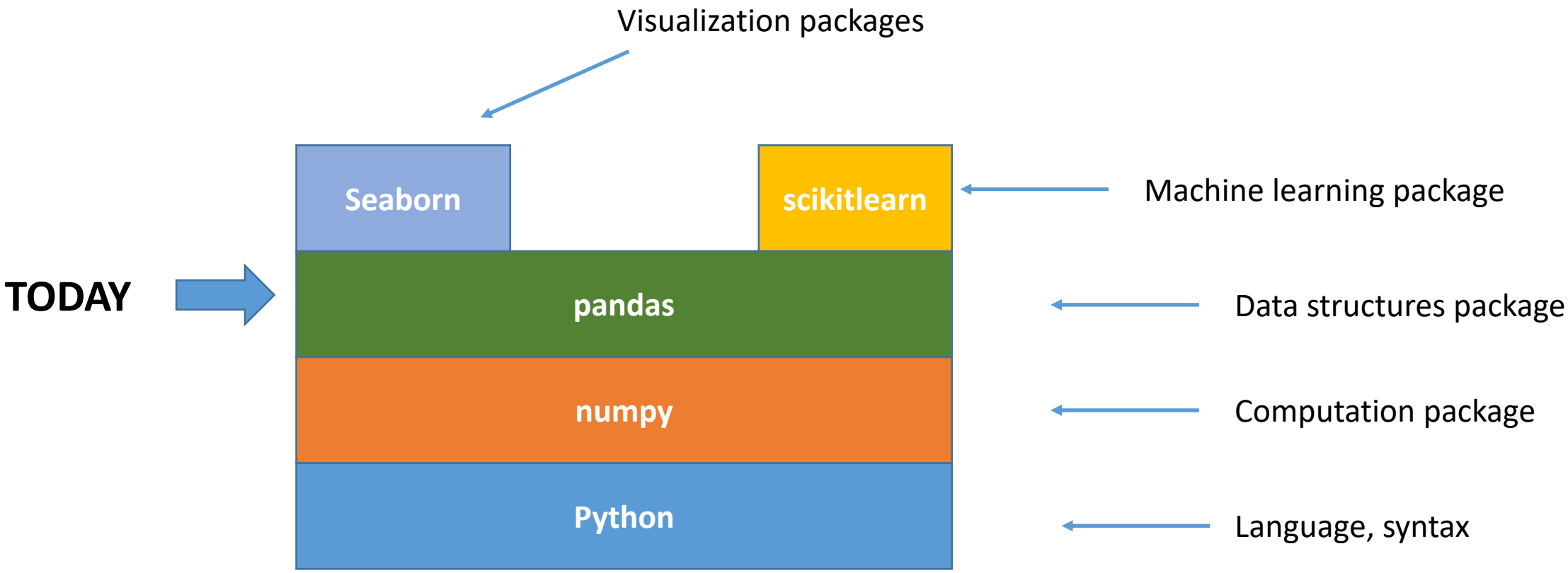
Univariate Time Series

module 3

This course


Visualization packages
Seaborn
scikitlearn
Machine learning package
TODAY
pandas
Data structures package
numpy
Computation package
Python
Language, syntax

Univariate Time Series

- Definition from here: *a sequence of measurements of the same variable collected over time*. Examples: stock prices, demand, housing prices.
- In pandas a univariate time series is a Series object where the index is a "timestamp".

Today's data

- Today we will look at two time series:
  - The price of Alphabet shares (GOOGL)
  - The median housing price in Santa Clara

- Our goals are:
  1. Practice data manipulation with Series
  2. Assess correlation between stock market and housing prices in the Bay Area

Get Stock Historical Data

- You can get the Google stock historical price data from https://finance.yahoo.com/quote/GOOGL/history?p=GOOGL
- In *'Historical Data'* tab, set the *'Time Period'* to Max, *'Apply'*, then *'Download Data'.*
- A *.csv* file will be downloaded into your local computer. Rename the file to *GOOGL.csv*

Take a look at `stock`

- This Series object has one entry for each trading day. The index (of type *datetime64*) is the day and the value (of type *float64*) is the closing price.

```
stock.head(10)
```

