Modern tools for ("automated") analysis of lattice data: A case study

William I Jay — University of Colorado Boulder
Lattice 2017 — Granada

In collaboration with
Venkitesh Ayyar — University of Colorado Boulder
Daniel Hackett — University of Colorado Boulder
Ethan Neil — University of Colorado Boulder, RIKEN-BNL

Goals & disclaimers

- This talk does:
    - Describe our solutions to some practical data science problems in lattice gauge theory
    - List tools we have found useful in practice
        - **Not** worry too much about the underlying theoretical computer / data science
        - **Not** pass value judgements on your existing science workflow. (But we do think our techniques can help make the most out of limited research hours)
        - **Not** claim originality. Most of it is research-by-google-search and digesting responses on StackExchange / StackOverflow. Wikipedia is also great.
- This talk is a “success” if it helps you save time or conduct your analyses more efficiently

Background & motivation

- Lattice gauge theory presents many challenges in data science
- The sheer volume of data can be a big difficulty
    - In our ongoing SU(4) project, our full dataset contains O(900) ensembles and O(50,000) gauge configurations
        - A variety of measurements on each configuration…
- Large datasets present two basic challenges
    1. How to generate the data efficiently?

        ☞See the previous talk by Daniel Hackett

    2. How to manage, record, and “automate” the analysis of the data?

        ☞This talk

The story so far…

Automated Data Pipeline

"I wonder what happens at $(\beta, \kappa_4, \kappa_6)$?"

Automated!

- Launch HMC stream
- Make measurements on new gauge configurations
- Parse data and sync to database
- Perform "bulk analysis" of data, e.g.:
  - Determine phases of ensembles (various diagnostics)
  - Compute meson masses, quark masses, decay constants
    - (Grid over various fit parameters, e.g.: t_{min}, t_{max})
- Compile summary/"digest" of data for all ensembles
- Look at observables at $(\beta, \kappa_4, \kappa_6)$

~~This talk~~ → Previous talk by Dan Hackett

→ This talk, but at the schematic level

Modern tools

Three tools have been key in our recent projects:

1. **Relational databases** for storing lattice results

2. **A notebook environment** for developing and automating analyses

3. **Fast open-source software** as the glue linking the database and the notebook

Databases

Motivation

- In the past my lattice “databases” have often been

    A. Files with a set of naming and output conventions

    B. Scripts to extract data from files using `awk` / `grep` / `sed`

- (At least) three difficulties:

    1. This approach has a *de facto* preoccupation with details of storing and accessing the data

    2. Software aging occurs when naming conventions drift over the course of a project, making analyses hard to maintain

    3. Differing conventions among collaborators complicates sharing data

- Solution: Standardize the storage and querying routines

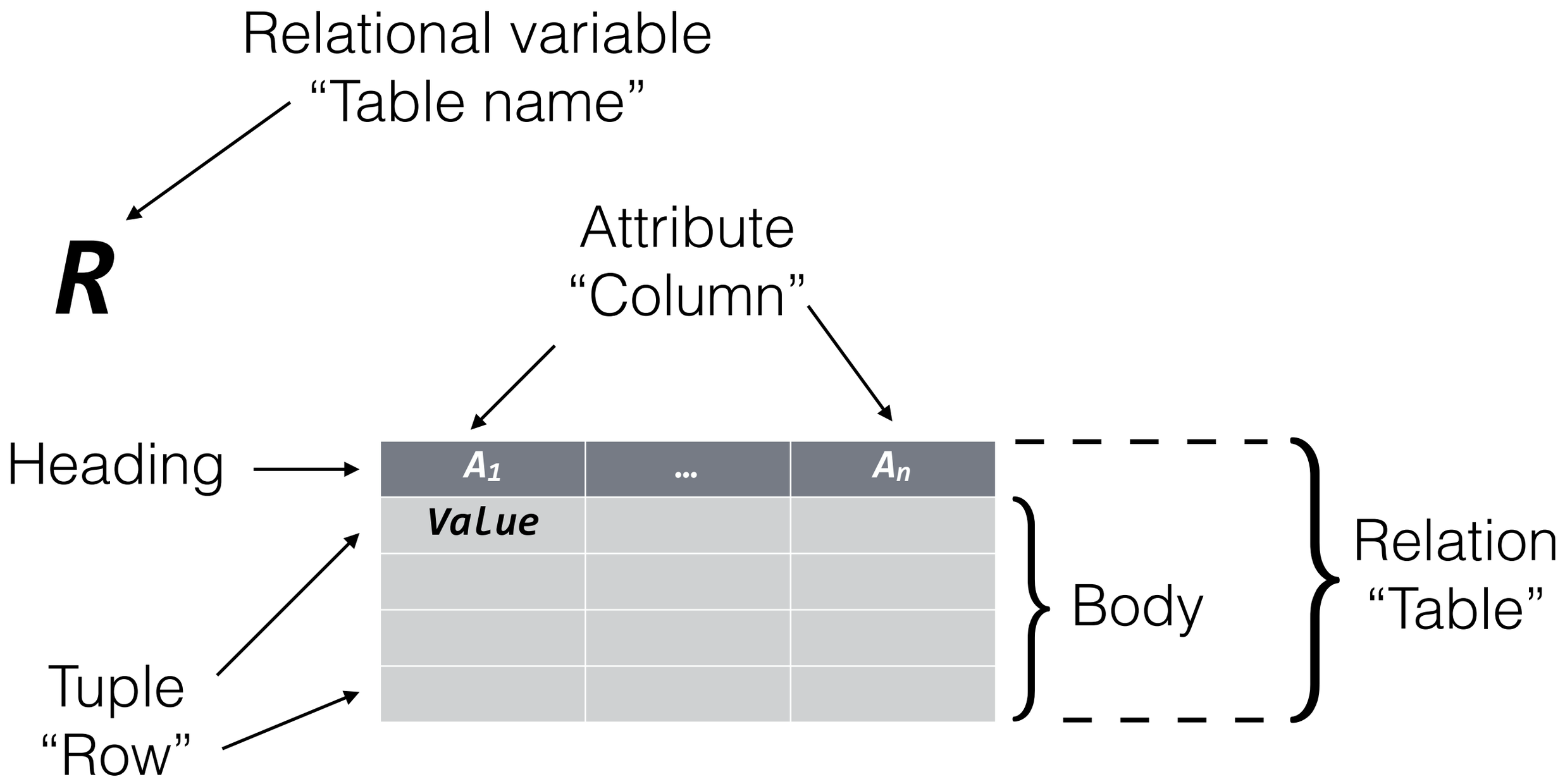
Databases

The relational model of data and SQL

- The relational model is an abstract model of data used in theoretical computer science.
- Structured Query Language = SQL [read: /ˈɛs kjuː ˈɛl/ or /ˈsiːkwəl/]
  - A language which (mostly) realizes the abstract relational model
  - Standardized (ANSI / ISO) and concrete
  - Easy to learn, read, and write
- Using a robust abstract model of data lets you:
  - Focus on the information itself and the relationships that exist within it
  - Specify which information you want out at a given time
  - Outsource the construction and management of data structures and querying routines
- We use PostgreSQL — a particular free and open-source implementation of SQL
- Critically, relational databases are well-adapted to typical lattice data

Relational Database:

A collection “relations” = tables with (unordered) columns and rows.
Tables can reference other tables.



