pvlib-python: pandas

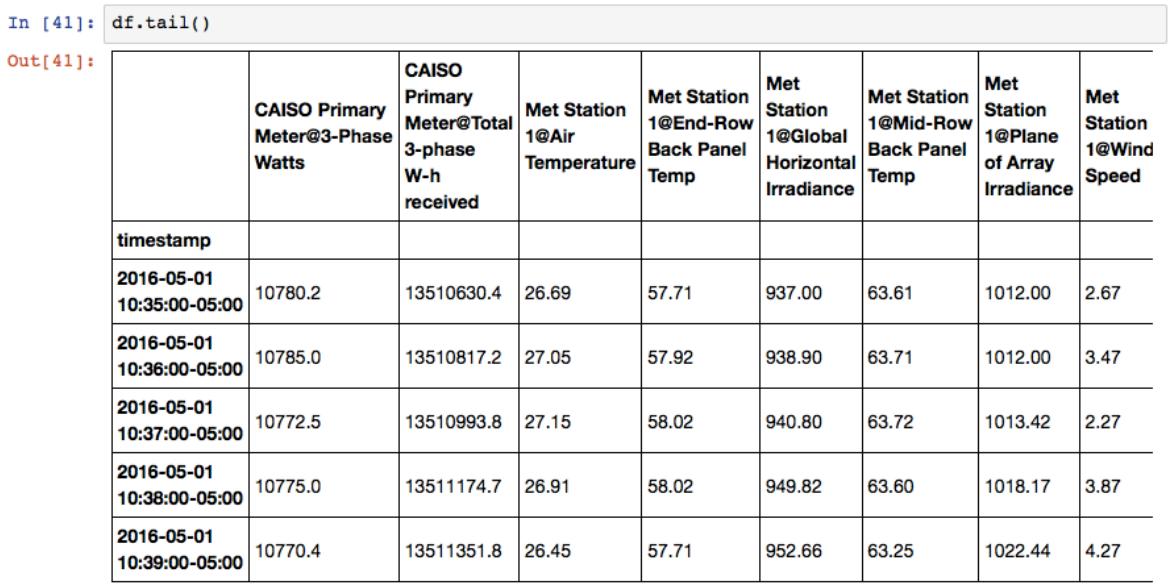
SANDIA PV MODELING CONFERENCE

MAY 10 2016

JESSICA FORBESS, SUNSHINE ANALYTICS

What is pandas?

- A package to handle data of mixed types
- The key structure is a table of data that can be of mixed types (by column), with rows of identical data
- Similar to a SQL table or an R dataframe
- Pandas provides a huge number of methods and functions to manipulate, analyze, and pull data out of the dataframe
- Particularly powerful in its handling of timeseries, where every row is a new time, regularly spaced or not



```
In [41]: df.tail()
```

Out[41]:

	CAISO Primary Meter@3-Phase Watts	CAISO Primary Meter@Total 3-phase W-h received	Met Station 1@Air Temperature	Met Station 1@End-Row Back Panel Temp	Met Station 1@Global Horizontal Irradiance	Met Station 1@Mid-Row Back Panel Temp	Met Station 1@Plane of Array Irradiance	Met Station 1@Wind Speed
timestamp								
2016-05-01 10:35:00-05:00	10780.2	13510630.4	26.69	57.71	937.00	63.61	1012.00	2.67
2016-05-01 10:36:00-05:00	10785.0	13510817.2	27.05	57.92	938.90	63.71	1012.00	3.47
2016-05-01 10:37:00-05:00	10772.5	13510993.8	27.15	58.02	940.80	63.72	1013.42	2.27
2016-05-01 10:38:00-05:00	10775.0	13511174.7	26.91	58.02	949.82	63.60	1018.17	3.87
2016-05-01 10:39:00-05:00	10770.4	13511351.8	26.45	57.71	952.66	63.25	1022.44	4.27

Overview

- Available Documentation
  - Versions matter, things are changing
  - Compare to R (a whole page on the official doc website)
- Loading data
- Time Series
  - Time zones and DST
- Slicing and Indexing
  - Best practices
  - Chained Indexing, a complicated no-no
- Merge and concat
- Plotting