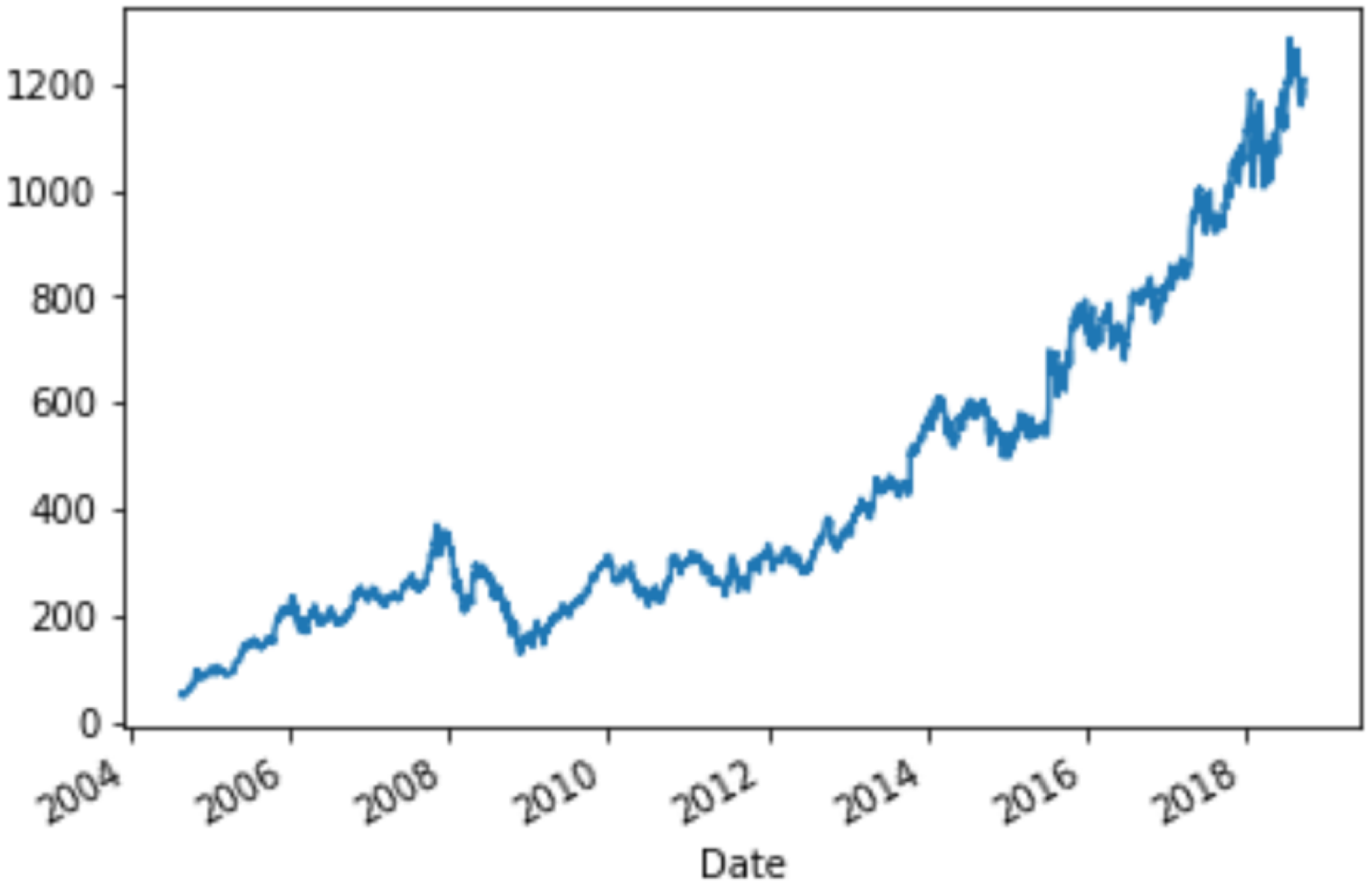
```
Date
2004-08-19    50.22
2004-08-20    54.21
2004-08-23    54.75
2004-08-24    52.49
2004-08-25    53.05
2004-08-26    54.01
2004-08-27    53.13
2004-08-30    51.06
2004-08-31    51.24
2004-09-01    50.18
Name: GOOGL, dtype: float64
```

```
stock.tail(10)
```

```
Date
2018-09-18    1167.109985
2018-09-19    1174.270020
2018-09-20    1191.569946
2018-09-21    1172.119995
2018-09-24    1179.560059
2018-09-25    1193.890015
2018-09-26    1194.060059
2018-09-27    1207.359985
2018-09-28    1207.079956
2018-10-01    1208.530029
Name: Close, dtype: float64
```

