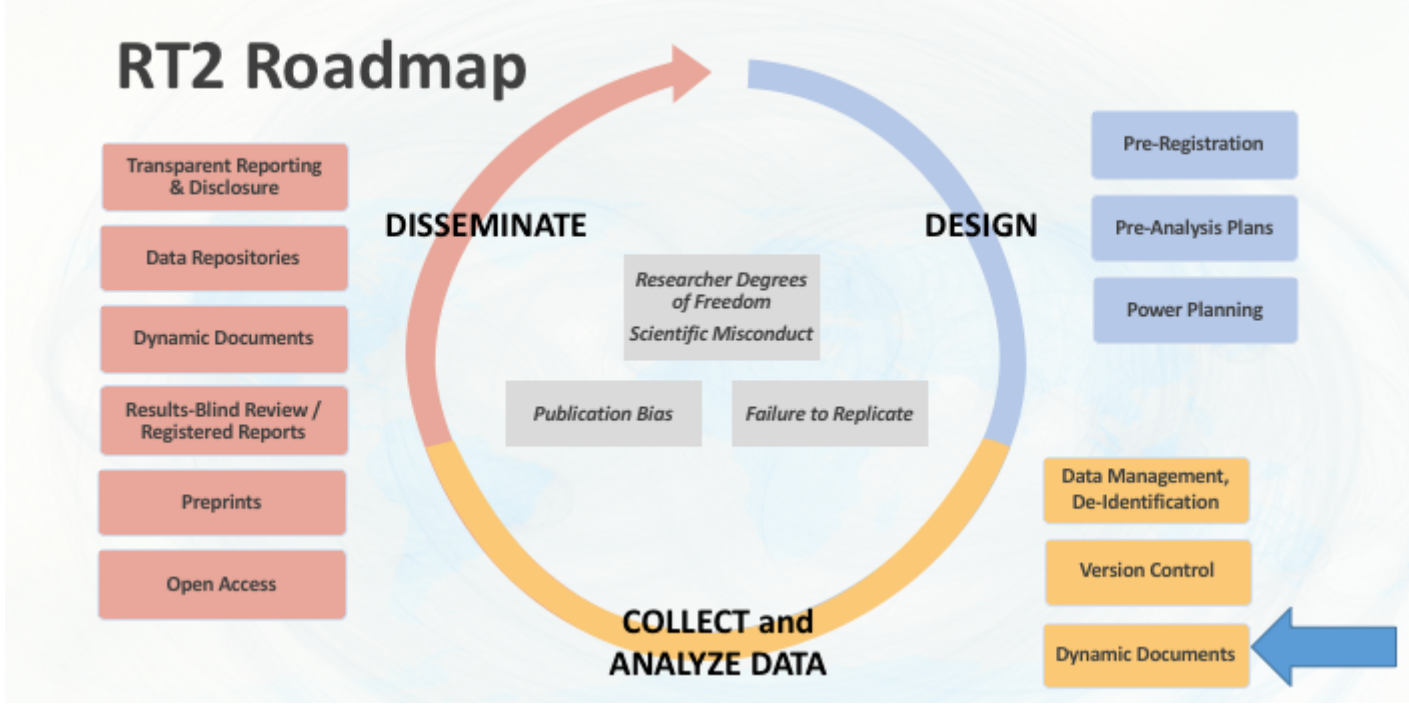
Jupyter Notebooks: Hands-on

Tim Dennis, UCLA/Library Data Science Center: I support researchers engaged in data-intensive scholarship primarily in the social and behavioral sciences


RT2 Roadmap
Transparent Reporting & Disclosure
Data Repositories
Dynamic Documents
Results-Blind Review / Registered Reports
Preprints
Open Access
DISSEMINATE
DESIGN
Pre-Registration
Pre-Analysis Plans
Power Planning
Researcher Degrees of Freedom
Scientific Misconduct
Publication Bias
Failure to Replicate
Data Management, De-Identification
Version Control
Dynamic Documents
COLLECT and ANALYZE DATA

Dynamic Documents

> *The basic idea behind dynamic documents stems from **literate programming**, a programming paradigm conceived by Donald Knuth...A **dynamic document** is not entirely different from a computer program: for a dynamic document, we need to run software packages to compile our ideas (often implemented as source code) into numeric or graphical output, and insert the output into our literal writings (like documentation).*
> *-- Yihui Xie, Dynamic Documents with Rand knitr*

Literate Programming

> *Let us change our traditional attitude to the construction of programs: 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, Literate Programming (1984)*

Dynamic documents are described as:

- Runnable
- Compilable
- Interleaved executable code with explanatory text, mathematics, and rich representations of resulting objects

- How to organize your work?
- How to make work more pleasant for you?
- How to make it navigable by others?
- How to reduce tedium and manual processes?
- How to reduce friction for collaboration?
- How to reduce friction for communication?