Relational Databases

A natural framework for lattice data

ensemble

ID	NS	NT	BETA	K
1	16	32	6.0	0.1280
2	16	32	6.2	0.1290
3	16	32	5.9	0.1295
4	24	18	6.1	0.1301

gauge_config

ID	ENS_ID	STORED	TRAJ
75	1	True	1
86	1	False	2
93	1	False	3
104	1	False	4

```
ens_id integer REFERENCES ensemble (id)
```

result_gauge

ID	GAUGE_ID	PLAQ	POLY_LOOP_RE	POLY_LOOP_IM
12	75	#	#	#
16	86	#	#	#
35	93	#	#	#
72	104	#	#	#

```
gauge_id integer REFERENCES gauge_config (id)
```


+ many others…

Relational Databases

A natural framework for lattice data

Example: selecting the plaquette for trajectories above some “thermalization cut”

```
SELECT traj, plaq FROM result_gauge

JOIN gauge_config ON (gauge_config.id = result_gauge.gauge_id)

WHERE (traj>100)
```

traj	plaq
100	#
101	#
102	#
…	…

Relational Databases

A natural framework for lattice data

- A lattice database is generally useful, particularly beyond the toy examples in this talk based on HMC

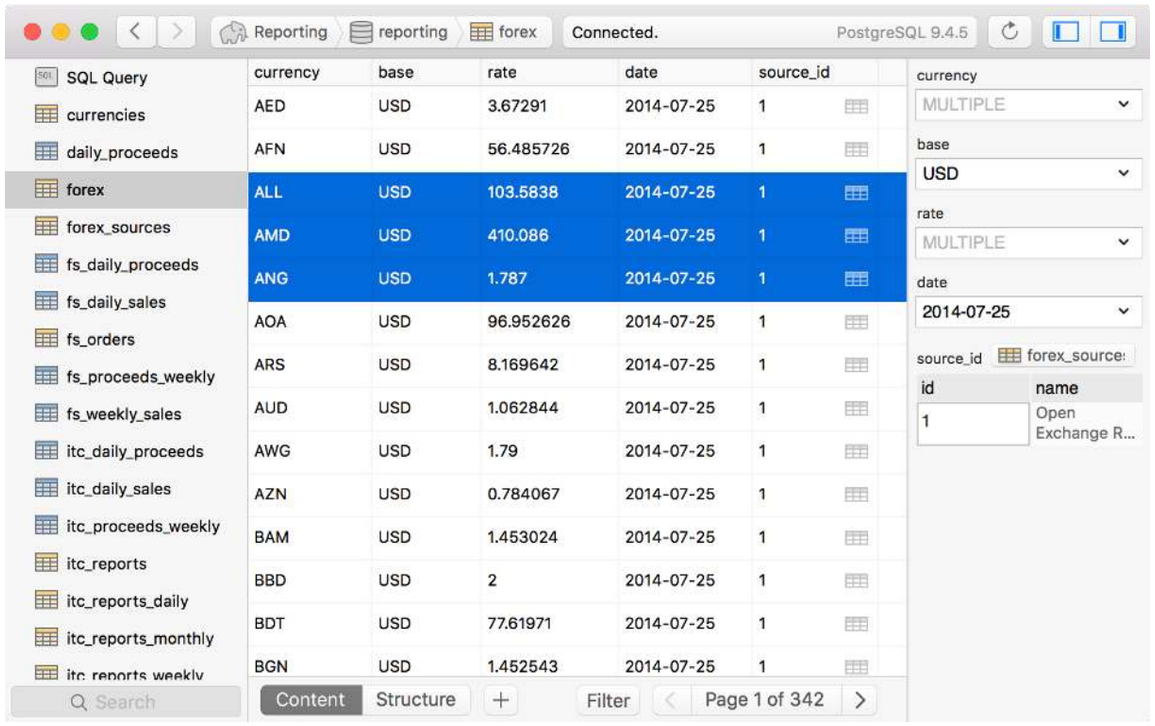
- Our working database is reasonably modular and stores correlation functions, fit results, jackknife / bootstrap results, systematic errors, and much more

