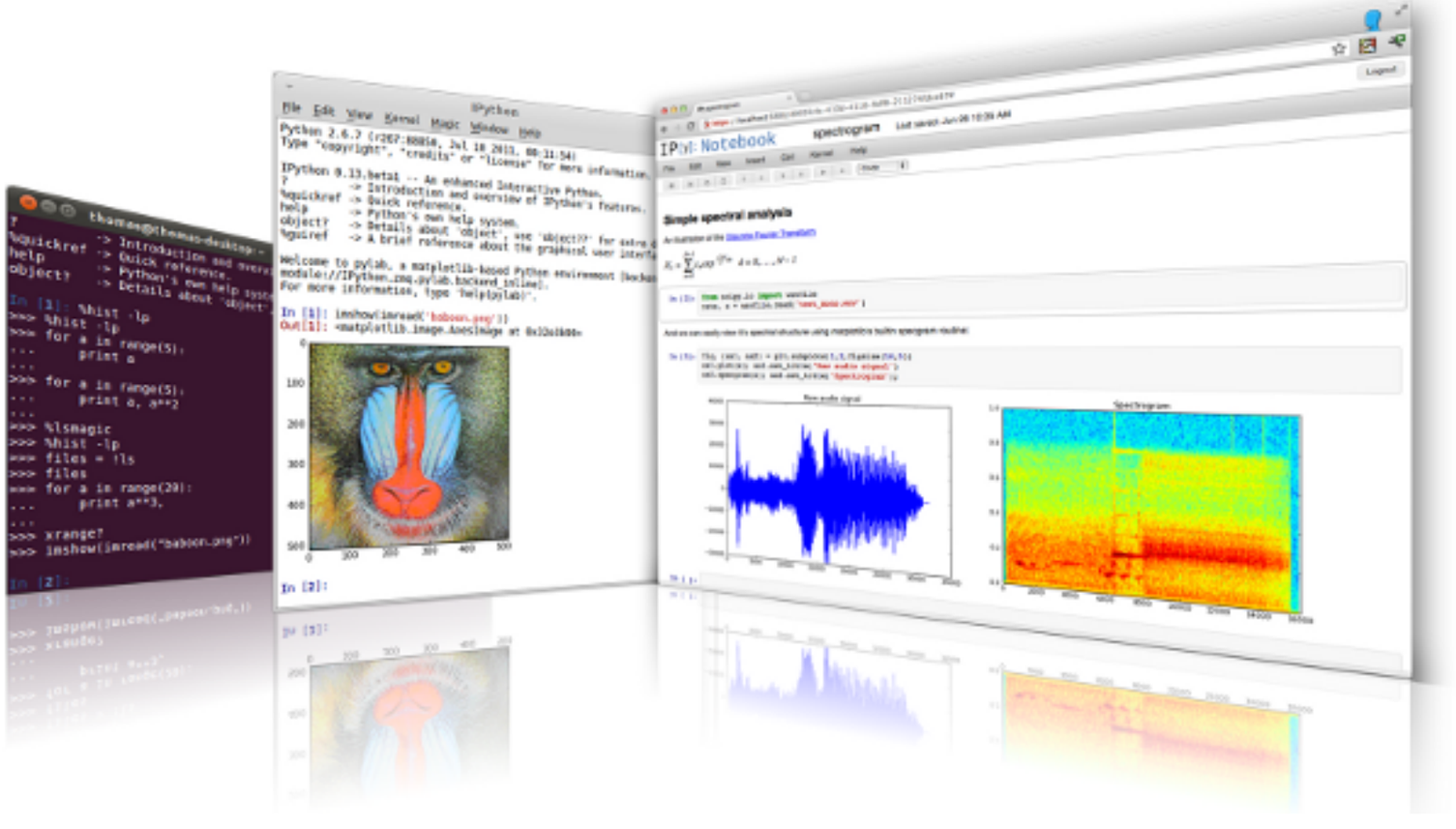
Exploring Science on Twitter

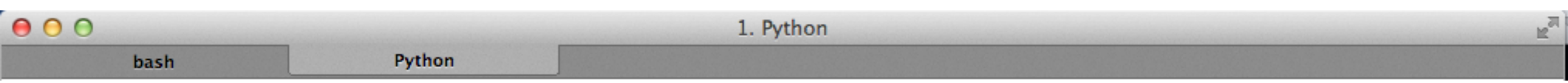
with IPython Notebook and Python Pandas

Brenda Moon
@brendam

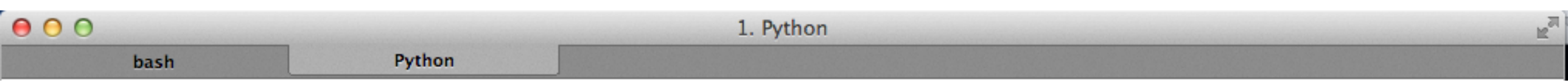
IP[y]: Notebook

http://www.ipython.org/






```
$ workon kiwipycon2013

(kiwipycon2013)$
```




```
$ workon kiwipycon2013

(kiwipycon2013)$ ipython notebook --pylab inline
```

```
(kiwipycon2013)$ ipython notebook --pylab inline

2013-08-31 22:33:35.614 [NotebookApp]

Using existing profile dir: '~/.ipython/profile_default'

Using local MathJax from ~/.ipython/profile_default/static/
mathjax/MathJax.js

Serving notebooks from local directory: /pyconNZ2013talk

The IPython Notebook is running at: http://
127.0.0.1:8888/

Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
```

IPy IPython Dashboard × IPy Example of an IPython Not ×

127.0.0.1:8888

IP[y]: Notebook

Notebooks | Clusters

To import a notebook, drag the file onto the listing below or **click here**.

Refresh | New Notebook

/ Volumes / HDD / Documents / pyconNZ2013talk /

Example of an IPython Notebook — Shutdown

Tweets per day — Delete

IPy IPython Dashboard × IPy Example of an IPython Not ×

127.0.0.1:8888/0eff4b35-415a-4b1d-99aa-a36867488bf9

IP[y]: Notebook Example of an IPython Notebook

File Edit View Insert Cell Kernel Help

Heading 1 Cell Toolbar: None

Example of an Ipy

virtualenv: kiwipycon2013

1 September 2013

Simple example of how you can record notes about what you are doing interleaved with the code you are writing.

In [11]:
```
import pandas
pandas.load??
```

IPy IPython Dashboard × IPy Example of an IPython Not ×

127.0.0.1:8888/0eff4b35-415a-4b1d-99aa-a36867488bf9

IP[y]: Notebook Example of an IPython Notebook

File Edit View Insert Cell Kernel Help

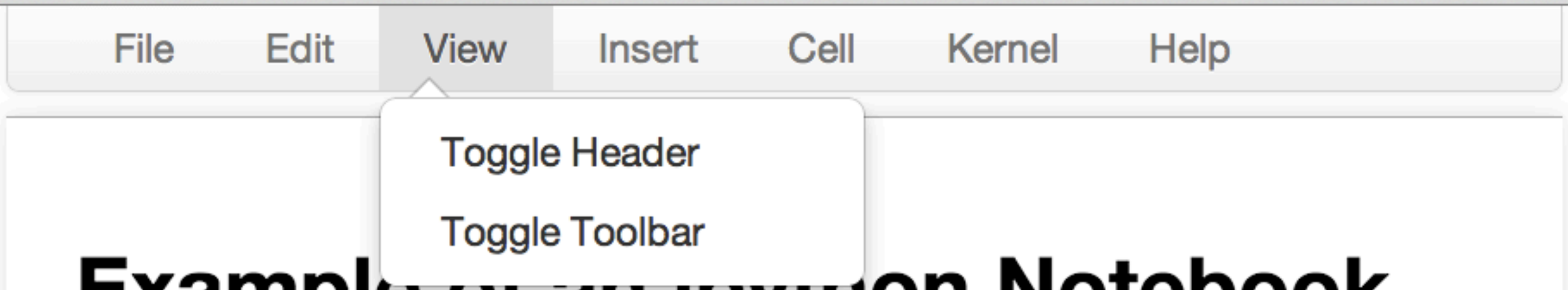
Heading 3 Cell Toolbar: None

Example of an ipython Notebook

virtualenv: kiwipycon2013

1 September 2013

Simple example of how you can record notes about what you are doing interleaved with the code you are writing.