Available Documentation

pandas.pydata.org

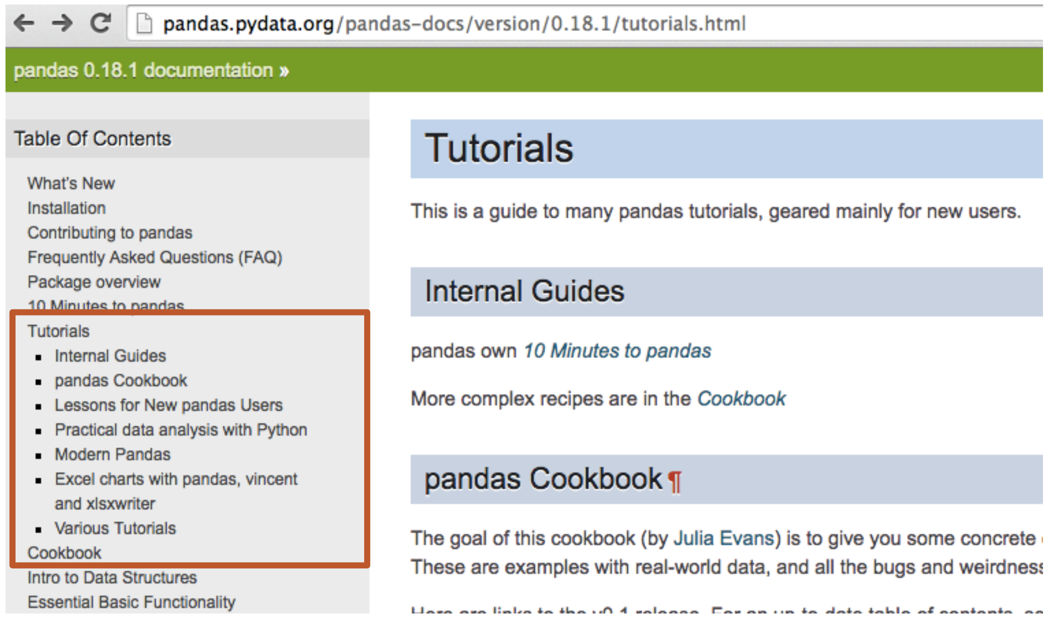
- API documentation as well as a reasonably complete manual
- Currently on 0.18.1, but was 0.16 as recently as a few months ago

stackoverflow.com

- If you google a question about python and pandas, this will be the first 3+ hits, with usually good answers
- Keep in mind that answers for specialty functions (esp Timeseries) may have changed in recent versions

So much more

- Lots of tutorials, everyone pulling out the key functions they use
- *A python notebook for pandas and data quality will be added to pvlib-python github shortly*



pandas.pydata.org/pandas-docs/version/0.18.1/tutorials.html

pandas 0.18.1 documentation »

Table Of Contents

Tutorials

This is a guide to many pandas tutorials, geared mainly for new users.

Internal Guides

pandas own *10 Minutes to pandas*

More complex recipes are in the *Cookbook*

pandas Cookbook¶

The goal of this cookbook (by Julia Evans) is to give you some concrete
These are examples with real-world data, and all the bugs and weirdness

Loading data

Multiple functions

- read_csv and read_excel are very easy to use as defaults
- parse_dates is a powerful way to convert dates on the fly (sometimes)
- Other file inputs available

Time Series

Define your index as a DatetimeIndex (dti), as discussed earlier

A dti has the concepts of:

- Frequency: hourly, minutely, five minutely, 45 minutely, whatever
- Time zone: both as originally created, and converted to (tz_localize, tz_convert)
- Properties like year, month, day, hour, minute, dayofyear, dayofweek, days_in_month, etc
- DateOffset: the default is 1 calendar day, but much more complicated offsets exist like business month begin, etc, though these are less useful for solar modeling.

Time Series

A key function is resample()

Note that resample() changed in v 0.18, and needs to be handled differently from now on

df.resample('H').mean()

df.resample('H').sum()

df.resample('H').apply(lambda x: x.sum()/12000)

- # terribly hardcoded transformation from 5 minute W to kWh

Time Series

Everyone in PV knows time zones and DST are the worst

When created, a DatetimeIndex is “timezone naïve”

df.index.tz_localize(tz=‘US/Eastern’, ambiguous=‘infer’)

- ambiguous only applies to the actual hour (2am Nov XX), not the whole timeseries following

df.index.tz_convert(‘EST’)

- EST exists, but PST does not! ‘Etc/GMT+8’ is useful here
  - (not sure why +8 rather than -8, but that’s what it is)

pvlib readthedocs has some good pointers here

PV modeling needs to handle timezones and DST differently than other use cases, so sometimes advice on stackoverflow isn’t useful

Indexing and Slicing

Lots of options, here’s what I consider easiest/best practice:

- `df[‘col name’]` rather than `df.colname` (if you have col names with spaces, etc, you can’t use the . access, and I like to be consistent) --- but it can get you in trouble, see below
- `df.loc[row, col]`
- access columns by name/label rather than integer location

Using .loc helps you avoid Chained Indexing:

- `df[‘POA'] = 22` is cool
- `d[d['Month']==3]['POA'] = np.nan` is not cool
- `d.loc[d['Month']==3,'POA'] = np.nan` is cool

Merge and concat

concat or append are useful to combine two data frames with no index overlap.