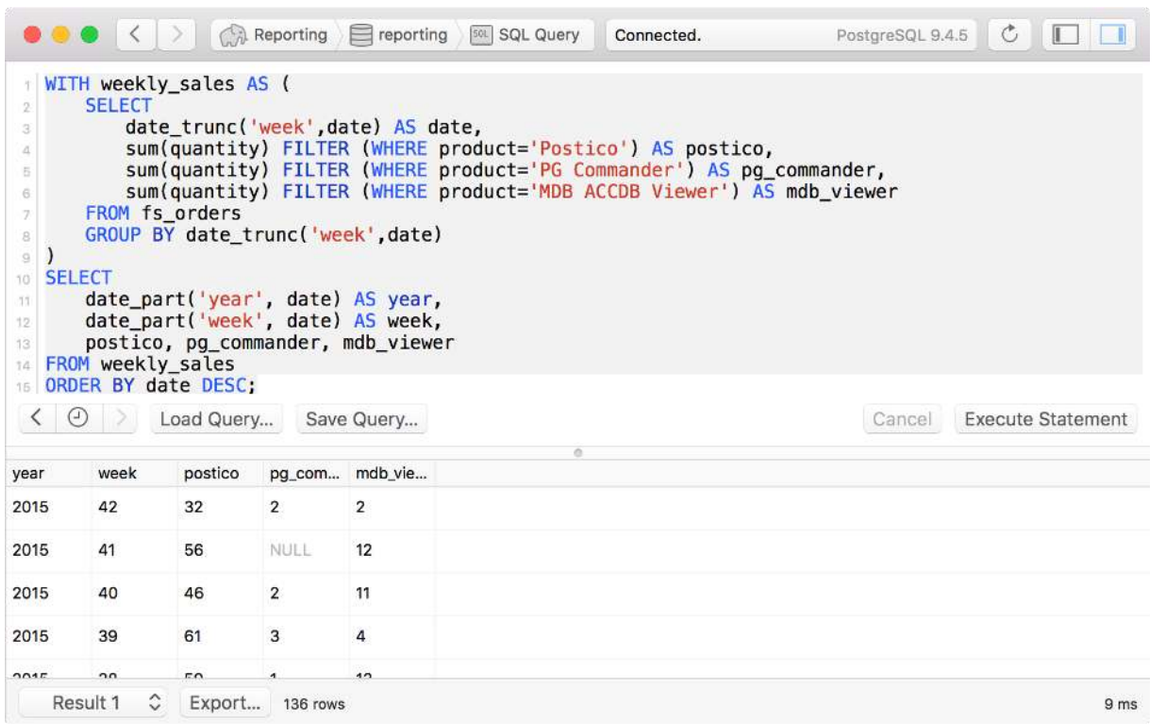
- Practical examples are still easy, but slightly too large for a 15 minute talk — stay tuned for minimal working examples on our GitHub page (and I'm taking requests!)

Relational Databases

Some useful tools

- GUIs are useful for exploring and debugging your database. We've had good luck with **Postico** (free indefinite trial) and **DBeaver** (free)





Postico screenshots (in a non-lattice setting)

A notebook environment

Motivation

- For me, a typical lattice analysis is a problem in data science combining many different tools
    - File or database I/O
    - Data wrangling
    - Storing and piping intermediate results
    - Running analysis / fitter packages
    - …
- Often, my own analyses have occurred “by hand” at the command line. This process
    - Becomes error-prone and hard to replicate
    - Makes it slow to iterate ideas
    - Hinders the transmission of new ideas
- Scripting, of course, partially solves the problems but is also not ideal for development.
- Is there a better framework for handling our data science problems?

The Jupyter notebook

What is Jupyter?

- Jupyter is a data science platform ("notebook") living in a web browser (but which runs locally and offline)

- It's like "**`Mathematica`**," but *open source* and with support for > 40 languages including **`JUlia`**, **`PYThon`** and **`R`** (= "JU+PYT+R")

- Notebooks consist of many cells, which can contain text (with Latex support!) or executable code. Variables stay in scope after evaluation.