Example of an IPython Notebook

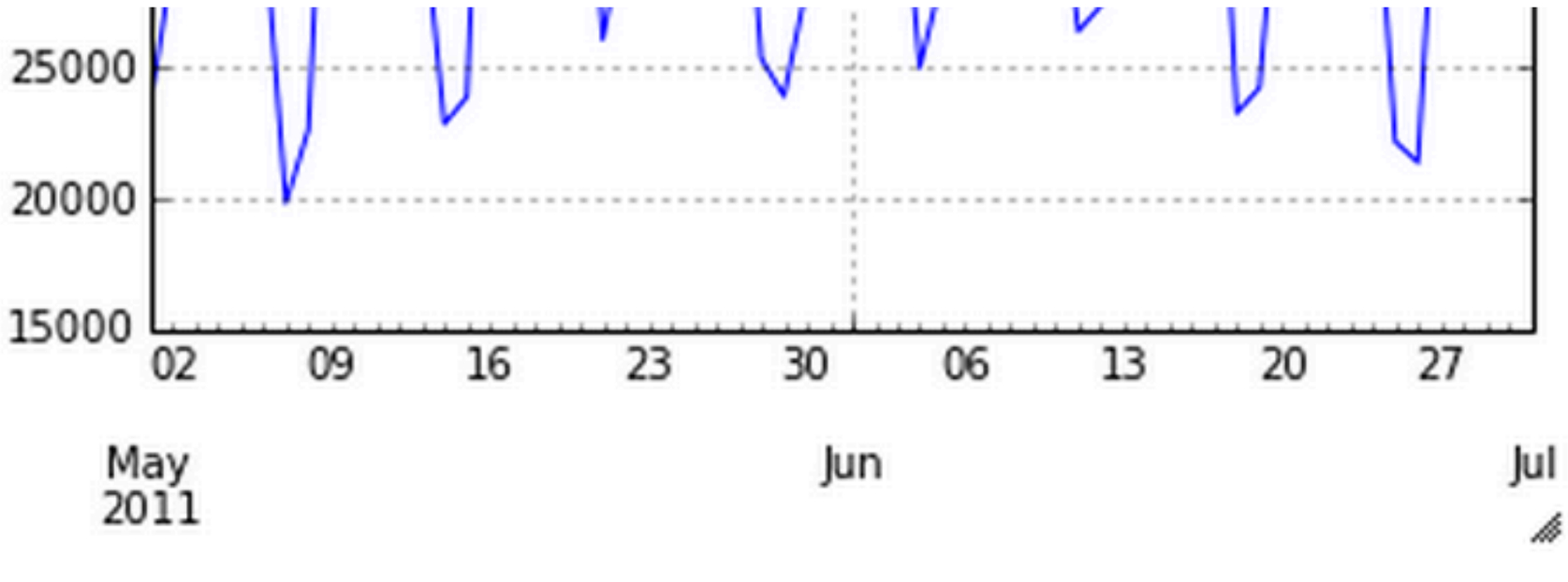
virtualenv: kiwipycon2013

1 September 2013

Simple example of how you can record notes about what you are doing interleaved with the code you are writing.

In [4]:
```
import pandas
```

IP[y]: Notebook



Matplotlib doesn't understand pandas datetime objects, so if you want to use matplotlib dates to format your tick lables, you need to convert them back using to_pydatetime(). This also shows how you can pass pandas data to matplotlib.

In [8]:
```
ax = plt.figure(figsize=(7,4), dpi=300).add_subplot(111)
two_weeks = science_tweets['2011-05-01':'2011-05-14']
ax.plot_date(two_weeks.index.to_pydatetime(), two_weeks, '
ax.xaxis.set_minor_locator(mpl.dates.WeekdayLocator(byweek
```

IPython Notebook Viewer

A Simple way to share your IP[y]thon Notebook as Gists.

Share your own notebook, or browse others'

Enter a gist number or url

Go!

Probabilistic Programming

Why would I want samples from the posterior, anyways?

We will deal with this question for the remainder of the book, and it is an understatement to say we can perform amazingly useful things. For now, let's finishing with using posterior samples to answer the follow question: what is the expected number of texts at day t, $0 \le t \le 70$? Recall that the expected value of a Poisson is equal to its parameter λ, then the question is equivalent to *what is the expected value of λ at time t?*

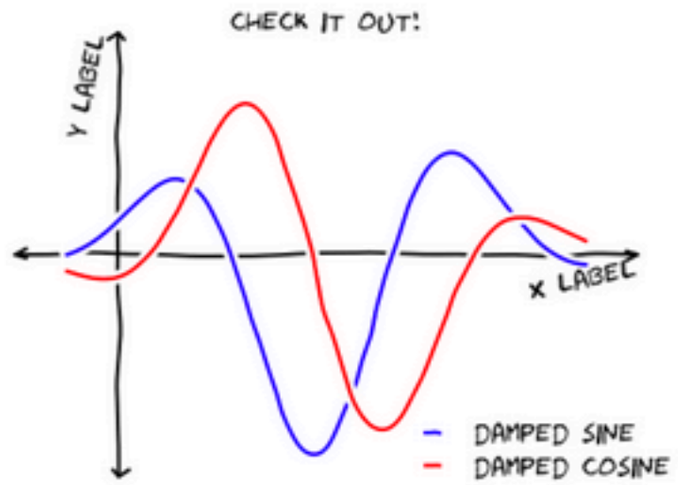
In the code below, we are calculating the following: Let i index a particular sample from the posterior distributions. Given a day t, we average over all λ_i on that day t, using $\lambda_{1,i}$ if $t < \tau_i$ else we use $\lambda_{2,i}$.

```
<matplotlib.legend.Legend at 0x149eba20>
```

XKCD Plot With Matplotlib

XKCD plots in Matplotlib

Out [1]:



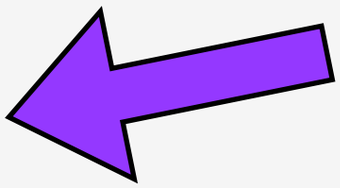
Sometimes when showing schematic plots, this is the type of figure I want to display. But drawing it by hand is a p
matplotlib. The problem is, matplotlib is a bit too precise. Attempting to duplicate this figure in matplotlib leads to s

http://nbviewer.ipython.org

File Edit View Insert Cell Kernel Help

Simple example of how you can record notes about what you are doing interleaved with the code you are writing.