GOOGL price

```
%pylab inline
stock.plot()
```



Some problems

1. What day had the largest stock price?
    1. In class
2. What are the 10 days with largest stock price? Report both the day and the price.
    1. In class
3. How much profit (%) would we make if we bought at the beginning and sold everything on the last day? Do not type in any date.
    1. In class
4. What is the moving average of the price at each trading session? Use a 50-trading-days window. *Hint*: Explore the method *rolling*
    1. In class
5. Consider this investment strategy: buy on day x and then sell after 5 days (on day x+5). Find the expected profit (in %) of this operation. *Hint*: explore the method *shift*
    1. In class
6. Consider this investment strategy: buy whenever the price goes above the 20-day moving average, and then sell after 5 trading sessions. How much profit (in %) would we make on average?
    1. In class

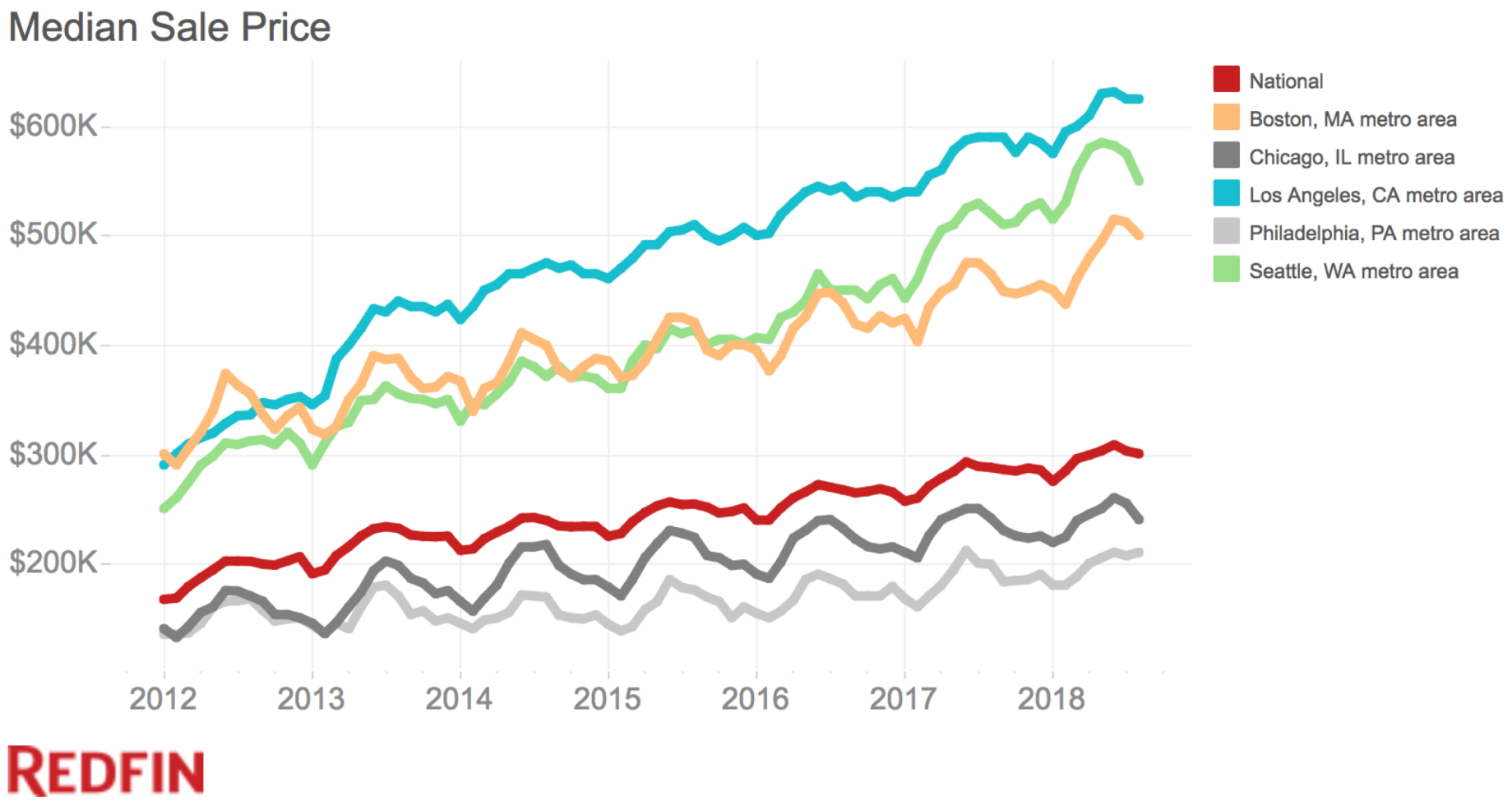
Housing Prices

- The file *santaclara_sfh.csv* was downloaded from redfin.com.
  - More data in https://www.redfin.com/blog/data-center
- For each end of month day, this file reports the median price of a single family home (sfh) in zip code 95050 over the previous 3 months.

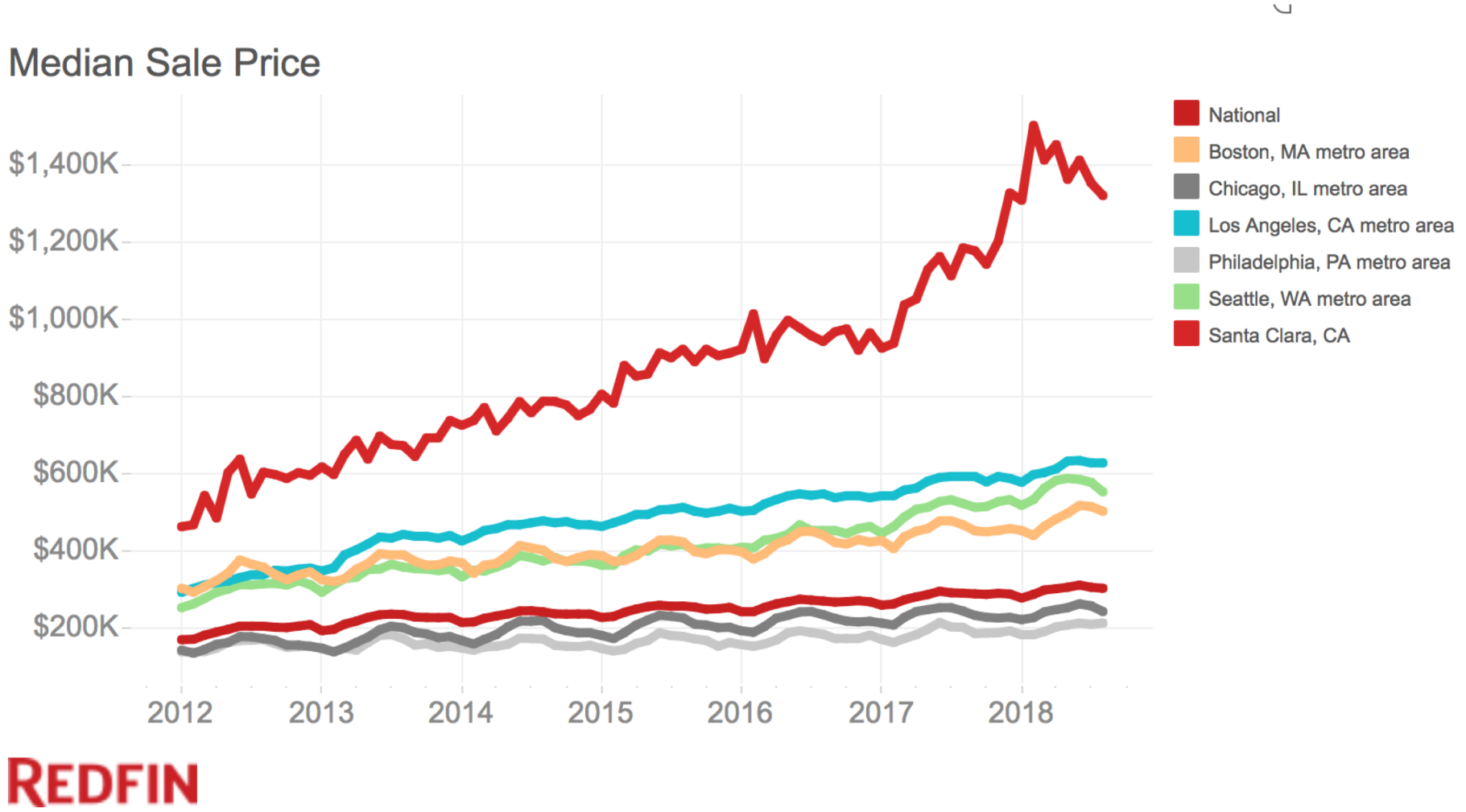
```
date
2012-01-31    516000.0
2012-02-29    520000.0
2012-03-31    520000.0
2012-04-30    520000.0
2012-05-31    524000.0
Name: housing, dtype: float64
```

