Collaboration between Data Scientists and Software Engineers

Common Issues & Patterns to address them

Where I am coming from

- Talk will be from the POV of a software engineer that has worked closely with data scientists

- Second time here
  - First talk: data science prototype from notebooks to production

Since then, have worked on:

- Data pipelines
- Data APIs
- Development of tools to support data scientists / researchers in asking the right questions
- Evaluation of algorithms from internal/external experts

Putting it in Context

- Collaboration between engineers (with very similar background) is already a big topic
  - version control
  - waterfall, agile etc
  - Specialized frameworks

Collaboration for web development

- Teams of engineers with different skill sets (backend, frontend)
- Collaborate on large scale projects
- Need to work in parallel towards common goal
- Web development becomes mainstream → frameworks and patterns emerge
  - Model-View-Controller
  - Django, Rails
- Patterns inspired by existing ways to develop desktop software (widgets, event-listeners)
  - adapted to changing requirements

Data-driven applications: new big thing

- As they become more mainstream, patterns emerge (an almost evolutionary process)
- Data scientists and software engineers need to work efficiently together
- Element of 'data exploration' without clear specs on outcome changes things
  - Web development does not quite have the same process
  - But more conventional research does

A comparison

	Data Scientist	Software Engineer in Data
Main motivation	Gather novel insights from data	Design and build a robust data management system
Core Competency	Asking the right questions to the data, interpreting the answers	Building & maintaining components like databases, queues, making sure code is production-ready
Reads about	Domain specific research	How data management systems work
Dislikes	Debugging low level errors	One-off work
Appreciates	Complex models that require smarts to formulate and prove	Elegant code, automation
Frequently used tools	Jupyter notebooks, sql, big data frameworks	Command line, big data frameworks

- Not diametrically different
- Just diverging core competencies and interests
- Let's see what issues can arise when they build data-driven applications together

Issue #1: Data Access

- Anti-patterns to watch out for
  - Do the data scientists have to ask somebody for a manual export of data to work on? (too little access)

    OR
  - Do the exploratory queries run on the production database? (too much access)
- Scientists need to be able to work independently, on fresh data and as big datasets as possible without breaking down production

The pattern: Data API & Replicas

- Need for OLTP/OLAP separation has been around since the 90s
- Have a replica of the main database for running analytics queries
- Might make sense to have a different layout than production to better serve the queries
- Build an API for data requiring special access
  (so as not to give blanket permission)

Issue #2: Evaluation

Anti-patterns:

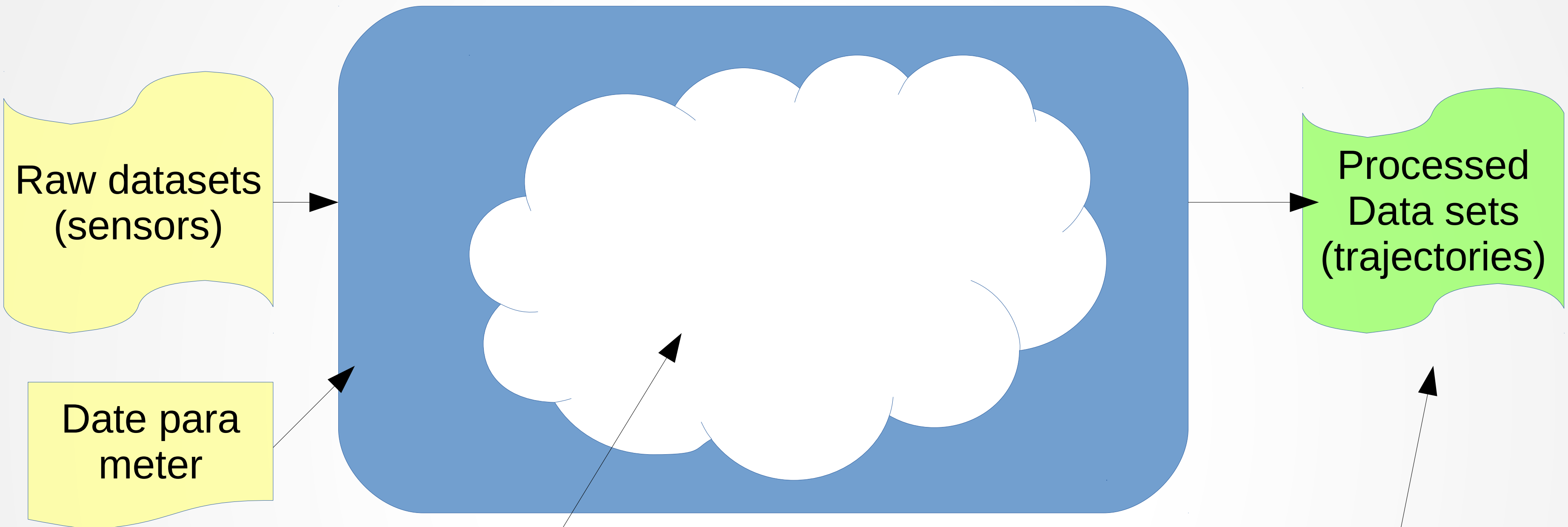
- Is it hard to find out what effect changes in the algorithm have to the resulting data?
- Is it hard to find out which input/commit combination produced a certain result?
- Do the data scientists spend most of their time producing reports for different combinations of inputs/commits?
- Is a lot of time spent to find out why two graphs that were generated "*the same way*" are different?