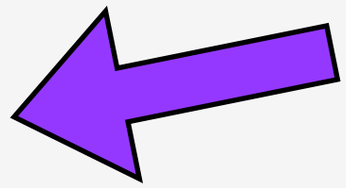
In [4]:
```
import pandas
pandas.read_pickle?
```



```
Type:        function
String Form:<function read_pickle at 0x108b5d848>
File:        /Users/brenda/.virtualenvs/kiwipycon2013/lib/pytho
n2.7/site-packages/pandas/io/pickle.py
Definition: pandas.read_pickle(path)
Docstring:
Load pickled pandas object (or any other pickled object) from
the specified
file path

Warning: Loading pickled data received from untrusted sources
can be unsafe.
See: http://docs.python.org/2.7/library/pickle.html
```

File Edit View Insert Cell Kernel Help

In [6]:
```
import pandas
pandas.read_pickle??
```



```
Type:           function
String Form:<function read_pickle at 0x108b5d848>
File:           /Users/brenda/.virtualenvs/kiwipycon2013/lib/pytho
n2.7/site-packages/pandas/io/pickle.py
Definition: pandas.read_pickle(path)
Source:
def read_pickle(path):
    """
    Load pickled pandas object (or any other pickled object) f
rom the specified
    file path

    Warning: Loading pickled data received from untrusted sour
ces can be unsafe.
    See: http://docs.python.org/2.7/library/pickle.html
```

IPython
%magic functions

run %magic to see available options

%timeit

Time the execution of a cell or line

%timeit

Time the execution of a cell or line

```
Number of tweets in month 1 is 802821
```

```
33]: %timeit tweet_text['text'].map(lambda x: x.lower())
     %timeit [x.lower() for x in tweet_text.text]
     %timeit [x.lower() for x in tweet_text['text']]
```

```
1 loops, best of 3: 2.04 s per loop
1 loops, best of 3: 1.9 s per loop
1 loops, best of 3: 1.81 s per loop
```

%pastebin

Share code as Gist on GitHub

%pastebin [-d "Custom description"] 1-7

%save / %load

Save a cell or range of cells to .py

Load a file into a cell

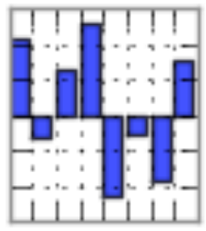
%run

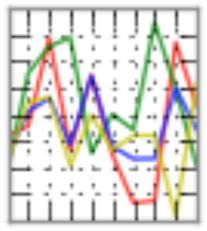
Run a python script.

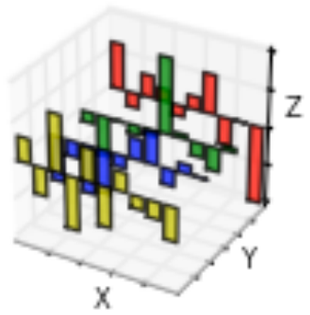
```
In [19]: %run '../graphLayoutFunctions.py'
```

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$







http://pandas.pydata.org/

"high-performance, easy-to-use data structures and data analysis tools"

- series (array like)
- dataFrame (table)

strong support for time based indexing

Tweets

keyword: ‘science’

```
import pandas
import couchdbkit

server = couchdbkit.Server('http://brenda:XXXXX@127.0.0.1:5984/')
tweetdb = server.get_db('tweets')
tweets = list(
    tweetdb.view(
        "tweetsPerDay/tweetsPerDay",
        reduce=True,
        group_level=3,
        startkey=[2011, 1, 1],
        endkey=[2012, 1, 1]))
date_list = [
    pandas.datetime(tweet["key"][0], (tweet["key"][1]), tweet["key"][2])
    for tweet in tweets]
date_index = pandas.DatetimeIndex(date_list)
data_list = [tweet["value"] for tweet in tweets]
science_tweets = pandas.Series(data_list, date_index)
# change to float so can have NaN values
science_tweets = science_tweets.astype(float)
# mask out the missing data period so it doesn't plot
science_tweets['2011-03-31':'2011-04-12'] = numpy.NaN
# final check that start and end dates are correct
print 'first element: ', science_tweets.first('1D')
print 'last element: ', science_tweets.last('1D')
# check than the missing values don't plot.
science_tweets.plot()
science_tweets.to_pickle('dataFiles/2011TweetsPerDay-final.pkl')
```

```
import pandas
import couchdbkit
```

```
server = couchdbkit.Server('http://brenda:XXXXX@127.0.0.1:5984/')
tweetdb = server.get_db('tweets')
tweets = list(
    tweetdb.view(
        "tweetsPerDay/tweetsPerDay",
        reduce=True,
        group_level=3,
        startkey=[2011, 1, 1],
        endkey=[2012, 1, 1]))
```

```
date_list = [
    pandas.datetime(tweet["key"][0], (tweet["key"][1]), tweet["key"][2])
    for tweet in tweets]
date_index = pandas.DatetimeIndex(date_list)
data_list = [tweet["value"] for tweet in tweets]
science_tweets = pandas.Series(data_list, date_index)
# change to float so can have NaN values
science_tweets = science_tweets.astype(float)
# mask out the missing data period so it doesn't plot
science_tweets['2011-03-31':'2011-04-12'] = numpy.NaN
# final check that start and end dates are correct
print 'first element: ', science_tweets.first('1D')
print 'last element: ', science_tweets.last('1D')
# check than the missing values don't plot.
science_tweets.plot()
science_tweets.to_pickle('dataFiles/2011TweetsPerDay-final.pkl')
```

```
import pandas
import couchdbkit

server = couchdbkit.Server('http://brenda:XXXXX@127.0.0.1:5984/')
tweetdb = server.get_db('tweets')
tweets = list(
    tweetdb.view(
        "tweetsPerDay/tweetsPerDay",
        reduce=True,
        group_level=3,
        startkey=[2011, 1, 1],
        endkey=[2012, 1, 1]))
date_list = [
    pandas.datetime(tweet["key"][0], (tweet["key"][1]), tweet["key"][2])
    for tweet in tweets]
date_index = pandas.DatetimeIndex(date_list)
data_list = [tweet["value"] for tweet in tweets]
science_tweets = pandas.Series(data_list, date_index)
# change to float so can have NaN values
science_tweets = science_tweets.astype(float)
# mask out the missing data period so it doesn't plot
science_tweets['2011-03-31':'2011-04-12'] = numpy.NaN
# final check that start and end dates are correct
print 'first element: ', science_tweets.first('1D')
print 'last element: ', science_tweets.last('1D')
# check than the missing values don't plot.
science_tweets.plot()
science_tweets.to_pickle('dataFiles/2011TweetsPerDay-final.pkl')
```