Specific tools and habits can build alot of this into the normal coding and analysis process

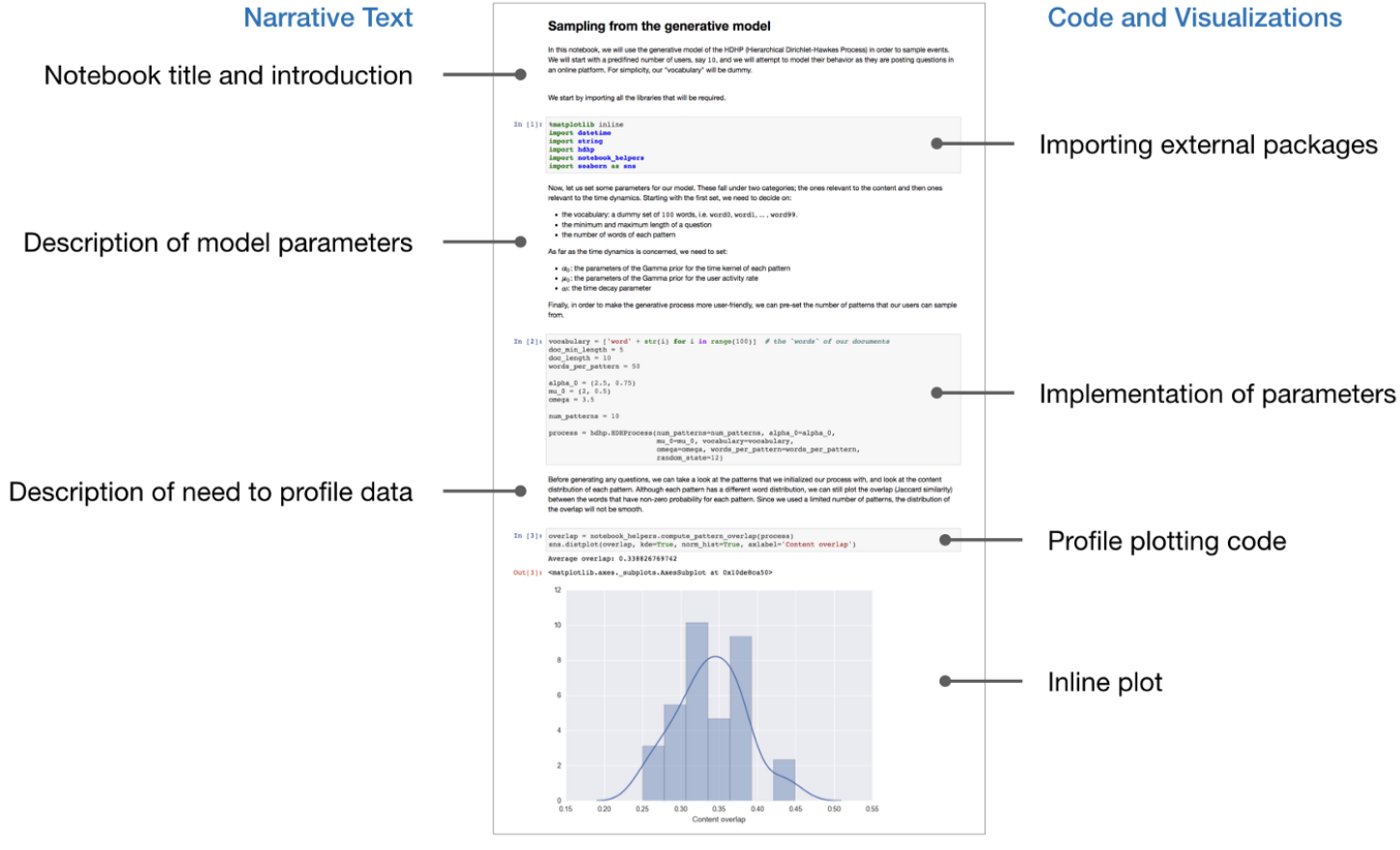
From: Jenny Bryan Reproducible Science Workshop (https://github.com/datacarpentry/rr-literate-programming/blob/gh-pages/slides/lit-prog-slides/lit-prog-slides.pdf)