- Or trying to run the data processing algorithms end to end in a local machine?
- Do errors become apparent late in the process?

The pattern: Continuous Evaluation

- Goal: Have 1-click reports!
  - In controlled environment
  - Efficient, reproducable research
- Reports generated for every commit
  - Regression testing
  - Business logic
  - Exploration
  - etc
- Need to devote time to collect datasets to use for regression testing
  (this is a whole topic in itself)

Specific Use Case



- Two different MATLAB scripts
(the first feeding its output to the second)
- Need to query a database based on the date parameter

Want to create reports on that output (error checks, evaluation etc)

Specific Use Case

- We want to be really **flexible** with what we and how check in the reports
- But also be really sure that we're running the complex pipeline in an organized way
  - Managed environment
  - Stages in pipeline are connected properly (inputs to outputs, no stale data etc)
  - Keeping track of provenance

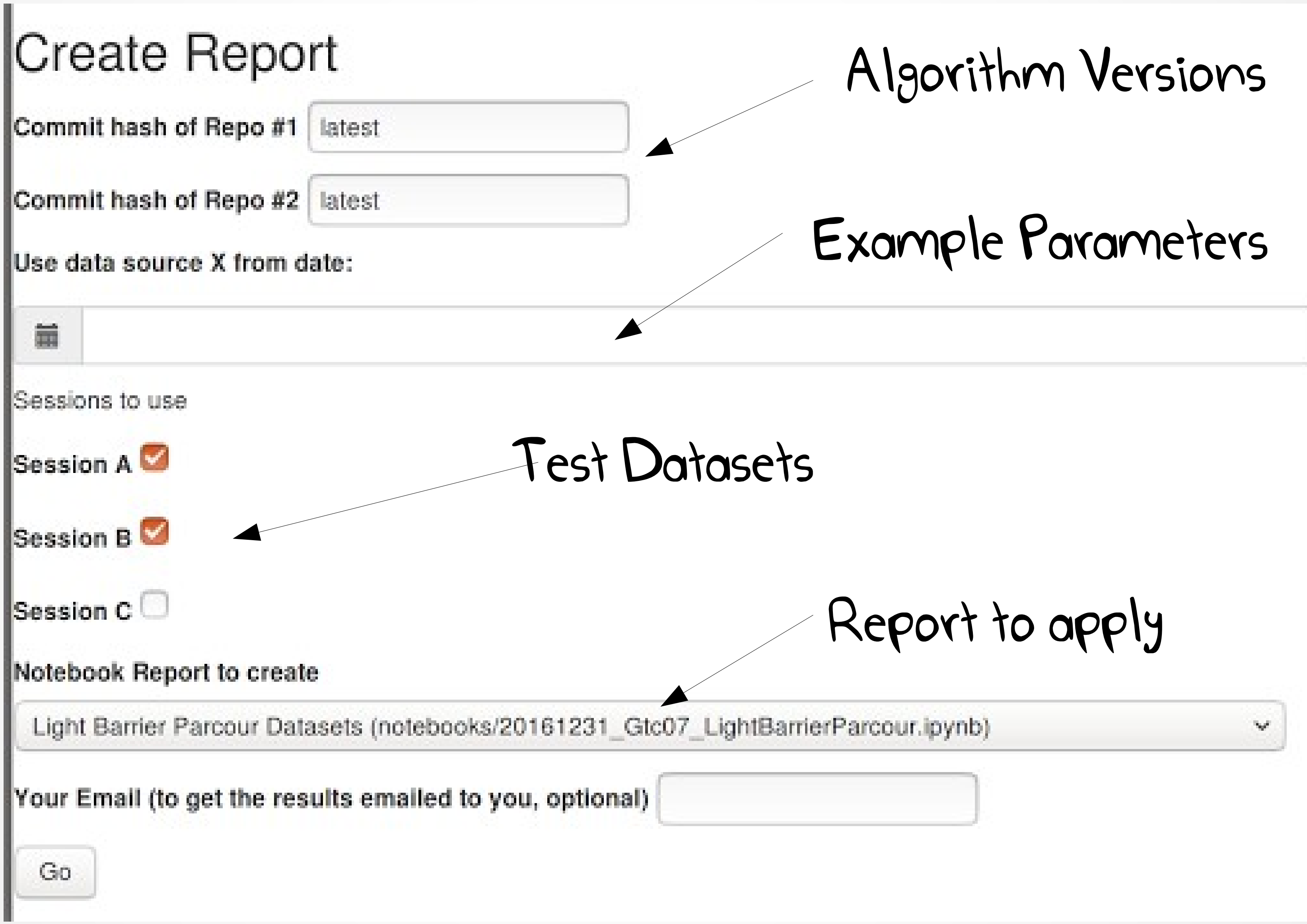
Things that engineers like to do!!