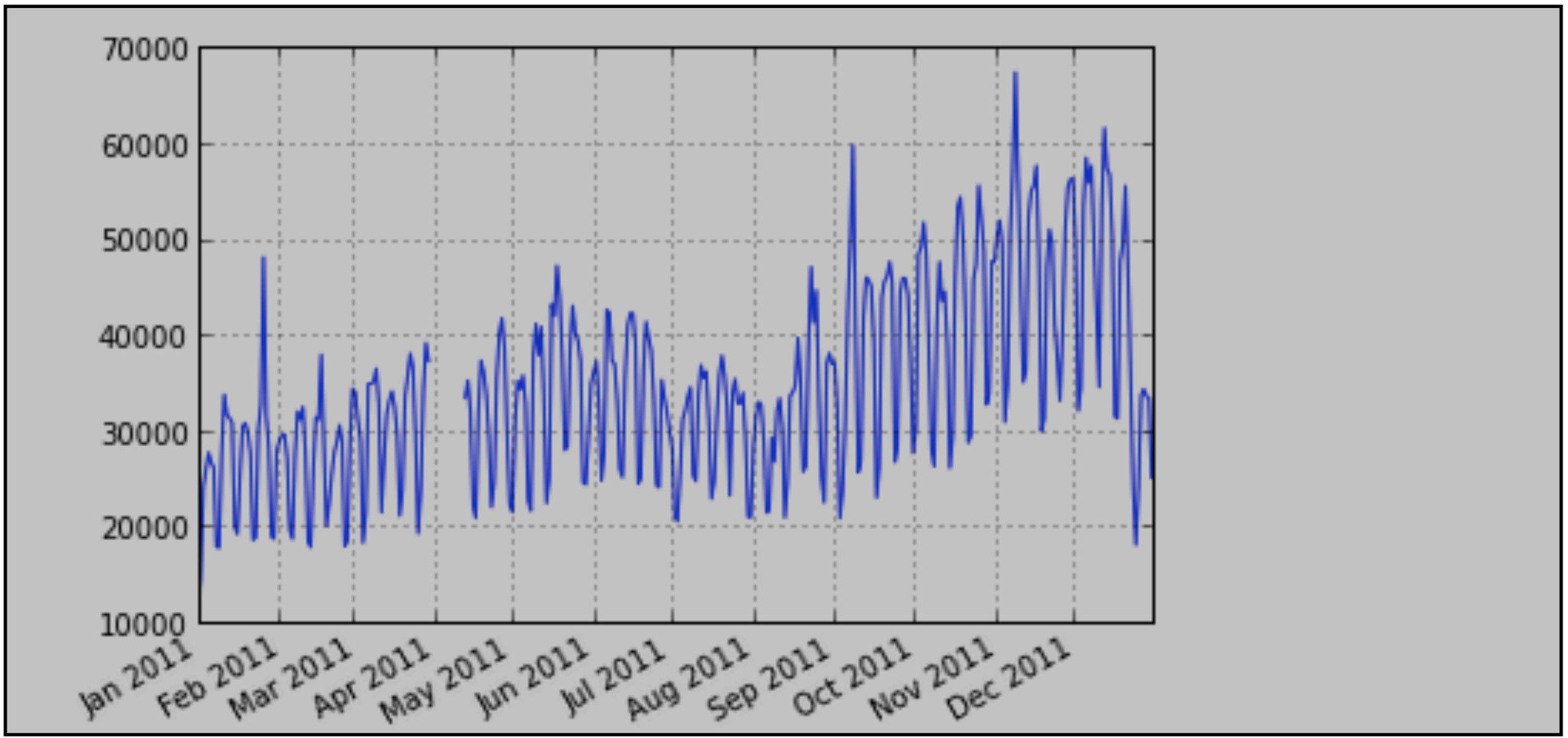
```
import pandas
import couchdbkit

server = couchdbkit.Server('http://brenda:XXXXX@127.0.0.1:5984/')
tweetdb = server.get_db('tweets')
tweets = list(
    tweetdb.view(
        "tweetsPerDay/tweetsPerDay",
        reduce=True,
        group_level=3,
        startkey=[2011, 1, 1],
        endkey=[2012, 1, 1]))
date_list = [
    pandas.datetime(tweet["key"][0], (tweet["key"][1]), tweet["key"][2])
    for tweet in tweets]
date_index = pandas.DatetimeIndex(date_list)
data_list = [tweet["value"] for tweet in tweets]
science_tweets = pandas.Series(data_list, date_index)
```

```
# change to float so can have NaN values
science_tweets = science_tweets.astype(float)
# mask out the missing data period so it doesn't plot
science_tweets['2011-03-31':'2011-04-12'] = numpy.NaN
```

```
# final check that start and end dates are correct
print 'first element: ', science_tweets.first('1D')
print 'last element: ', science_tweets.last('1D')
# check than the missing values don't plot.
science_tweets.plot()
science_tweets.to_pickle('dataFiles/2011TweetsPerDay-final.pkl')
```

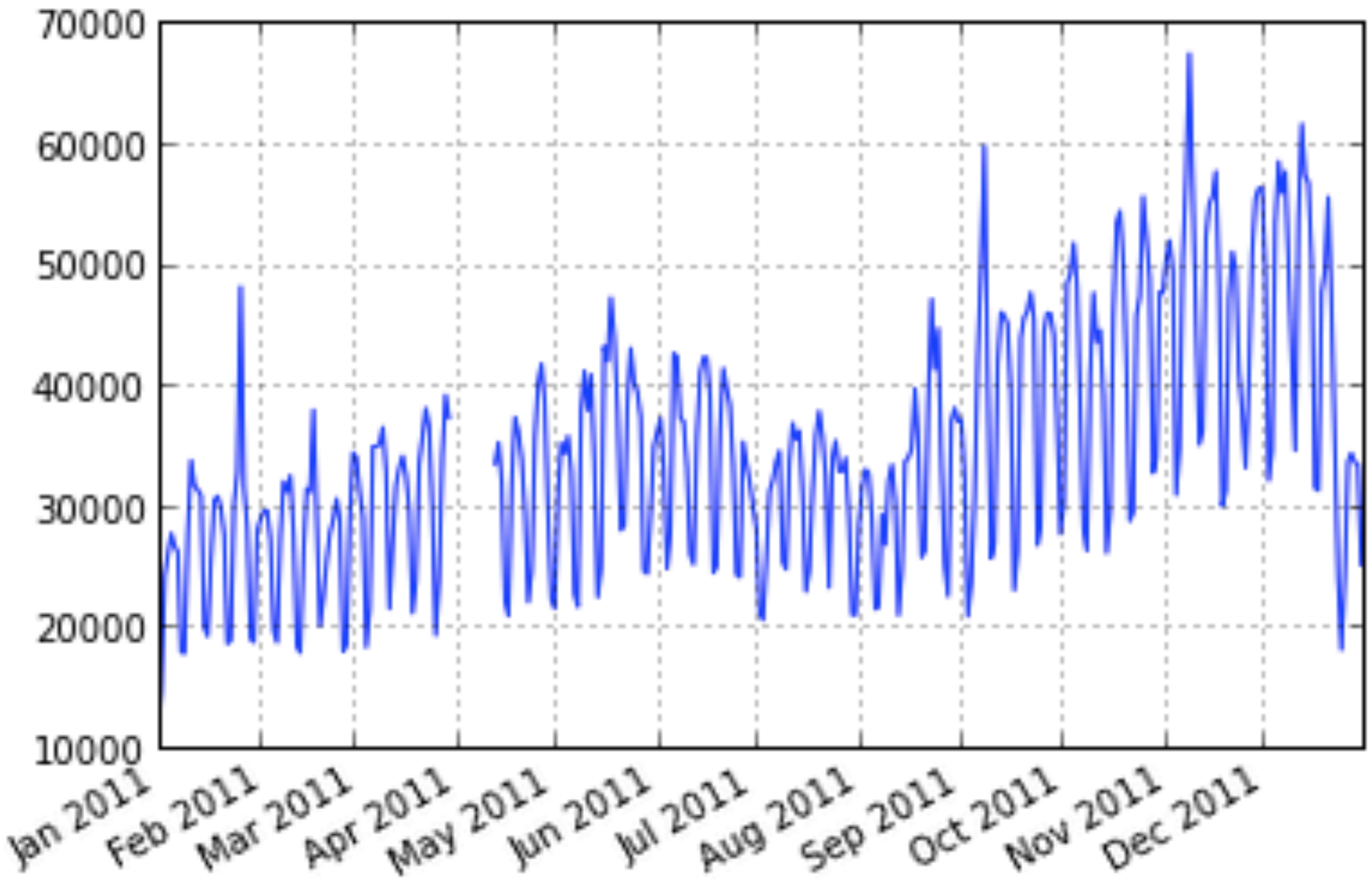
```
# final check that start and end dates are correct
print 'first element: ', science_tweets.first('1D')
print 'last element: ', science_tweets.last('1D')
# check than the missing values don't plot.
science_tweets.plot()
science_tweets.to_pickle('dataFiles/2011TweetsPerDay-final.pkl')
```

```
first element:  2011-01-01    11070
dtype: float64
last element:  2011-12-31    25067
dtype: float64
```



```
# final check that start and end dates are correct
print 'first element: ', science_tweets.first('1D')
print 'last element: ', science_tweets.last('1D')
# check than the missing values don't plot.
science_tweets.plot()
science_tweets.to_pickle('dataFiles/2011TweetsPerDay-final.pkl')
```

```
first element:  2011-01-01    11070
dtype: float64
last element:  2011-12-31    25067
dtype: float64
```



