ETL Testing with Python

Sev Leonard
October 25 2018

About me

Red2Blue

Claire
MCCASKILL

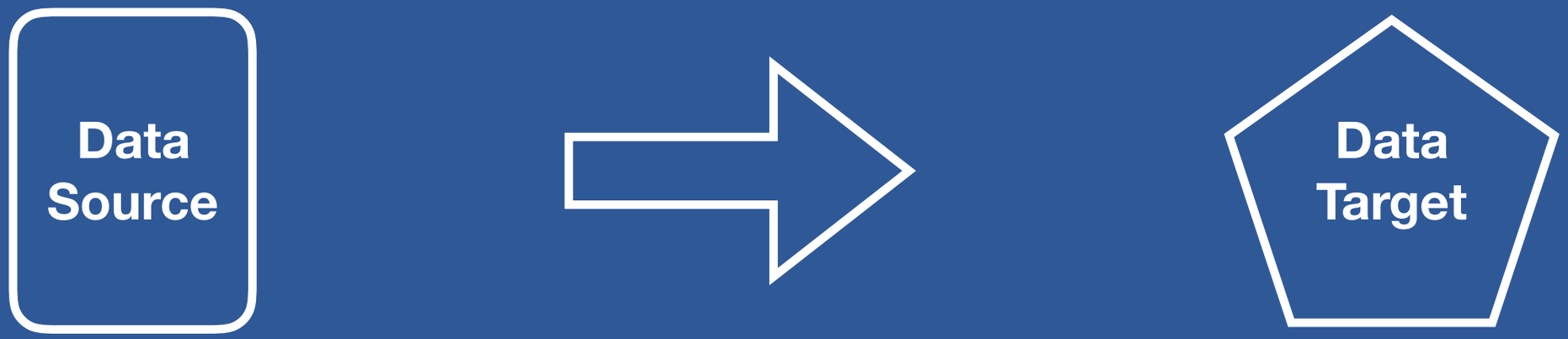
#WontBeErased

- Vote, volunteer, and donate to Kate Brown
- Vote, volunteer, and donate to JoAnn Hardesty
- Believe Trans and GNC folks, and other minorities
- https://www.out.com/news-opinion/2018/10/21/hell-no-memo-7-action-items-protect-trans-and-gnc-people
- Public comment period for memo

Agenda

- What is ETL?
- Example ETL
- Modeling ETLs for testing
- Testing ETLs with pytest

What is ETL?