- Jupyter is the modern incarnation of the iPython notebook, which was originally developed by F. Perez during his PhD in lattice gauge theory around 2001

“Computers are good at consuming, producing and processing data. Humans, on the other hand, process the world through narratives. Thus, in order for data, and the computations that process and visualize that data, to be useful for humans, they must be embedded into a narrative—a computational narrative—that tells a story for a particular audience and context.”

–Fernando Perez and Brian E. Granger in a proposal for Project Jupyter

The Jupyter notebook

Why use Jupyter?

- Scientists tend to reinvent tools like these continually (e.g., “hacks” to keep variables in scope for experimenting, etc…). We’ll get better mileage using an established tool with a large, active, and supportive community.

- The notebook is quick to install and easy to use (and may have come bundled with your python distribution).

- The notebook quickly improved our daily scientific workflow and ability to work collaboratively. We’ve heard good reports from the lattice community.

- Many different modes for usage:

    1. As a notebook in a browser—the default running mode

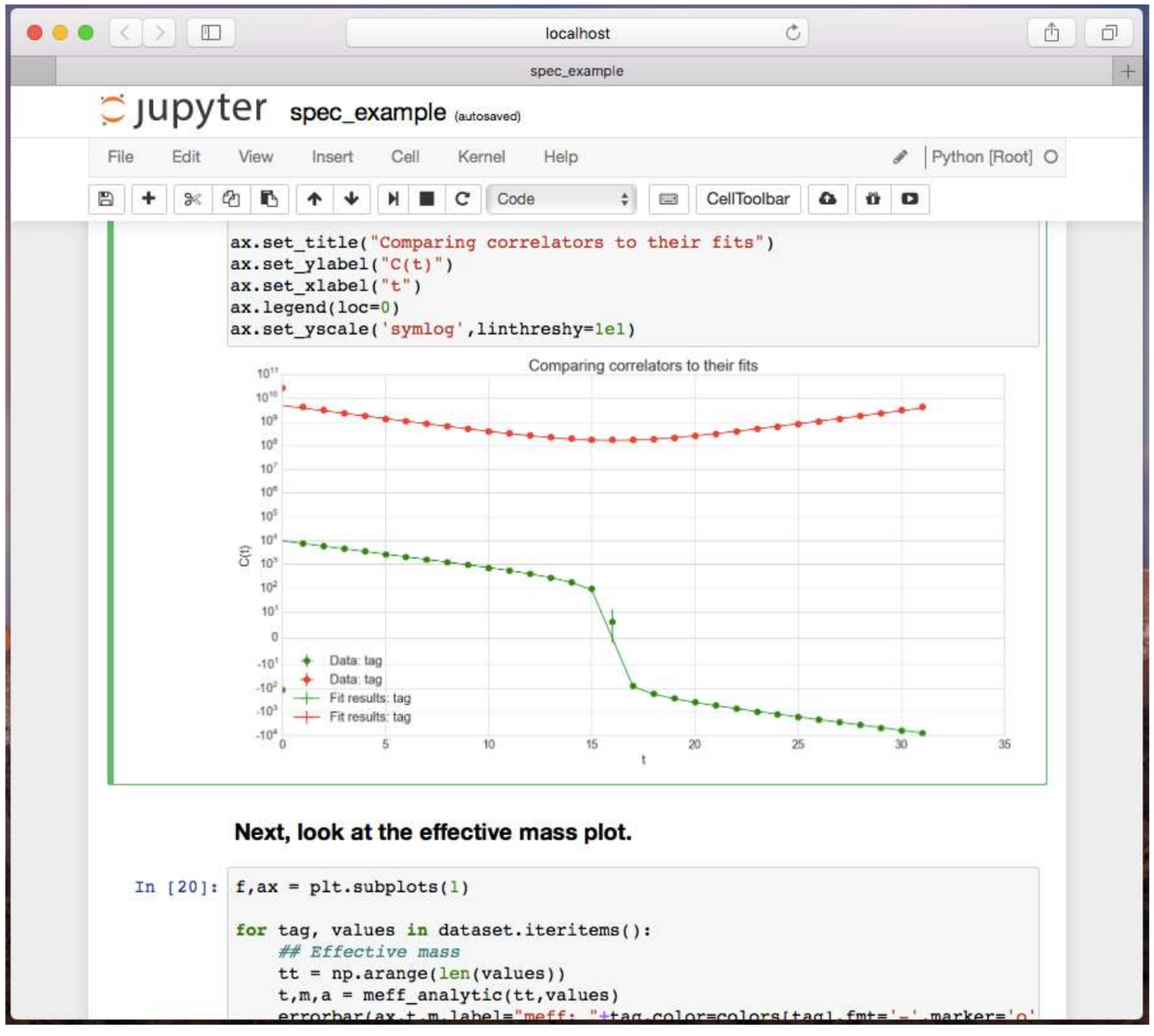
    2. As an executable script from the command line—plots and output saved from last run