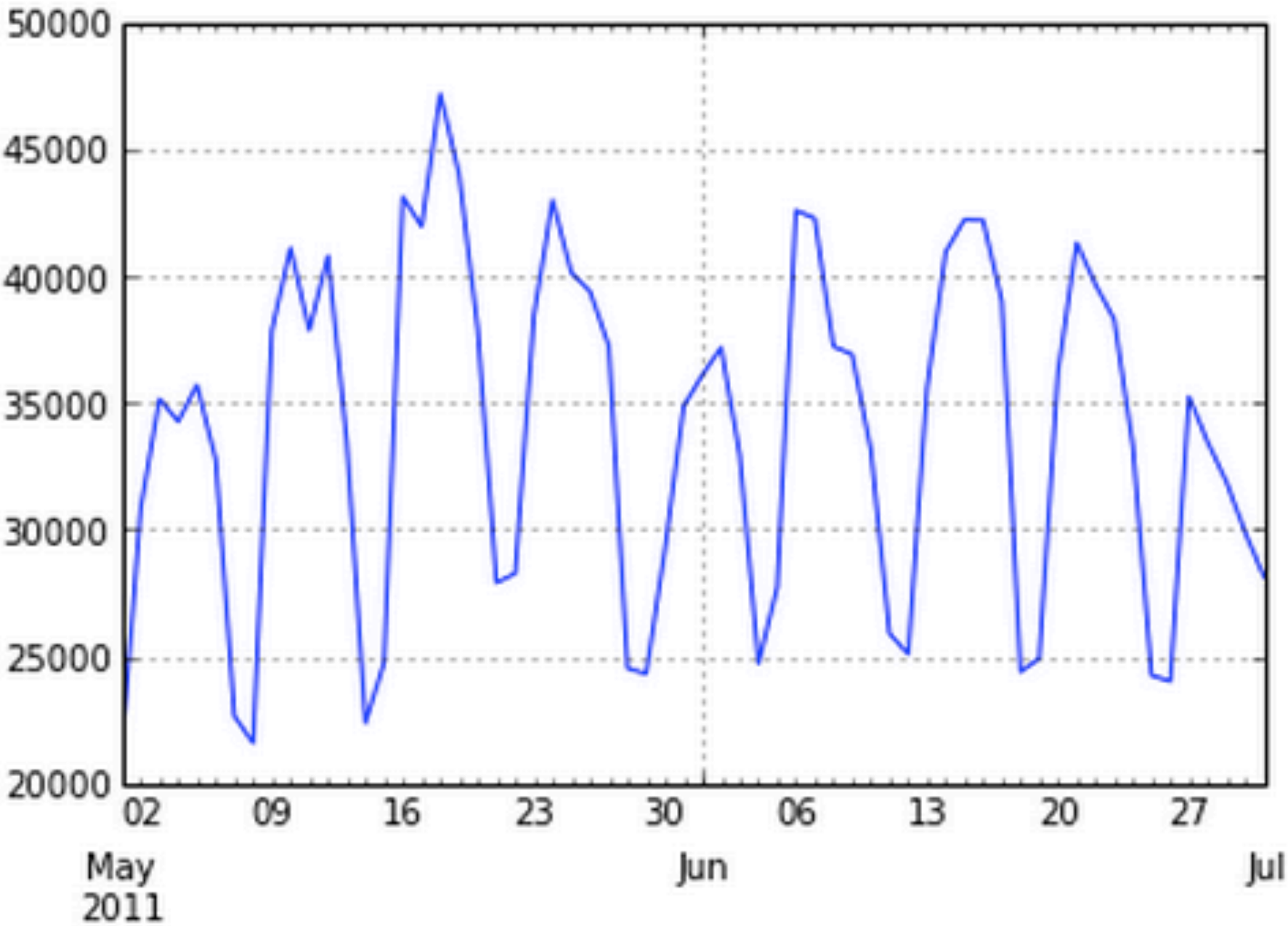
```python
# different ways to access the start of the timeseries
print "science_tweets.first('1D')", science_tweets.first('1D')
print '\nscience_tweets.head(1)', science_tweets.head(1)
print '\nscience_tweets[0]', science_tweets[0]
print "\nscience_tweets['2011-01-01']", science_tweets['2011-01-01']
print "\nscience_tweets.first('1W')", science_tweets.first('1W')
```

```
science_tweets.first('1D') 2011-01-01    11070
dtype: float64
science_tweets.head(1) 2011-01-01    11070
dtype: float64

science_tweets[0] 11070.0

science_tweets['2011-01-01'] 11070.0

science_tweets.first('1W') 2011-01-01    11070
2011-01-02    14542
dtype: float64
```