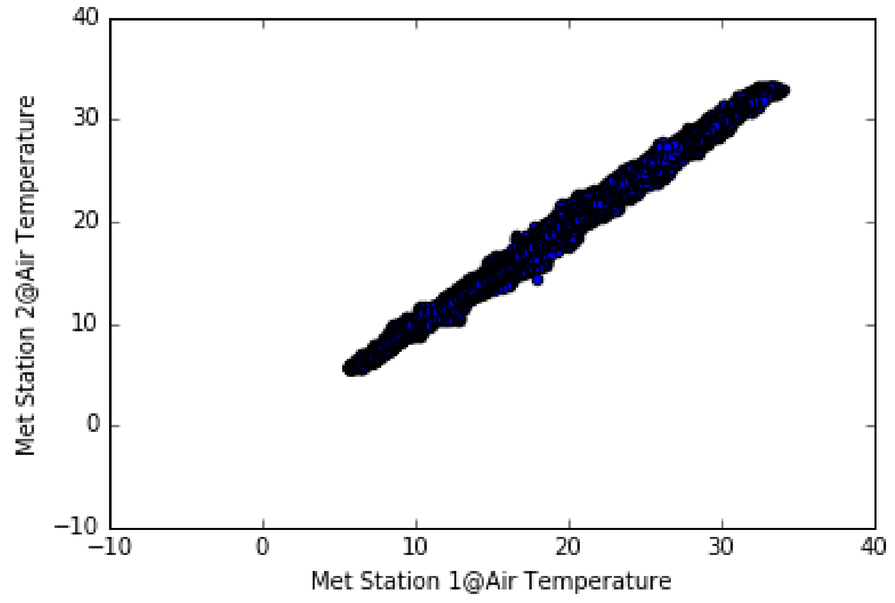
merge is similar to a SQL merge of tables, and can either keep or drop rows that don’t have data in both tables

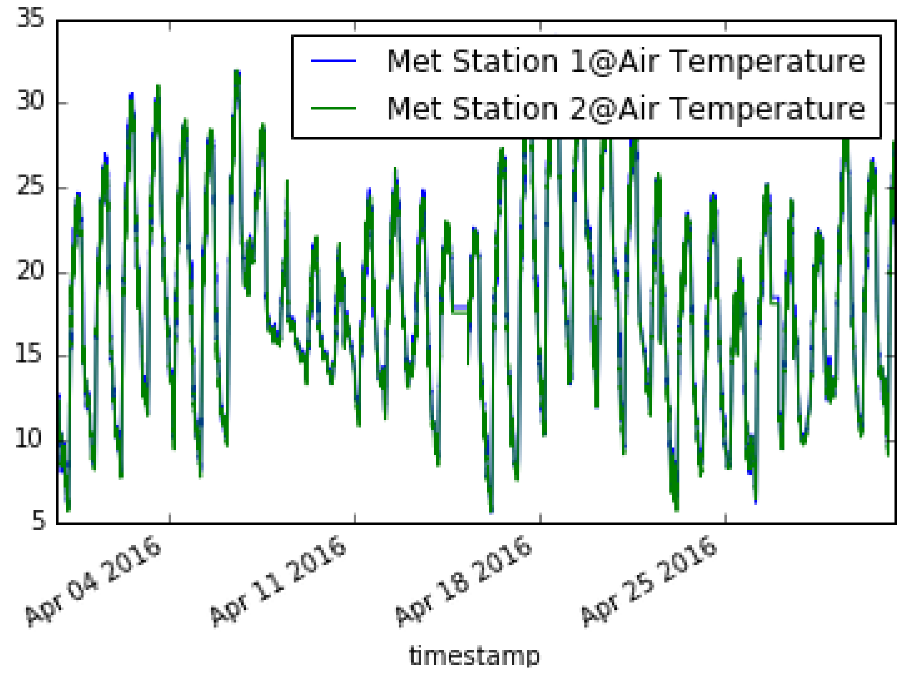
join is a simpler version of merge. I typically just use merge.

Plotting

df.plot can be useful to quickly check out your data

df.plot(kind = scatter) is the other common style I use


Met Station 2@Air Temperature
Met Station 1@Air Temperature


Met Station 1@Air Temperature
Met Station 2@Air Temperature
Apr 04 2016
Apr 11 2016
Apr 18 2016
Apr 25 2016
timestamp