Median price of sales between 2012-02-01 and 2012-04-30

Data from Redfin.com

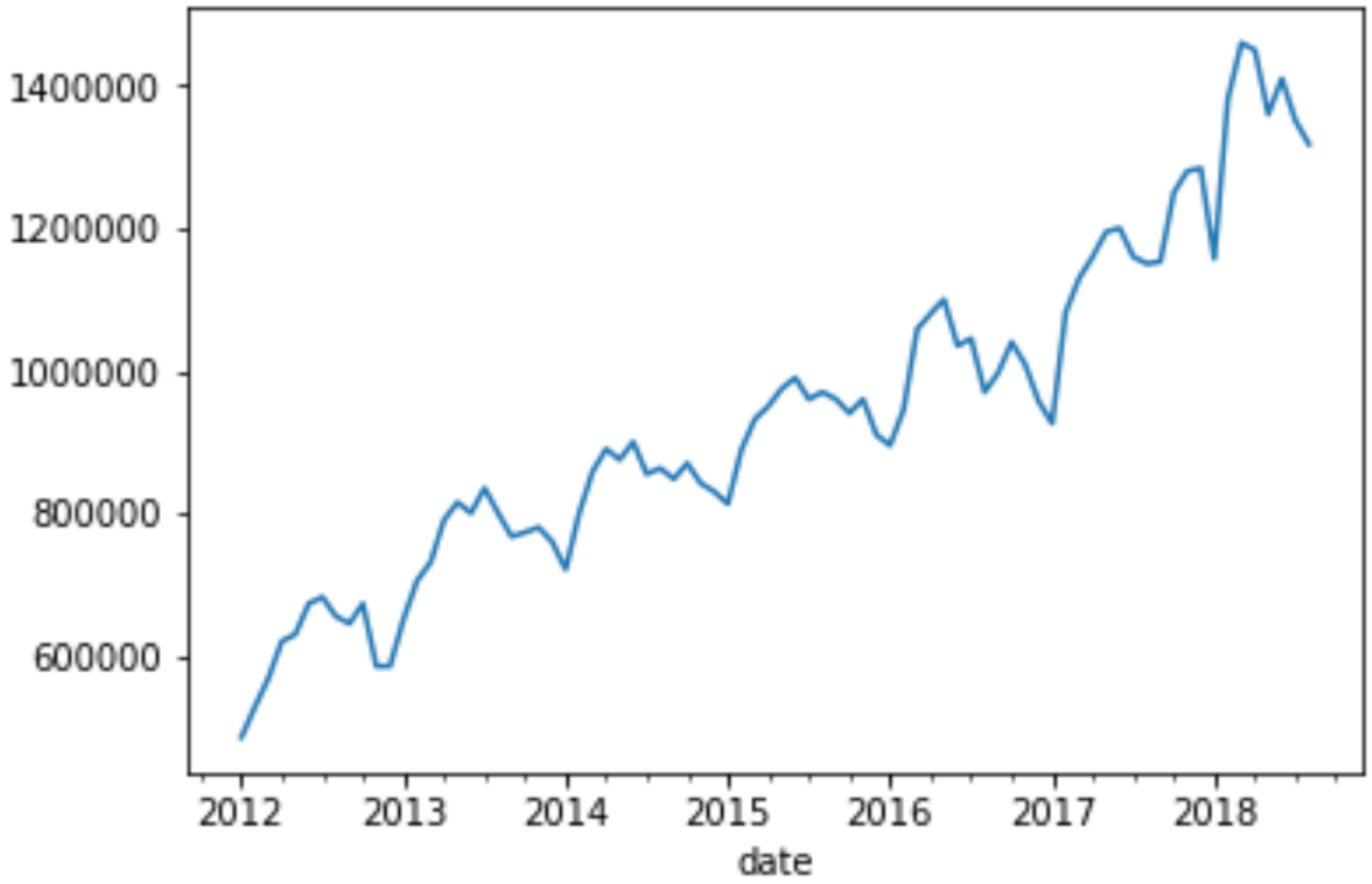


Data from Redfin.com (with Santa Clara, CA)



Housing Prices – Santa Clara County

```
%pylab inline
housing.plot()
```



Are housing prices correlated to Alphabet stock prices?

- To answer this question, we first need to align the two Series. The Series *housing* has one try for each ending day of **each month** and a value that is the 3-month moving median; the series *stock* has one entry for each **trading day** and the value that is the closing price.
- We will do the following:
  1. "Pad" *stock* so that there are no missing days (currently, holidays are missing)
  2. Compute the 3-month moving median of *stock*
  3. Retain only the end-of-month days in *stock*

1. Add missing days with **asfreq**

```
stock.tail(10)
```

```
Date
2017-09-13    950.440002
2017-09-14    940.130005
2017-09-15    935.289978
2017-09-18    929.750000
2017-09-19    936.859985
2017-09-20    947.539978
2017-09-21    947.549988
2017-09-22    943.260010
2017-09-25    934.280029
2017-09-26    937.429993
Name: Close, dtype: float64
```

```
padded = stock.asfreq('1D',method='ffill')
```

The method **asfreq** changes the frequency of the series (in this case to 1 day) and fill the holes by propagating forward (ffill) or backward (bfill)

```
padded.tail(10)
```

```