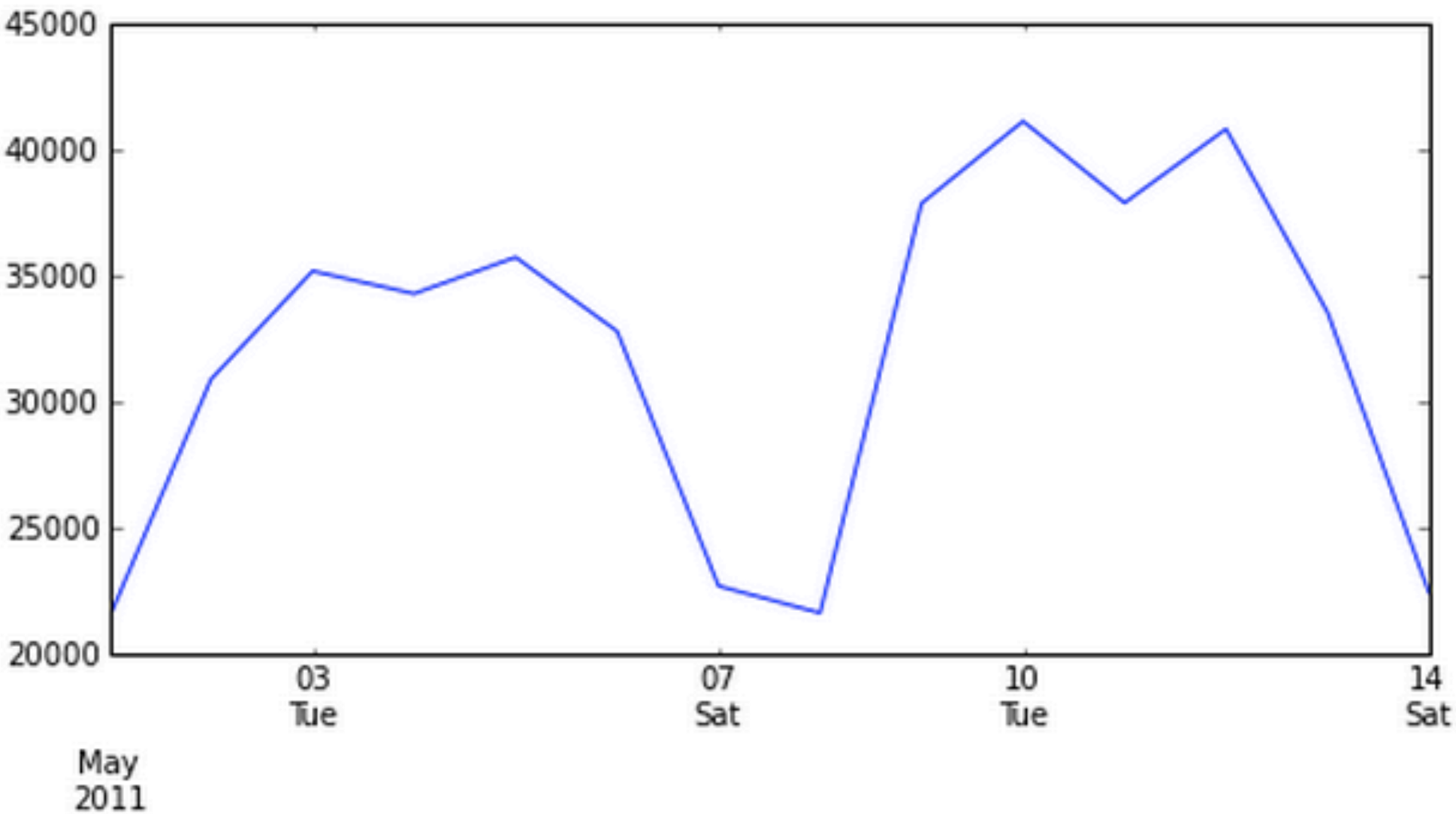
```python
# ambiguity of date strings
print "\nscience_tweets['2011-07-01']", science_tweets['2011-07-01']
print "\nscience_tweets['2011-01-07']", science_tweets['2011-01-07']
print "\nscience_tweets[pandas.datetime(2011,1,7)]"
print science_tweets[pandas.datetime(2011,1,7)]
print "\npandas.to_datetime(['07-01-2011'], dayfirst=True)"
science_tweets[pandas.to_datetime(['07-01-2011'], dayfirst=True)]
```

```
science_tweets['2011-07-01'] 28146.0

science_tweets['2011-01-07'] 26095.0

science_tweets[pandas.datetime(2011,1,7)]
26095.0

pandas.to_datetime(['07-01-2011'], dayfirst=True)
2011-01-07    26095
dtype: float64
```

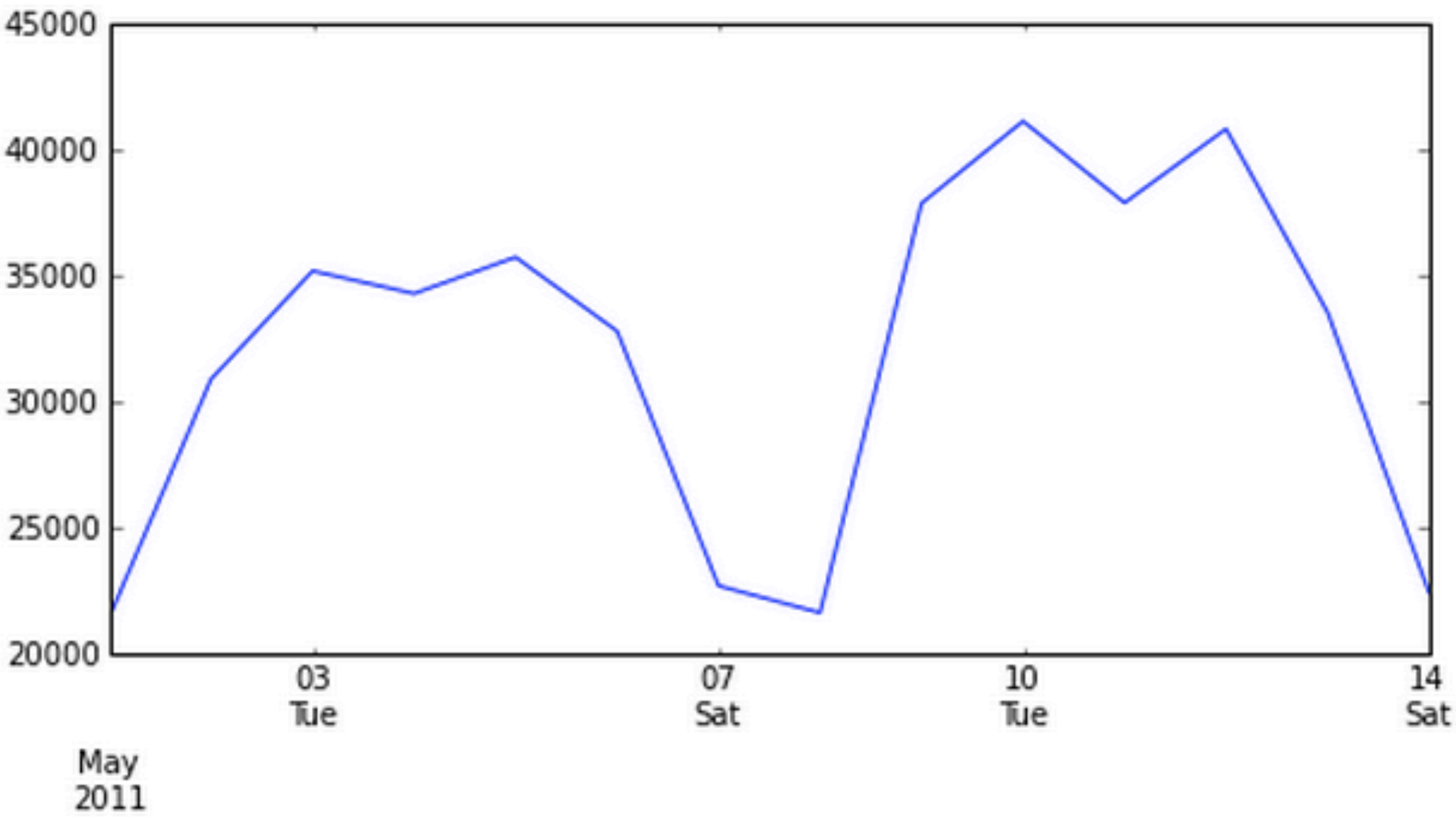
```
# zoom in using a date range
science_tweets['2011-05-01':'2011-07-01'].plot();
```