Getting the analysis right is only one link

Process, packaging, and presentation are often the weak link

From: Jenny Bryan Reproducible Science Workshop (https://github.com/datacarpentry/rr-literate-programming/blob/gh-pages/slides/lit-prog-slides/lit-prog-slides.pdf)

Example from Jupyter



(https://cdn-images-1.medium.com/max/2000/1*dl30JL7jWGPtxuJIuJGGRg.png)

So, what is Jupyter?

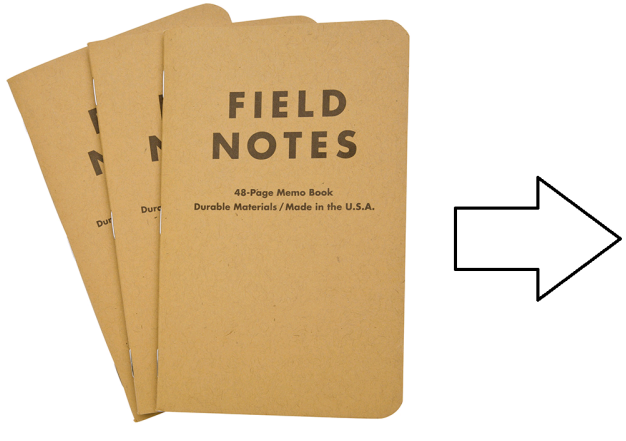
Jupyter notebook, formerly known as IPython (or Interactive Python), is a flexible and powerful open source research tool that can help you keep a narrative of your coding process. The name Jupyter is an acronym of the three core languages it was designed for: **JU**lia, **PYT**hon, and **R**. Project Jupyter supports interactive data science and scientific computing across more than 40 programming languages (https://github.com/jupyter/jupyter/wiki/Jupyter-kernels) (Stata, SAS, etc.).

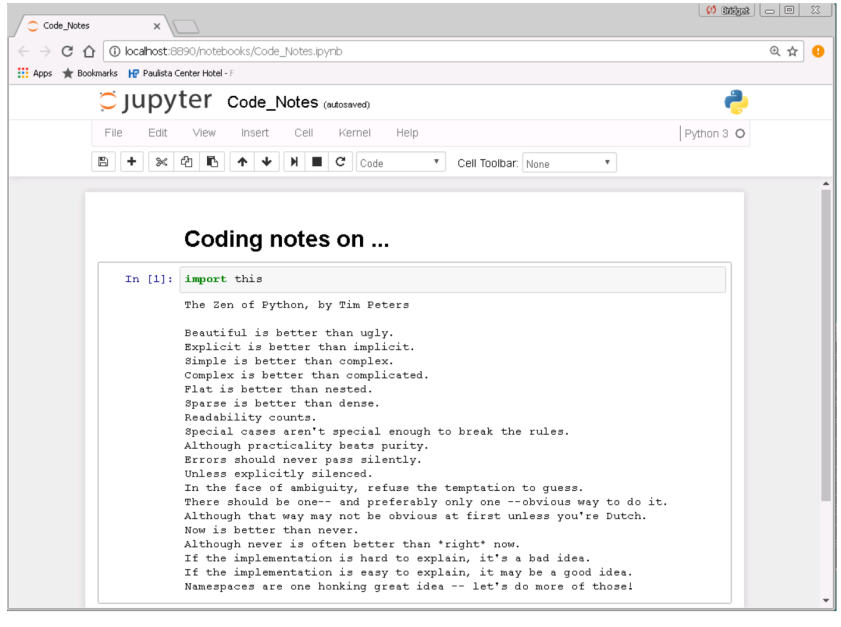
Open source, interactive data science and scientific computing across over 40 programming languages.

https://jupyter.org/

Jupyter as Coding Diary

You can think of the notebook as a lab or field diary that keeps a detailed record of the steps you take as you develop scripts and programming workflows. Just as you would with a field notebook, it is important to develop good note-taking habits. This workshop is designed to impart a set of skills, tools, and best practices you can implement in your own research to enhance reprodubility, which will make modifications, collaboration, and publishing easier.