Solution

Things data scientists like to do

- Data scientists write the MATLAB code
- Data scientists create the reports once (as **templates**)
  - A re-imagining of frontend templates from Web Dev world
- Software engineers build the data pipeline
  - out of the components given by the data scientists
  - keeping track of **data provenance**
- Reports can be triggered automatically with each commit or manually with a web interface
- Reports get populated by the data from the data pipeline and it is always clear how they came to be (code + data + parameters)

What it looks like from user side



What it looks like from Data Scientist side

jupyter 20161231_Gtc07_LightBarrierParcour Last Checkpoint: 19 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Python 2

Code CellToolbar

In [4]:
```python
# PROCESSED DATA (variable `input_sessions` gets replaced when notebook is used as template
# with the new paths of results )
# but it can still be used for local and offline development by the data scientist
input_sessions = ["/my_local_machine/carolinux/evaldata/07_lig_bar_par/session_A/",
                  "/my_local_machine/carolinux/evaldata/07_lig_bar_par/session_B/",
                  "/my_local_machine/carolinux/evaldata/07_lig_bar_par/session_C/",]
```

Pathnames
overriden
by data pipeline

Report Logic

In [6]:
```python
sessions = []
for session_file in input_sessions:
    session = load(session_file)
    sessions.append(session)
    # DO something with the session, plots, analysis etc (data scientist domain)
```

Exploration of Processed Data
begins here!

So, can have 'regression test', 'anomaly report', 'sanity check'
etc all developed by data scientists and able to
be run for any parameter which is overriden in the template