    3. As rendered PDF or HTML files—convenient for sharing analyses in a relatively “static” form

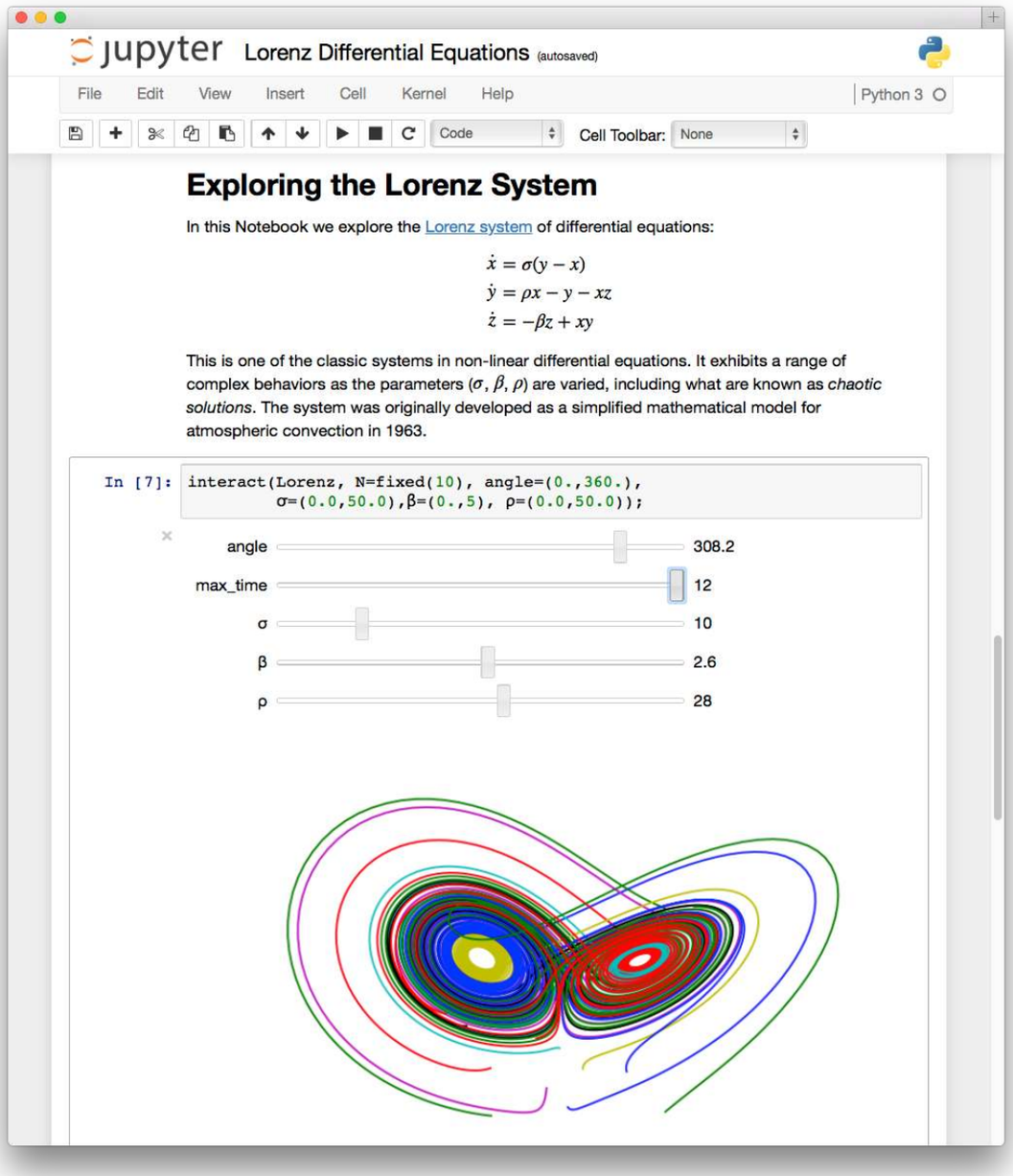
    4. As a “compiled” dashboard for exploring data while minimizing / hiding code