Jupyter Architecture

Jupyter is comprised of several components, some of which the user doesn't directly interact with, but should at least be aware of. On the **front-end**, the user will work with the:

1. **Web Application:** Browser-based tool for interactive development of notebook documents
2. **Notebook Document:**

    - All content displayed in the application, including inputs and outputs of the code, explanatory text, mathematics, images, and rich media representations of objects.
    - Documents are internally JavaScript Object Notation (JSON) files and are saved with the `.ipynb` extension. Since JSON is a plain text format, they can be version-controlled and shared with colleagues.

This is what `.ipynb` notebooks look like in the file as JSON:

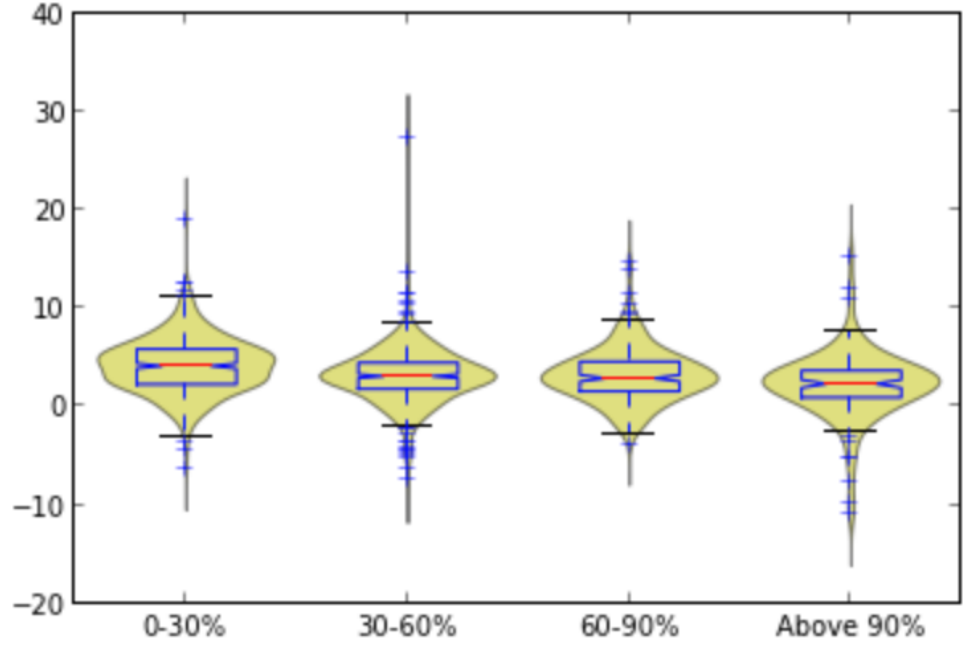
```
- {
    cell_type: "markdown",
    metadata: { },
  - source: [
        "## Categorical scatterplot"
    ]
  },
- {
    cell_type: "code",
    collapsed: false,
  - input: [
        "labels = ["0-30%","30-60%","60-90%","Above 90%"] ",
        "dat = [np.array(RR.dRGDP[RR.dgcat==x]) for x in labels] ",
        "print sm.graphics.violinplot(dat, labels=labels)"
    ],
    language: "python",
    metadata: { },
  - outputs: [
      - {
            output_type: "stream",
            stream: "stdout",
          - text: [
                "Figure(480x320) "
            ]
        },
      - {
            output_type: "display_data",
            png: "iVBORw0KGgoAAAANSUhEUgAAAXQAAAD9CAYAAACsq4z3AAAABHNCSVQICAgIfAhkiAAAAAlwSFlz
            AAALEgAACxIB0t1+/AAAIABJREFUeJzs3XdYVFf++PH3DB2RJjAoiCAoNppIsUXssaMxiRoNyW6y
            2ZqYrJtk96cbXbNJNsluNNlks6n96cbXk96ubxXQZYURFFUREVpEgRkF5n5vz+YCF2ygwMM57X88wj
            zty59zNnZj5z7rmnKIQQAkmSJMnoKQ0dgCRJkqQfMqFLkiSZCJnQJUmSTIRM6JIkSSSZCJnRJkiQT
            IRO6JEmSidA5oWs0GkJCQpg2bRoA5eXlREdHxgYyMyZM6moqNA5SEmSJKl5Oif0lStXMmDAABQK
            BQDLly9n2LBhJCcnExkZyZtvvqlzkJIkSVLzdEro2dnZ7tivqg+eee47G8Ulbt24lJiYGgJiYGDZv
            3qx7lJIkSVKzzHV58808ssvv8x77771HWVlZ0335+fmoVCoV+fn5dz2vsTYvSZIktc6DBve3uYb+
            ww8/4ObmH0PvFAo7pu8hRCd/vbGG28YPAZTusnyl0XZWW/GUpbNaXMN/ciRI2zdupUd03ZQ
            U1NDWVkZCxYsQKVSSkZeXh7u7O7m5ubi5ubX1EJIkSVIrtLmG/tZb5b5GVlUV6ejrr1q1jzJgxfPPN
            N0yfPp3Vq1cDsHr1laqKjo/UWrCRJknR/OrWh36qxaWJkiUswLCAwMBAfH19+eabb/R1iA4XFRVl
            6BBMiixP/ZLqT+mUpYK0ZKGGX0fVKFoUXuQJEmS9JPmcqcKSpJkmQiZEKXJEkyETIhS5IkmQiZ
            0CVJkkyETOiSJEmSJJkIkRClyRJMhEyoUuSJJkIkRJJkImdEmSJBMhEE7okSZKJkAldkiTJRJkImdEmSJBMhE7okSZKJkAldkiTJRMiELkmSZCJnQJUmS
            TIRM6JIkSSZCJnRJkiQTIRO6JEmSiZAJXZIkyUTIhC5JkmQiZEKXOszhw4eJi4szdBiSZLJkQpc6
            ...
```