API

Database

Web scraping

1. Extract from source

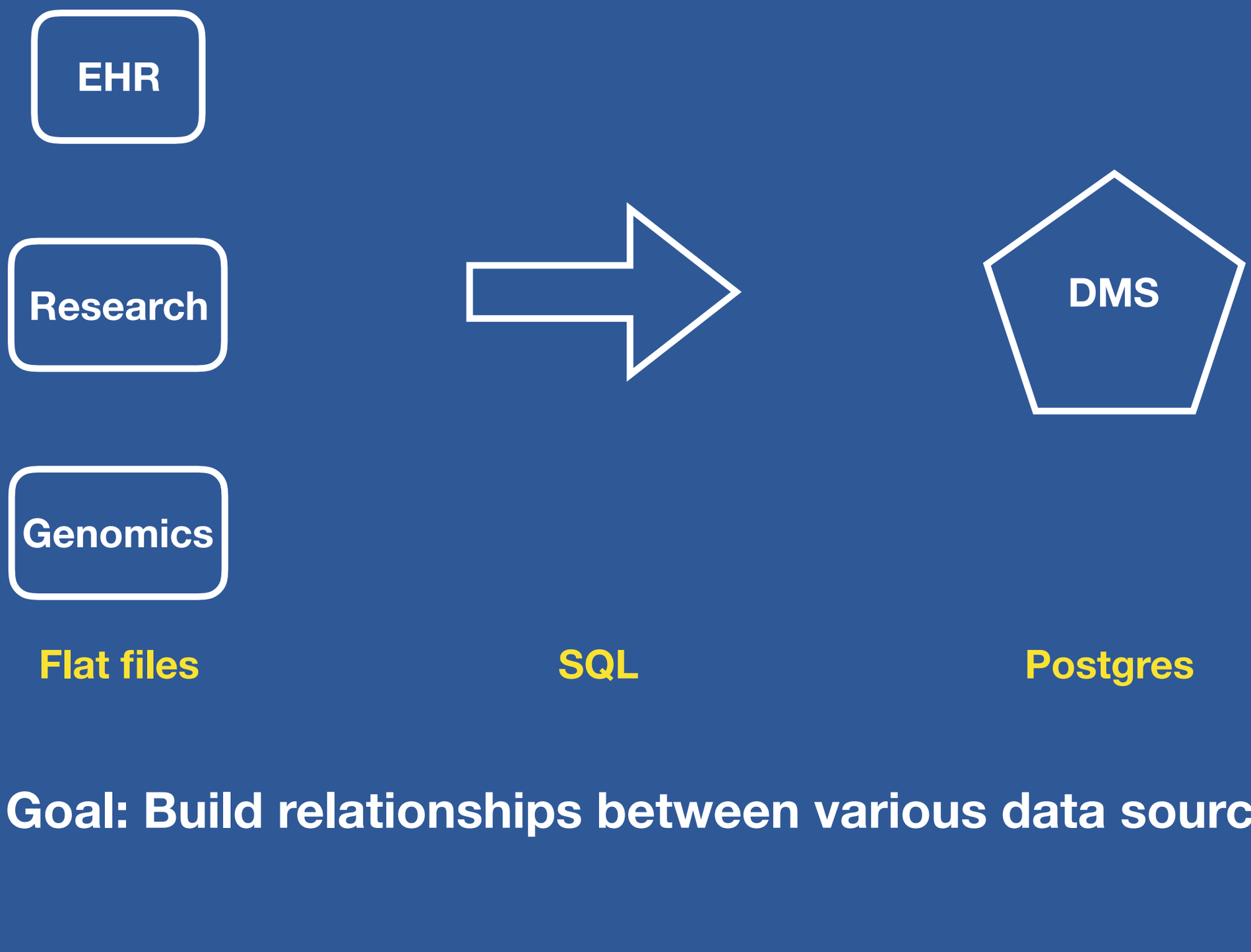
2. Transform

3. Load into target

Data Warehouse

Data Lake

Example



Goal: Build relationships between various data sources

ETL steps

1. Extract the source data
2. Create data relationships
3. Validate data expectations
4. Determine record disposition
5. Load accepted records into target

Creating relationships

111	CA XAMALIB IV 500mg
111	XAMALIB-MIX INTRAVENOUS 20mL/mg
111	RATAPIB TAB 10mg

treatment_table

PatientID	AgentID	DeliveryID
111	1	2
111	2	2
111	3	1

agent_table

AgentID	Name
1	XAMALIB
2	XAMALIB-MIX
3	RATAPIB

delivery_table

DeliveryID	Name
1	TAB
2	IV

alt_delivery_table

AltID	DeliveryID	Name
1	2	INTRAVENOUS

Transformations must provide an accurate representation of source data

Validate & Dispose

111	MEOWRAPIB IV 10mg
111	CA XAMALIB IV 500mg

agent_table

AgentID	Name
1	XAMALIB
2	XAMALIB-MIX
3	RATAPIB

interface_treatment_table

PatientID	AgentID	DeliveryID	Accept
111	X	2	**FALSE**
111	1	2	**TRUE**

What can go wrong?

- Incorrect matching logic
- Forget to add validation check for a new field
- Any bug
- Accept records that should be rejected and vice versa

agent_table

AgentID	Name
1	XAMALIB
2	XAMALIB-MIX
3	RATAPIB

Testing an ETL

- Create test source data
- Run ETL on test data
- Verify results are as expected

PatientID	Treatment	AgentID	DeliveryID	Accept
111	MEOWRAPIB IV 10mg		2	FALSE
111	CA XAMALIB IV 500mg	1	2	TRUE

interface_treatment_table

PatientID	AgentID	DeliveryID	Accept
111		2	TRUE
111	1	2	TRUE

Example scenario

pusheen_test

pusheen_prod

Studying impact of drug treatment on certain specimen assays

hellokitty_test

hellokitty_prod

Studying relationship between genomics and diagnosis

Modeling ETLs

- Consider ETL as an object:
  - Properties:
    - Source data and results columns
    - Interface table
    - SQL procs to call for each method
- Methods:
  - Load, Match, Validate, Dispose, Apply

ETL Object

Pseudo code walk through: ETL.py and etl_config.py

Pytest

- Parametrization
- Markers
- Classes optional
- Fixtures

https://www.slant.co/versus/9148/9149/~unittest_vs_pytest

Fixture & command line option setup

- conftest.py

Markers & Parametrization

- test_suite.py

Parametrization

- Two ways to parametrize:
    - pytest.mark.parametrize
    - pytest_generate_tests: for parametrizing dynamically

Reference: https://docs.pytest.org/en/latest/parametrize.html#pytest-generate-tests

Techniques

- Using command line options to dynamically create test suite
- Using markers to filter tests
- Using parametrization to run a test over a list of objects
- Need to add a new ETL? Just update etl_config.py!

Pytest docs

Parametrizing test methods through per-class configuration

Here is an example `pytest_generate_tests` function implementing a parametrization scheme similar to Michael Foord's unittest parametrizer but in a lot less code:

```
# content of ./test_parametrize.py
import pytest

def pytest_generate_tests(metafunc):
    # called once per each test function
    funcarglist = metafunc.cls.params[metafunc.function.__name__]
    argnames = sorted(funcarglist[0])
    metafunc.parametrize(argnames, [[funcargs[name] for name in argnames]
            for funcargs in funcarglist])
```

?

Navigating docs

pytest

Table Of Contents

Metafunc

class **Metafunc**(*definition*, *fixtureinfo*, *config*, *cls=None*, *module=None*) [source]

Metafunc objects are passed to the **pytest_generate_tests** hook. They help to inspect a test function and to generate tests according to test configuration or values specified in the class or module where a test function is defined.

config = *None*

access to the **_pytest.config.Config** object for the test session

module = *None*

the module object where the test function is defined in.

function = *None*

underlying python test function

fixturenames = *None*

set of fixture names required by the test function

cls = *None*

class object where the test function is defined in or None.

Use print, pdb to inspect objects

Considerations

- Changes in data and relationships can make maintaining test data a pain
- Test each ETL step as unit
- Many ways to filter test suite: markers, command line options, pytest_generate_tests

Thanks!

sev@thedatascout.com

@gizm0_0