Date
2017-09-17    935.289978
2017-09-18    929.750000
2017-09-19    936.859985
2017-09-20    947.539978
2017-09-21    947.549988
2017-09-22    943.260010
2017-09-23    943.260010
2017-09-24    943.260010
2017-09-25    934.280029
2017-09-26    937.429993
Freq: D, Name: Close, dty
```

3. Retain the same days as in the housing Series

```
movmed.tail(10)
```

```
Date
2017-09-17    944.885010
2017-09-18    944.230011
2017-09-19    943.910004
2017-09-20    943.910004
2017-09-21    943.910004
2017-09-22    943.459992
2017-09-23    943.274994
2017-09-24    943.260010
2017-09-25    943.260010
2017-09-26    942.920013
Freq: D, Name: Close, dty
```

```
mod_stock = movmed[housing.index]
```

```
mod_stock.tail(10)
```

```
date
2016-11-30    800.970001
2016-12-31    803.580017
2017-01-31    807.799988
2017-02-28    824.790008
2017-03-31    841.399994
2017-04-30    847.809998
2017-05-31    865.244995
2017-06-30    954.990021
2017-07-31    964.959992
2017-08-31    953.020019
Name: Close, dtype: float
```

Correlation Housing Prices vs GOOGL

```
housing.corr(mod_stock)
```

```
0.9591267279580493 2
```

REALLY HIGH!

```
norm_housing = housing / housing[0]
```

```
norm_stock = mod_stock / mod_stock[0]
```

```
import matplotlib.pyplot as plt

plt.plot(norm_housing,'r') # r for "red"
plt.plot(norm_stock,'b')  # b for "blue"
plt.legend(loc=2)
plt.show()
```

Normalized at Feb 2012