This is what it looks like rendered in a notebook:

Categorical scatterplot

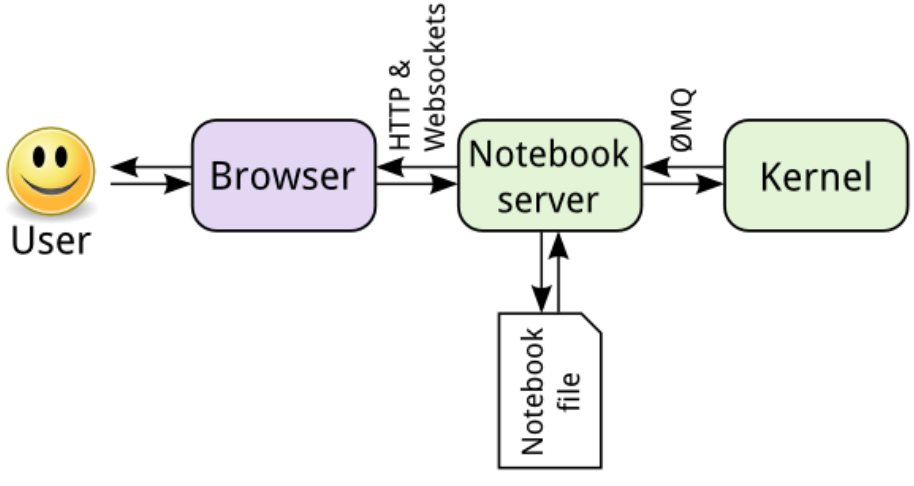
In [23]:
```
labels = ["0-30%","30-60%","60-90%","Above 90%"]
dat = [np.array(RR.dRGDP[RR.dgcat==x]) for x in labels]
print sm.graphics.violinplot(dat, labels=labels)
```

```
Figure(480x320)
```



Jupyter also has some **back-end** processes, including the:

1. **Kernel:** A separate process responsible for running user code. For the purposes of this workshop, we will be working on Python kernels, although Jupyter is capable of interfacing with other programming languages as well.
2. **Notebook Server:** Communicates with kernel and routes the Python programming language to the web browser.