- Notebooks can be run remotely anywhere accessible by `ssh`—convenient when computational resources are distributed, as in many lattice studies

Example notebooks



A lattice notebook I used to share basic lattice spectroscopy methods



Many interesting example notebooks exist online

How do we combine these tools in practice?

Task 1: Interact cleanly with database: SQLAlchemy

- Recall: the whole "point" of using SQL was to standardize interaction with data. It's imperative to avoid re-inventing the wheel.

- Solution: Interact with the database using high level, open source code

- `SQLAlchemy` harnesses the power / flexibility of `SQL` to build, populate, and query the database using the python

  A. Supports concrete usage, with hard-coded queries like the example

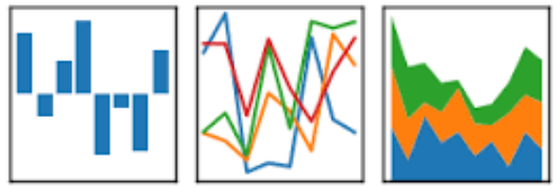
```
SELECT * FROM table1 JOIN table2 ON (table1.id = table2.id)
```

  B. Supports abstract usage, with automatic queries generated and mapped from object-oriented code ⟹ very convenient to combine with common lattice packages, e.g., Lepage's `lsqfit` or `corrfitter`

Task 2: Exploit convenient data structures

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



- Pandas: a Python library for high-performance data structures / data analysis built around "DataFrame" objects
  - Pandas = "**pan**el **da**ta" = econometrics jargon for multidimensional structured datasets
  - Broad support and usage across academic and industrial "data science"
- DataFrames are:
  - Roughly, a python implementation of the "tables" in the relational model.
  - Convenient when paired with an SQL database
  - Even *more* convenient when coupled with SQLAlchemy

```
import pandas as pd
import sqlalchemy as sqla
engine = sqla.build_engine()
dataframe = pd.read_sql_query("select * from table", engine)
```

Take-home points

- SQL databases are a standardized, stable option for storing lattice data. They work in practice.
- The Jupyter notebook is a powerful tool for analyzing lattice data and sharing “computational narratives.”
- Pandas and SQLAlchemy tie these tools together, making each more useful.
- All of these tools can be used as stepping stones toward an “open science” approach to lattice gauge theory
- Code examples are (and will appear) on GitHub
  - Our group page: https://github.com/CULattice
  - My page: https://github.com/willjay

Useful resources

Best practices in computational science can help ensure that results are correct, reproducible, and accessible. For example,

- G. Wilson *et al*. “Best Practices for Scientific Computing” doi.org/10.1371/journal.pbio.1001745 (or arXiv:1210.0530)

- Sandve, Nekrutenko, Taylor, and Hovig. “Ten Simple Rules for Reproducible Computational Research.”doi.org/10.1371/journal.pcbi.1003285