Jupyter Notebooks as Templates

- Can generate html from a notebook

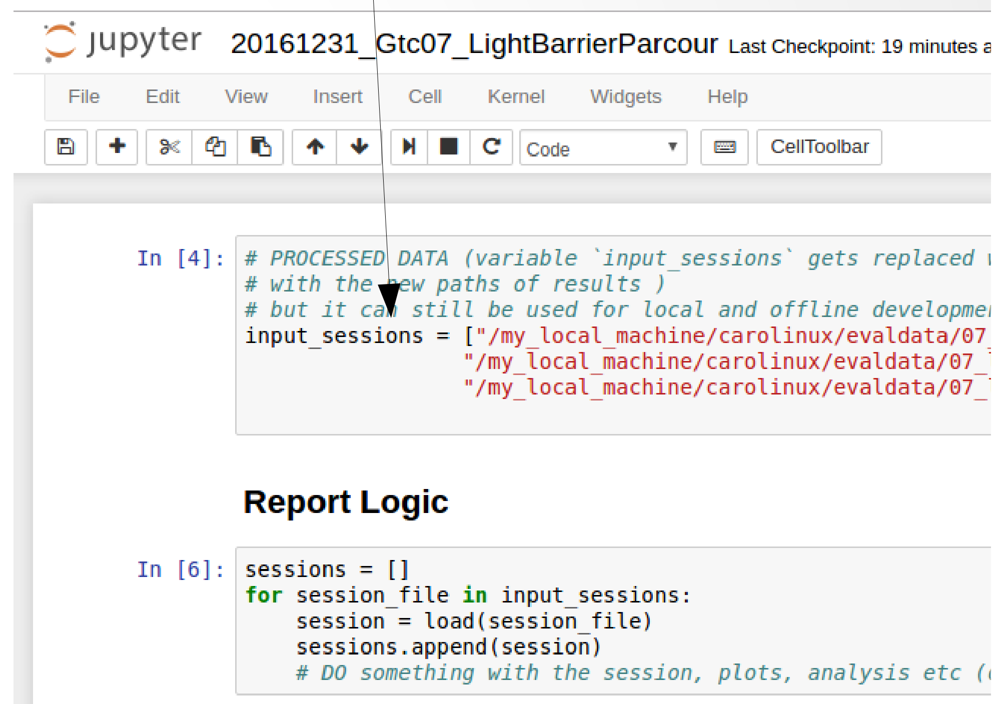
```
jupyter nbconvert --
ExecutePreprocessor.timeout=100000  --output
output_html.html --execute jupyter.ipynb
--to "html"
```

- No canonical way to pass parameters to a notebook (yet)
  - So as to create different html files for different inputs
- A first attempt is here

  https://github.com/takluyver/nbparameterise

  https://github.com/carolinux/nbparameterise
- Can replace variables by name and put any value (simple values, lists, dicts)

We can replace this list with any Python list when generating the notebook



What it looks like from Software Engineer Side

- Flask webserver + celery queue as pipeline system
- For each output create an associated **metadata file** with how it was generated (provenanance)
  - just a hash of significant parameters
- Tasks run according to inputs/parameters and populate jupyter notebook (ie the report template)
  - Could have used a combination of Jenkins + airflow or luigi

Advantages

- Data Scientists have control of the reports and can add new ones without interference
  - Previous iterations that didn't have this control, quickly became obsolete
  - The data scientists know which questions to ask, let them!
- When they change one of the repos, they can see what happens in the report
- Engineers have control of data provenance and environment where reports are generated
- Everyone can see the progress of algorithms over time → **trust**

Take-Away

- Whatever your data pipeline, certain parts need to be strictly managed
- But data scientists need to have **freedom to explore** the data
  - Without getting bogged down debugging library dependencies, connections to databases etc
- Watch the community for frameworks that enable this workflow
- For example: Databricks
  - Designed *around* notebook/report creation
  - Friendly UI so that data scientists can freely create clusters and experiment
  - APIs/airflow support etc for the software engineers to build the pipeline

Issue #3: Code

Anti-patterns:

- Are variations of the same code/sql living in several notebooks where they cannot be shared?
- Does code consist of a main function that does all the work?
- Are dataframe objects the only objects around?

The pattern: Tool Building

- Data Science teams benefit from having team members that are **tool builders**
- The role
  - Have overview of code
  - Detect frequently used patterns & create modules out of them
  - Nudge people to create re-usable functions out of their code
  - Find new (perhaps OO) levels of abstraction
    - Pandas/Pyspark dataframes are great, but they can are a low level component that doesn't know the purpose of what they represent
    - Data can be modelled as classes, that, on a first glance, abstract the underlying dataframes
    - Less low level thinking → productivity

How to adapt these patterns

- How to discover what works for your organization? (meta)
- Interdisciplinary meetings
  - Data Science & Engineering & OPS
- Internal Presentations
  - Knowledge transfer
  - Presentation of tools
- Cultivating a culture of collaboration

We have too many meetings like this

- Designing architecture
- Getting caught in details/framework wars

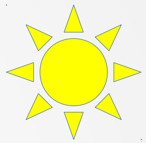
??!#^&@ ?

Let's have more meetings like this

And like this

Closing statements

- There is nothing (totally) new under the sun. Whatever problem you have, some previous generation of programmers or people in other disciplines have faced a variation of it



- Data-driven applications need to be designed for **flexibility** in exploration, but on a solid foundation to keep data **organized** and the results able to be **evaluated**

Thank you!

carolinux@github