Curious to learn more? Refer to these Jupyter documents:

http://jupyter-notebook.readthedocs.io/en/latest/notebook.html (http://jupyter-notebook.readthedocs.io/en/latest/notebook.html)
http://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

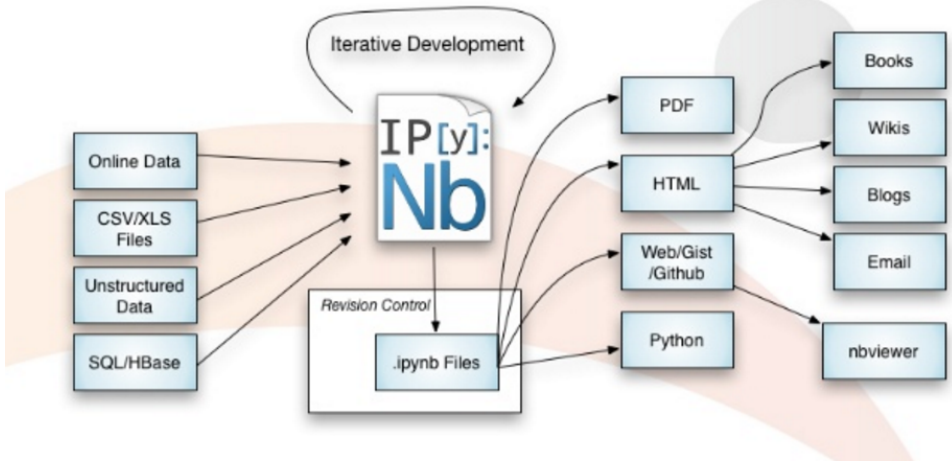
Why are Jupyter Notebooks a powerful tool for conducting reproducible research?

Jupyter Notebooks are great because they **facilitate**:

- **Documentation and literate programming** by combining rich-text narrative *concepts* & machine-readable *code*. The notebeook itself is a data-structure with metadata that can be easily read and parsed.
- **Exploration & development**: Intermediate steps are saved in a clean, well documented format
- **Communication/Collaboration**: sharing research with peers, collaborators, reviewers, public
- **Publishing**: It is simple and quick switch between the development & publishing stage

Jupyter Ecosystem

There has been considerable development by both Project Jupyter and external collaborators that have yielded a multitude of options for Jupyter users. This diagram gives a sample of some of the possibilities.



Jupyter Notebooks Summary

- In-browser editing for code
- Executes code from the browser, with the results of code following the code cell that generates it Displays results of computation using outputs, such as HTML, LaTeX, PNG, SVG, etc.
- In-browser editing for rich text using the Markdown markup language that provides commentary for the code
- Can include mathematical notation within markdown cells using LaTeX, and rendered natively by MathJax

Project Jupyter Resources

- Project Jupyter Homepage (http://jupyter.org/)
- Project Jupyter Google group (https://groups.google.com/forum/#!forum/jupyter)
- Jupyter documentation (https://jupyter.readthedocs.io/en/latest/)
- GitHub (https://github.com/jupyter/help)
- Free Project Jupyter tutorials:
  - Readthedocs (https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/)
  - YouTube (https://www.youtube.com/watch?v=Rc4JQWowG5I)
- List of interesing Jupyter Notebooks (https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks)

Markdown

Markdown

- **Markdown** is a particular type of markup language. Markup languages are designed to produce documents from plain text.
- You may be familiar with **LaTeX**, another (though less human friendly) text markup language.
- Tools render markdown to different formats (for example, HTML/pdf/Word).
    - The main tool for rendering Markdown is pandoc (http://pandoc.org/).

This section adapted from: Carson Sievert's markdown slides (http://cpsievert.github.io/slides/markdown/#/1)

Markdown enables fast publication to the web

- **Markdown** Easy to write and read in an editor.
- **HTML** Easy to publish and read on web.

````
# Title (header 1, actually)

This is a Markdown document.

## Medium header (header 2, actually)

It's easy to do *italics* or __make things bold__.

> All models are wrong, but some are useful. An
approximate answer to the right problem is worth a
good deal more than an exact answer to an
approximate problem.

