pandas and data analysis

Ben Bolker

21:36 24 March 2015

- pandas cheat sheet

`pandas` stands for **pan**el **da**ta **s**ystem. It's a convenient and powerful system for handling large, complicated data sets.

Download US measles data from Project Tycho.

- `read_csv` reads a CSV file as a **data frame**; it automatically interprets the first row as headings
- `df.iloc[]` indexes the result as though it were an array
- `df.head()` shows just at the beginning; `df.tail()` shows just the end

Data frames: * rectangular data structure, a lot like an array. * can have columns (**Series**) of different types * can index by labels as well as positions * handles **missing data** * convenient plotting * fast operations with keys

```
import pandas as pd
v = "MEASLES_Cases_1909-2001_20150322001618.csv"
p  = pd.read_csv(v,skiprows=2,na_values=["-"])  ## read in data
print(p.iloc[:,0:3].head())                     ## look at the first little bit
```

```
##    YEAR  WEEK  ALABAMA
## 0  1909     1      NaN
## 1  1909     2      NaN
## 2  1909     3      NaN
## 3  1909     4      NaN
## 4  1909     5      NaN
```

Selecting

- Pandas doc, indexing and selecting
- Choosing specific columns of a data frame
- `df["NAME"]`: extract one series (column)
- `df.NAME` (*attribute* operator)
- `df.loc[:,"MASSACHUSETTS":"NEVADA"]` (index by *label*; includes endpoint)
- `df.iloc[:,range]` (index by *integer*)

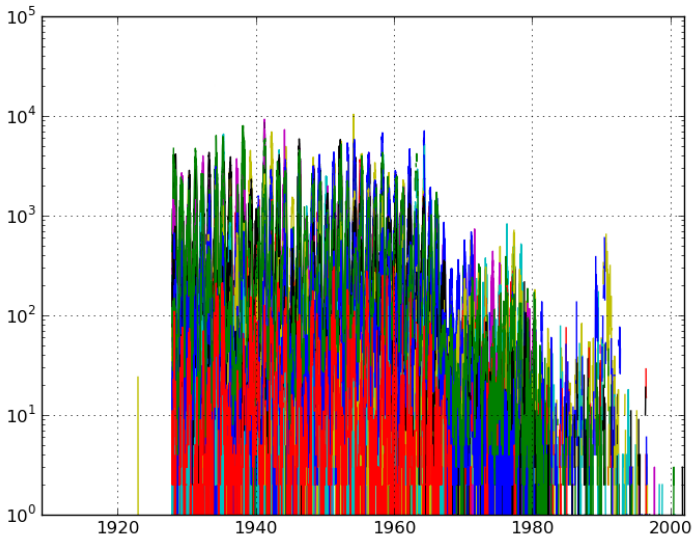
Filtering

Choosing specific rows of a data frame; `&`, `|` ,`~` correspond to `and`, `or`, `not` (individual elements *must* be in parentheses)

```
ariz = p.ARIZONA                          ## pull out a column (attribute)
ariz[(p.YEAR==1970) & (ariz>50)]          ## *must* use parentheses!
```

Basic plotting

```
pp = p.drop(["YEAR","WEEK"],axis=1)
pp.index = p.YEAR+(p.WEEK-1)/52           ## assign index
pp.plot(legend=False,logy=True)           ## plot method (non-Pythonic)
plt.savefig("pix/measles1.png")
```



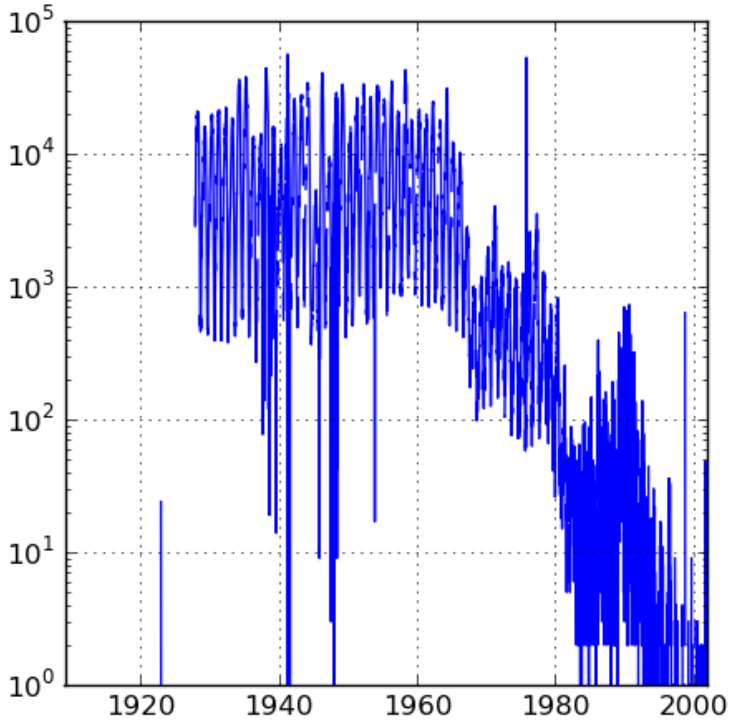

```
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(pp.index,np.log10(pp.ARIZONA))
```