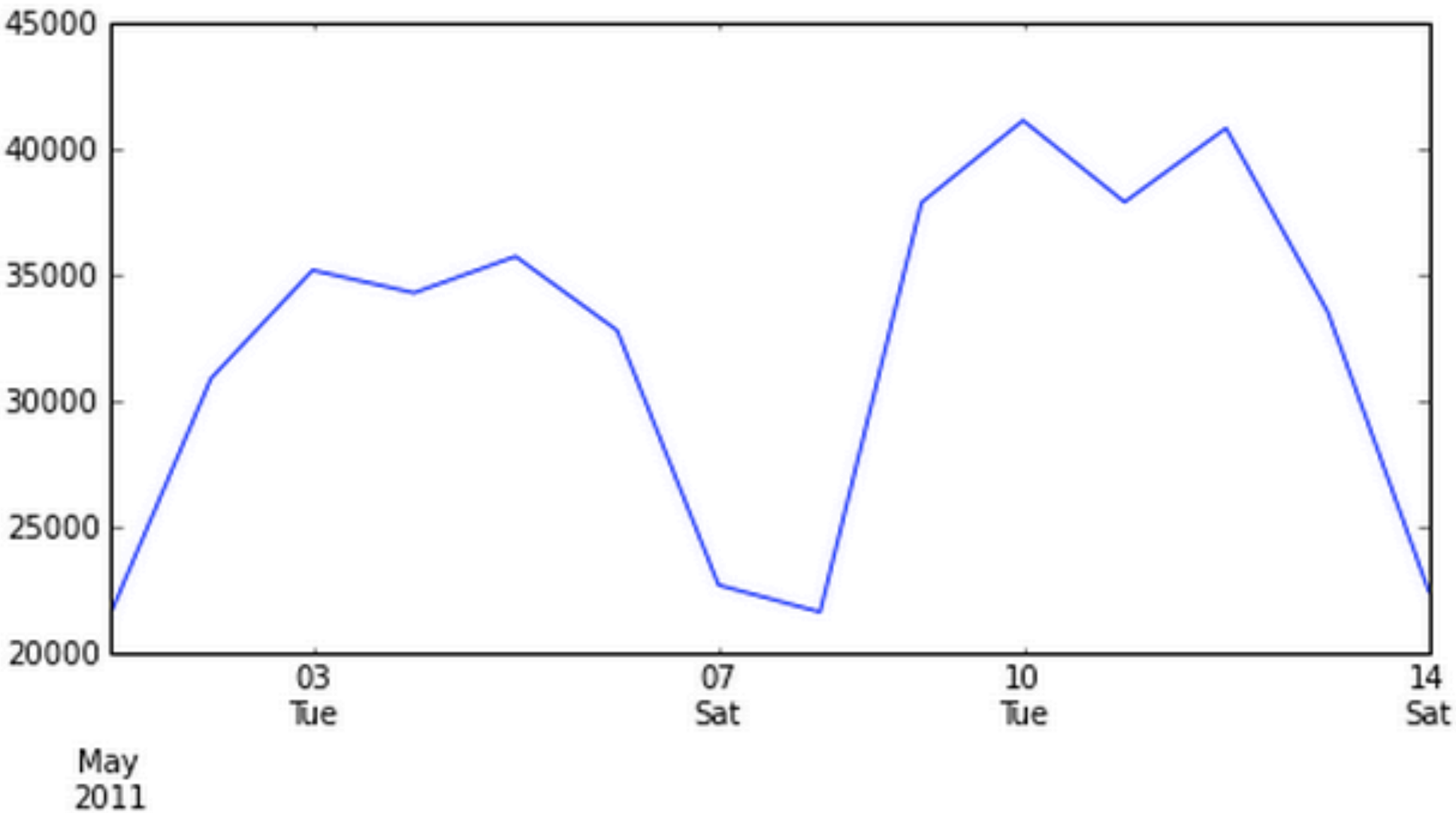
```
ax = plt.figure(figsize=(7,4), dpi=300).add_subplot(111)
two_weeks = science_tweets['2011-05-01':'2011-05-14']
ax.plot_date(two_weeks.index.to_pydatetime(), two_weeks, '-')
ax.xaxis.set_minor_locator(mpl.dates.WeekdayLocator(byweekday=(1,5),
                                                    interval=1))
ax.xaxis.set_minor_formatter(mpl.dates.DateFormatter('%d\n%a'))
ax.xaxis.set_major_locator(mpl.dates.MonthLocator())
ax.xaxis.set_major_formatter(mpl.dates.DateFormatter('\n\n\n%b\n%Y'))
plt.tight_layout()
plt.show()
```



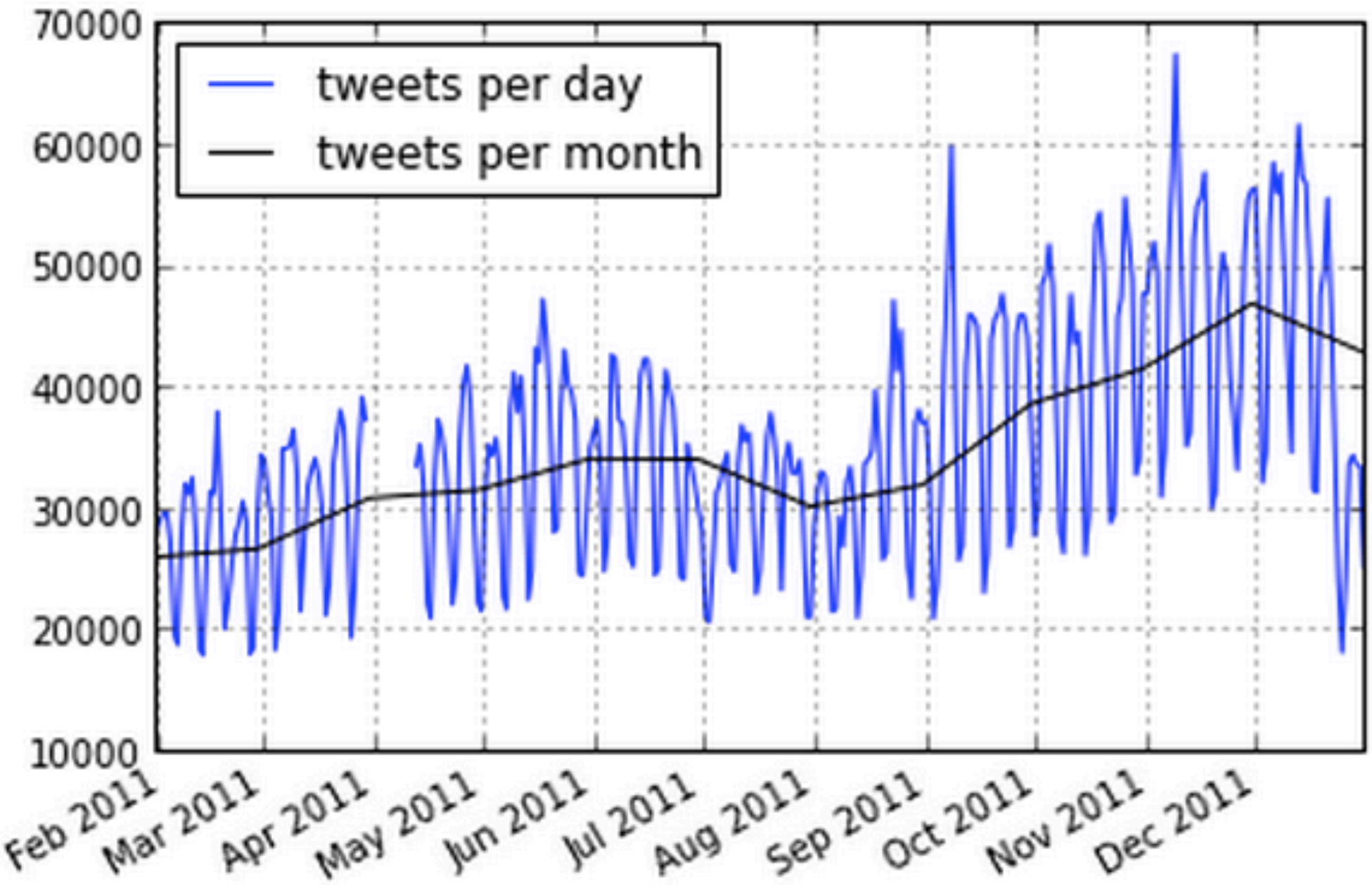
```
ax = plt.figure(figsize=(7,4), dpi=300).add_subplot(111)
two_weeks = science_tweets['2011-05-01':'2011-05-14']
ax.plot_date(two_weeks.index.to_pydatetime(), two_weeks, '-')
ax.xaxis.set_minor_locator(mpl.dates.WeekdayLocator(byweekday=(1,5),
                                                    interval=1))
ax.xaxis.set_minor_formatter(mpl.dates.DateFormatter('%d\n%a'))
ax.xaxis.set_major_locator(mpl.dates.MonthLocator())
ax.xaxis.set_major_formatter(mpl.dates.DateFormatter('\n\n\n%b\n%Y'))
plt.tight_layout()
plt.show()
```



```
ax = plt.figure(figsize=(7,4), dpi=300).add_subplot(111)
two_weeks = science_tweets['2011-05-01':'2011-05-14']
ax.plot_date(two_weeks.index.to_pydatetime(), two_weeks, '-')
ax.xaxis.set_minor_locator(mpl.dates.WeekdayLocator(byweekday=(1,5),
                                                    interval=1))
ax.xaxis.set_minor_formatter(mpl.dates.DateFormatter('%d\n%a'))
ax.xaxis.set_major_locator(mpl.dates.MonthLocator())
ax.xaxis.set_major_formatter(mpl.dates.DateFormatter('\n\n\n%b\n%Y'))
plt.tight_layout()
plt.show()
```



```
monthly_tweets = science_tweets.resample('M', how='mean')
science_tweets.plot(label='tweets per day')
monthly_tweets.plot('tweets per month', color='k')
legend(loc=0);
```



```
science_tweets.describe()
```

```
count       352.000000
mean      34646.264205
std       10178.287440
min       11070.000000
25%       27626.750000
50%       33414.500000
75%       40963.500000
max       67375.000000
dtype: float64
```

```
df = pandas.DataFrame(data={'tweets per day':science_tweets,
                            'tweets per month':monthly_tweets})
df[28:33]
```