```
x <- 3 * 4
```

I can haz equations. Inline equations, such as ...
the average is computed as $\frac{1}{n} \sum_{i=1}
^{n} x_{i}$. Or display equations like this:

$$
\begin{equation*}
|x|=
\begin{cases} x & \text{if $x≥0$,} \\\\
-x &\text{if $x\le 0$.}
\end{cases}
\end{equation*}
$$
````

Title (header 1, actually)

This is a Markdown document.

Medium header (header 2, actually)

It's easy to do *italics* or **make things bold**.

> *All models are wrong, but some are useful. An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem.*

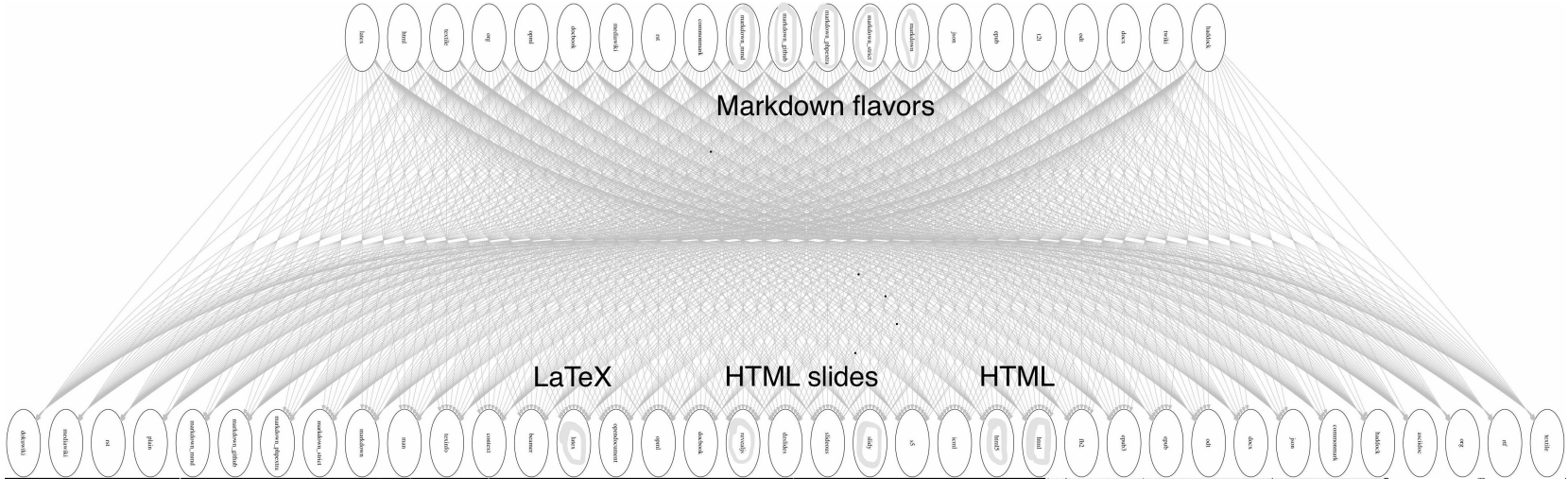
```
x <- 3 * 4
```

I can haz equations. Inline equations, such as ... the average is computed as $\frac{1}{n}\sum_{i=1}^{n} x_i$. Or display equations like this:

$$|x| = \begin{cases} x & \text{if } x \geq 0, \\ \\ -x & \text{if } x \leq 0. \end{cases}$$

Markdown can be rendered to multiple formats

- **pandoc** is a Swiss-army knife tool for conversion`



(https://raw.githubusercontent.com/datacarpentry/rr-literate-programming/gh-pages/media/pandoc-diagram.jpg)

Why it is great:

- Easy to learn and use
- Focus on **content**, rather than **coding** and **debugging errors**.
- It's flexible. Markdown was created to simplify HTML, but with the right tools, your Markdown files can easily be converted to many different formats!
- Once you get the basics down, you can get fancy with `HTML`, `CSS` and `JavaScript`

Today: Hands-on with Jupyter Notebooks

- You need Anaconda Python to follow along https://www.anaconda.com/download (https://www.anaconda.com/download)
- We'll use this Etherpad to share instructions, notes, data and links: https://pad.carpentries.org/rt2-dynamic-jupyter (https://pad.carpentries.org/rt2-dynamic-jupyter)
    - Please open that page up!