Column and row manipulations

- totals by week

```
ptot = pp.sum(axis=1)
```

- `df.min`, `df.max`, `df.mean` all work too ...



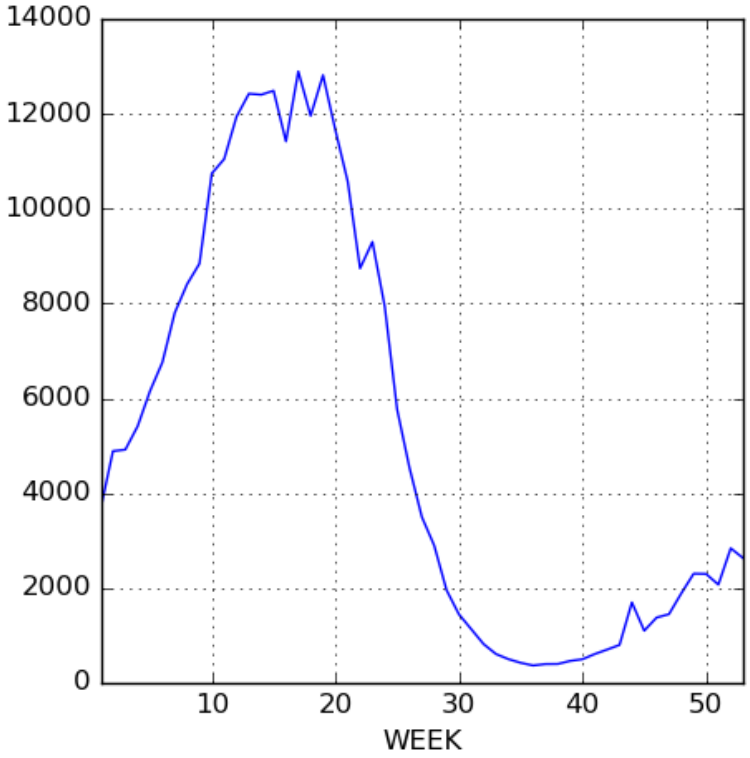
Aggregation

```
ptotweek = ptot.groupby(p.WEEK)
ptotweekmean = ptotweek.aggregate(np.mean)
ptotweekmean.plot()
```

Dates and times

reference

- (Another) complex subject.
- Lots of possible date formats
- Basic idea: something like `%Y-%m-%d`; separators just match whatever's in your data (usually "/" or "-"). Results need to be unambiguous, and ambiguity is dangerous (how is day of month specified? lower case, capital? etc.)


14000
12000
10000
8000
6000
4000
2000
0
10
20
30
40
50
WEEK

- `pandas` tries to guess, but you shouldn't let it.

```
import pandas as pd
print(pd.to_datetime("05-01-2004"))
print(pd.to_datetime("05-01-2004",format="%m-%d-%Y"))
```

```
## 2004-05-01 00:00:00
## 2004-05-01 00:00:00
```

- Time zones and daylight savings time can be a nightmare
- May need to have the right number of digits, especially in the absence of separators:

```
import pandas as pd
print(pd.to_datetime("1212004",format="%m%d%Y"))
print(pd.to_datetime("12012004",format="%m%d%Y"))
```

```
## 2004-12-01 00:00:00
## 2004-12-01 00:00:00
```

For our measles data we have week of year, so things get a little complicated

```
yearstr = p.YEAR.apply(format)
weekstr = p.WEEK.apply(format,args=["02"])
datestr = p.YEAR+"-"+weekstr+"-0"
dateindex = pd.to_datetime(datestr,format="%Y-%U-%w")
```