	tweets per day	tweets per month
2011-01-29	18913	NaN
2011-01-30	18651	NaN
2011-01-31	27853	25897.903226
2011-02-01	28808	NaN
2011-02-02	29517	NaN

```
df['cumulative'] = df['tweets per day'].cumsum()
total_tweets = df['tweets per day'].sum()
df['percent'] = df['tweets per day'].map(lambda x:
                                         x*100/total_tweets)
df['dayofweek'] = df.index.map(lambda x: x.dayofweek)
df.head()
```

	tweets per day	tweets per month	cumulative	percent	dayofweek
2011-01-01	11070	NaN	11070	0.090771	5
2011-01-02	14542	NaN	25612	0.119241	6
2011-01-03	24121	NaN	49733	0.197786	0
2011-01-04	25984	NaN	75717	0.213062	1
2011-01-05	27639	NaN	103356	0.226633	2

```
df['cumulative'] = df['tweets per day'].cumsum()
total_tweets = df['tweets per day'].sum()
df['percent'] = df['tweets per day'].map(lambda x:
                                         x*100/total_tweets)
df['dayofweek'] = df.index.map(lambda x: x.dayofweek)
df.head()
```

	tweets per day	tweets per month	cumulative	percent	dayofweek
2011-01-01	11070	NaN	11070	0.090771	5
2011-01-02	14542	NaN	25612	0.119241	6
2011-01-03	24121	NaN	49733	0.197786	0
2011-01-04	25984	NaN	75717	0.213062	1
2011-01-05	27639	NaN	103356	0.226633	2

```
df['cumulative'] = df['tweets per day'].cumsum()
total_tweets = df['tweets per day'].sum()
df['percent'] = df['tweets per day'].map(lambda x:
                                         x*100/total_tweets)
df['dayofweek'] = df.index.map(lambda x: x.dayofweek)
df.head()
```

	tweets per day	tweets per month	cumulative	percent	dayofweek
2011-01-01	11070	NaN	11070	0.090771	5
2011-01-02	14542	NaN	25612	0.119241	6
2011-01-03	24121	NaN	49733	0.197786	0
2011-01-04	25984	NaN	75717	0.213062	1
2011-01-05	27639	NaN	103356	0.226633	2

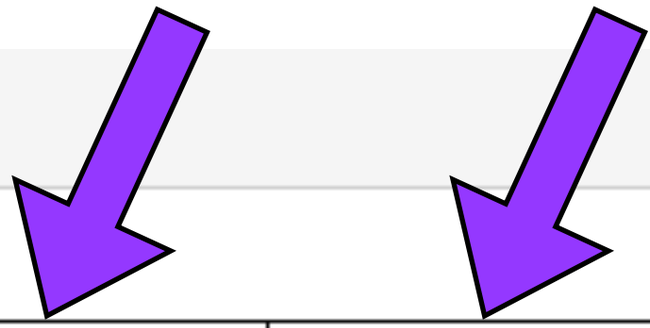
```
df['cumulative'] = df['tweets per day'].cumsum()
total_tweets = df['tweets per day'].sum()
df['percent'] = df['tweets per day'].map(lambda x:
                                         x*100/total_tweets)
df['dayofweek'] = df.index.map(lambda x: x.dayofweek)
df.head()
```

	tweets per day	tweets per month	cumulative	percent	dayofweek
2011-01-01	11070	NaN	11070	0.090771	5
2011-01-02	14542	NaN	25612	0.119241	6
2011-01-03	24121	NaN	49733	0.197786	0
2011-01-04	25984	NaN	75717	0.213062	1
2011-01-05	27639	NaN	103356	0.226633	2

```
df.groupby('dayofweek').sum()
```

	tweets per day	tweets per month	cumulative	percent
dayofweek				
0	1833426	93883.930876	274073252	15.033646
1	2010642	33999.741935	276083894	16.486774
2	2084425	78640.716129	280636847	17.091776
3	2012464	64687.800000	280180783	16.501714
4	1758167	38536.800000	281938950	14.416540
5	1233825	74176.073477	283172775	10.117064
6	1262536	30059.516129	272239826	10.352487

```
df.groupby('dayofweek').sum()
```



	tweets per day	tweets per month	cumulative	percent
dayofweek				
0	1833426	93883.930876	274073252	15.033646
1	2010642	33999.741935	276083894	16.486774
2	2084425	78640.716129	280636847	17.091776
3	2012464	64687.800000	280180783	16.501714
4	1758167	38536.800000	281938950	14.416540
5	1233825	74176.073477	283172775	10.117064
6	1262536	30059.516129	272239826	10.352487

```
df.groupby('dayofweek').sum()[['tweets per day', 'percent']]
```

	tweets per day	percent
dayofweek		
0	1833426	15.033646
1	2010642	16.486774
2	2084425	17.091776
3	2012464	16.501714
4	1758167	14.416540
5	1233825	10.117064
6	1262536	10.352487

```
df.groupby('dayofweek').sum()['percent'].sum()
```

```
100.0
```

```
print df.groupby('dayofweek').sum()['tweets per day'].sum()
print df['tweets per day'].sum()
```

```
12195485.0
12195485.0
```

```
df2 = df.groupby('dayofweek').sum()
print df2.to_latex(columns=['tweets per day', 'percent'])
```

```
\begin{tabular}{lrr}
\toprule
{} &  tweets per day &     percent \\
\midrule
dayofweek &                 &             \\
0         &         1833426 &   15.033646 \\
1         &         2010642 &   16.486774 \\
2         &         2084425 &   17.091776 \\
3         &         2012464 &   16.501714 \\
4         &         1758167 &   14.416540 \\
5         &         1233825 &   10.117064 \\
6         &         1262536 &   10.352487 \\
\bottomrule
\end{tabular}
```

Questions?

Slides and notebook available on GitHub

https://github.com/brendam/pyconnz2013talk

@brendam