DplyPY

Corbin Charpentier
Matthew Chau
Jessie Ren

Background

- Motivated by ease of use of R's Dplyr package
- Functional-style implementation (instead of OOP):

```
dataframe %>%
    filter(…) %>%
    select(…) %>%
    mutate(…) %>%
    arrange(…) %>%
    summarise(…)
```

(https://www.sharpsightlabs.com/blog/dplyr-quick-introduction/)

- Accomplishes function chaining using pipe operator (%>%)
- We wanted to implement something similar in Python

Data used

- Did not use any public data for unit testing
- Used seaborn built-in dataset “titanic” for integration testing and demo

Use cases

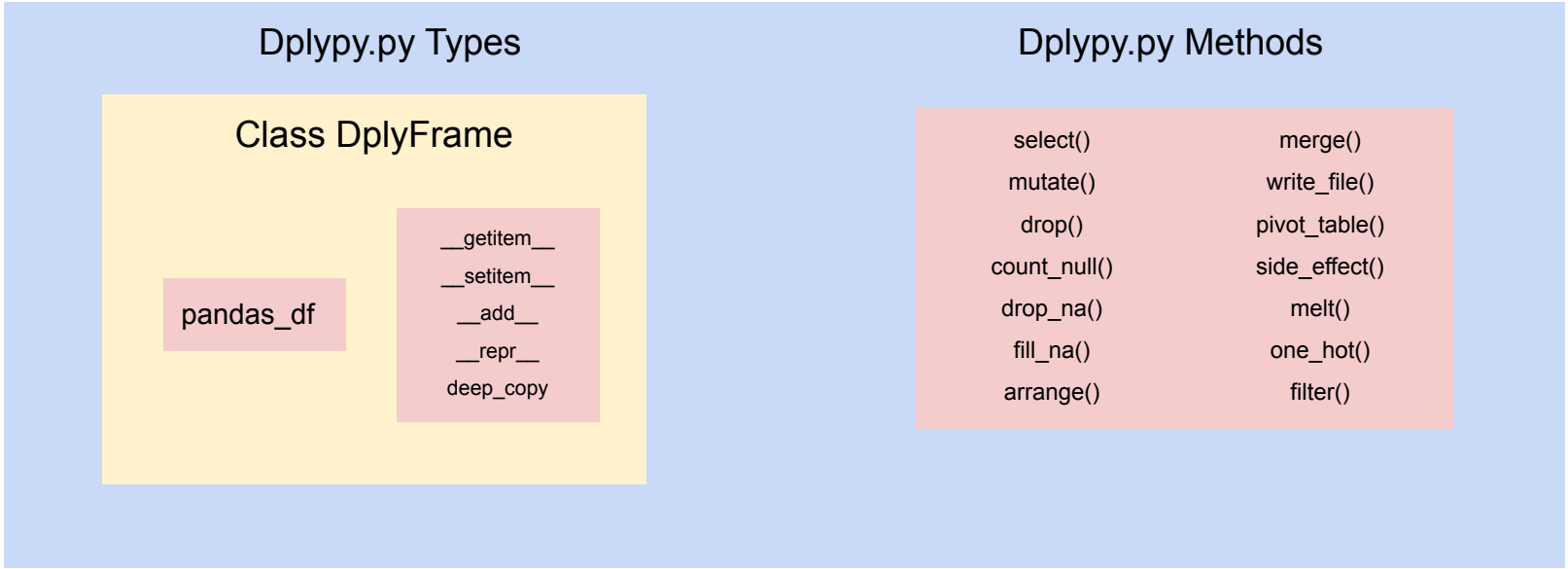
- Traditional data-frame transformations/manipulations and querying
  - We implement the basic capabilities of Python's Pandas and R's Dplyr

select	count_null
mutate	drop_na
drop	fill_na
merge	pivot_table
melt	one_hot
side_effect	write_file
arrange	filter

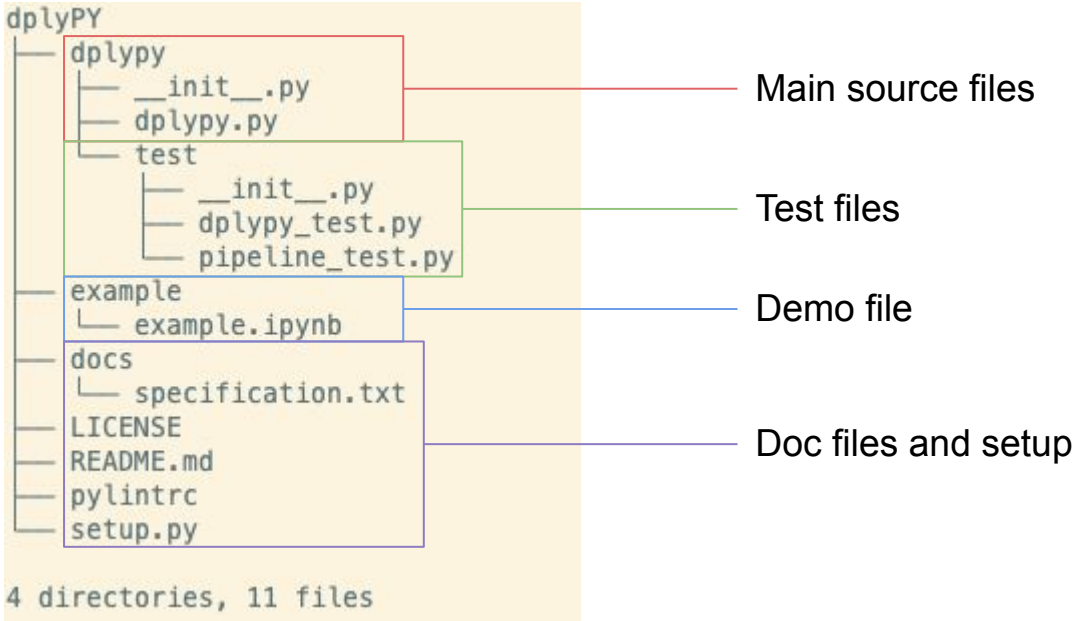
- What Pandas doesn't do: inject side-effects into pipeline

demo

Design


Dplypy.py Types
Class DplyFrame
pandas_df
__getitem__
__setitem__
__add__
__repr__
deep_copy
Dplypy.py Methods
select()
mutate()
drop()
count_null()
drop_na()
fill_na()
arrange()
merge()
write_file()
pivot_table()
side_effect()
melt()
one_hot()
filter()

Github Directory



Project Structure and Process

- Standard Python package
  - Subdirectory same name as git repository containing all source and tests
- Auto-linter on commit via local commit hook
- Used Pytest as testing framework
- Tests run server-side on each commit via github Action
- Development process
  - New features and bug fixes tracked with Github Issues
  - New features implemented in a feature branch
  - One pull-request per issue
    - Reviewed by all members of team before approved for merge

Lessons learned and future work

- When it comes to errors, follow precedent
  - Pandas provides good errors–use them
  - Don’t reinvent the wheel
- More functions with fewer parameters
  - Pandas methods are highly parameterized
    - This can be confusing
  - E.g. break merge into left_join, right_join, etc.
- Data provenance
  - Important in highly regulated domains (like clinical trials)
  - Store a breadcrumb trail of transformations for later auditing

Thank you!