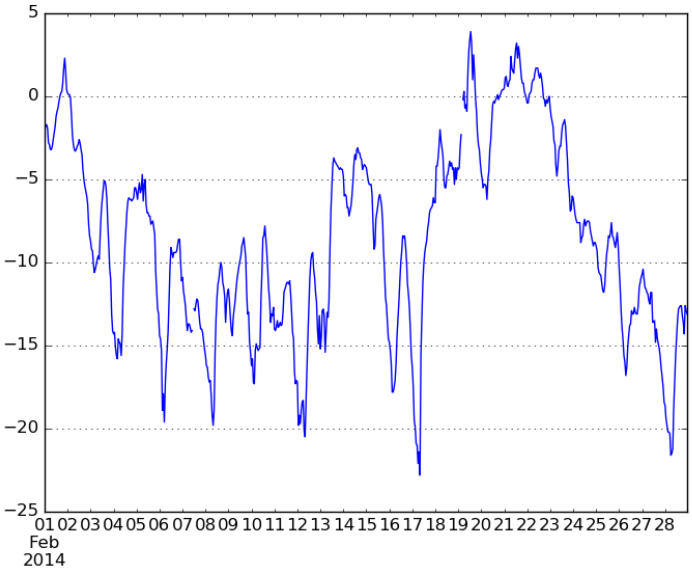
Binning results

- turn a quantitative variable into categories
- `pd.cut(x,bins=...)`; decide on bins
- `pd.qcut(x,n)`; decide on number of bins (equal occupancy)

Weather data

```
## fancy stuff: automatically look for index and convert it to a date/time
p = pd.read_csv("eng2.csv",skiprows=14,encoding="latin1",index_col="Date/Time",parse_dates=True)
## rename columns
p.columns = [
    'Year', 'Month', 'Day', 'Time', 'Data Quality', 'Temp (C)',
    'Temp Flag', 'Dew Point Temp (C)', 'Dew Point Temp Flag',
```

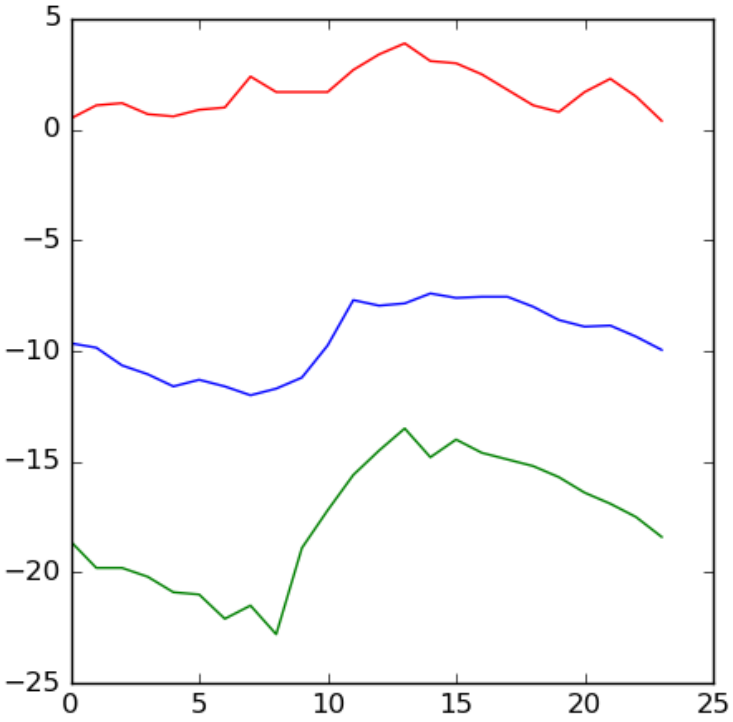
```
    'Rel Hum (%)', 'Rel Hum Flag', 'Wind Dir (10s deg)', 'Wind Dir Flag',
    'Wind Spd (km/h)', 'Wind Spd Flag', 'Visibility (km)', 'Visibility Flag',
    'Stn Press (kPa)', 'Stn Press Flag', 'Hmdx', 'Hmdx Flag', 'Wind Chill',
    'Wind Chill Flag', 'Weather']
## drop columns that are *all* NA
p = p.dropna(axis=1,how='all')
p["Temp (C)"].plot()
## get rid of columns (axis=1) we don't want
p = p.drop(['Year', 'Month', 'Day', 'Time', 'Data Quality'], axis=1)
```



Now pull out the temperature and take the median by hour:

```
temp = p[['Temp (C)']]
temp["Hour"] = temp.index.hour
temphr = temp.groupby('Hour')
medtmp = temphr.aggregate(np.median)
maxtmp = temphr.aggregate(np.max)
mintmp = temphr.aggregate(np.min)
```

Now plot these ...


5
0
−5
−10
−15
−20
−25
0
5
10
15
20
25