Time Series Analysis with Python

Aileen Nielsen

July, 12, 2016

aileen.a.nielsen@gmail.com

Installation instructions

- Please install Conda per ‘quick install’ instructions: http://conda.pydata.org/docs/install/quick.html
- Make sure you have the following packages installed:
  - pandas
  - numpy
  - Statsmodels
  - scikit-learn
  - scipy
- These would be good to have but are not essential:
  - pytz
  - hmm

Dealing with time

Pandas functionality

- Generate sequences of fixed-frequency dates and time spans
- Conform or convert time series to a particular frequency
- Compute ’relative’ dates based on various non-standard time increments (e.g. 5 business days before the last day of the year) or ’roll’ dates backward and forward

http://pandas.pydata.org/pandas-docs/stable/timeseries.html

Date/Time components

Time/Date Components

There are several time/date properties that one can access from `Timestamp` or a collection of timestamps like a `DateTimeIndex`.

Property	Description
year	The year of the datetime
month	The month of the datetime
day	The days of the datetime
hour	The hour of the datetime
minute	The minutes of the datetime
second	The seconds of the datetime
microsecond	The microseconds of the datetime
nanosecond	The nanoseconds of the datetime
date	Returns datetime.date
time	Returns datetime.time
dayofyear	The ordinal day of year
weekofyear	The week ordinal of the year
week	The week ordinal of the year
dayofweek	The numer of the day of the week with Monday=0, Sunday=6
weekday	The number of the day of the week with Monday=0, Sunday=6
weekday_name	The name of the day in a week (ex: Friday)
quarter	Quarter of the date: Jan=Mar = 1, Apr-Jun = 2, etc.
days_in_month	The number of days in the month of the datetime
is_month_start	Logical indicating if first day of month (defined by frequency)
is_month_end	Logical indicating if last day of month (defined by frequency)
is_quarter_start	Logical indicating if first day of quarter (defined by frequency)
is_quarter_end	Logical indicating if last day of quarter (defined by frequency)
is_year_start	Logical indicating if first day of year (defined by frequency)
is_year_end	Logical indicating if last day of year (defined by frequency)

http://pandas.pydata.org/pandas-docs/stable/timeseries.html

Offset Aliases

A number of string aliases are given to useful common time series frequencies. We will refer to these aliases as *offset aliases* (referred to as *time rules* prior to v0.8.0).

Alias	Description
B	business day frequency
C	custom business day frequency (experimental)
D	calendar day frequency
W	weekly frequency
M	month end frequency
BM	business month end frequency
CBM	custom business month end frequency
MS	month start frequency
BMS	business month start frequency
CBMS	custom business month start frequency
Q	quarter end frequency
BQ	business quarter endfrequency
QS	quarter start frequency
BQS	business quarter start frequency
A	year end frequency
BA	business year end frequency
AS	year start frequency
BAS	business year start frequency
BH	business hour frequency
H	hourly frequency
T, min	minutely frequency
S	secondly frequency
L, ms	milliseconds
U, us	microseconds
N	nanoseconds

http://pandas.pydata.org/pandas-docs/stable/timeseries.html

DateOffset objects

In the preceding examples, we created DatetimeIndex objects at various frequencies by passing in frequency strings like 'M', 'W', and 'BM to the `freq` keyword. Under the hood, these frequency strings are being translated into an instance of pandas `DateOffset`, which represents a regular frequency increment. Specific offset logic like "month", "business day", or "one hour" is represented in its various subclasses.

Class name	Description
DateOffset	Generic offset class, defaults to 1 calendar day
BDay	business day (weekday)
CDay	custom business day (experimental)
Week	one week, optionally anchored on a day of the week
WeekOfMonth	the x-th day of the y-th week of each month
LastWeekOfMonth	the x-th day of the last week of each month
MonthEnd	calendar month end
MonthBegin	calendar month begin
BMonthEnd	business month end
BMonthBegin	business month begin
CBMonthEnd	custom business month end
CBMonthBegin	custom business month begin
QuarterEnd	calendar quarter end
QuarterBegin	calendar quarter begin
BQuarterEnd	business quarter end
BQuarterBegin	business quarter begin
FY5253Quarter	retail (aka 52-53 week) quarter
YearEnd	calendar year end
YearBegin	calendar year begin
BYearEnd	business year end
BYearBegin	business year begin
FY5253	retail (aka 52-53 week) year
BusinessHour	business hour
CustomBusinessHour	custom business hour
Hour	one hour
Minute	one minute
Second	one second
Milli	one millisecond
Micro	one microsecond
Nano	one nanosecond

http://pandas.pydata.org/pandas-docs/stable/timeseries.html

Anchored offsets

Anchored Offsets

For some frequencies you can specify an anchoring suffix:

Alias	Description
W-SUN	weekly frequency (sundays). Same as 'W'
W-MON	weekly frequency (mondays)
W-TUE	weekly frequency (tuesdays)
W-WED	weekly frequency (wednesdays)
W-THU	weekly frequency (thursdays)
W-FRI	weekly frequency (fridays)
W-SAT	weekly frequency (saturdays)
(B)Q(S)-DEC	quarterly frequency, year ends in December. Same as 'Q'
(B)Q(S)-JAN	quarterly frequency, year ends in January
(B)Q(S)-FEB	quarterly frequency, year ends in February
(B)Q(S)-MAR	quarterly frequency, year ends in March

http://pandas.pydata.org/pandas-docs/stable/timeseries.html

Indices

DatetimeIndex

One of the main uses for `DatetimeIndex` is as an index for pandas objects. The `DatetimeIndex` class contains many timeseries related optimizations:

- A large range of dates for various offsets are pre-computed and cached under the hood in order to make generating subsequent date ranges very fast (just have to grab a slice)
- Fast shifting using the `shift` and `tshift` method on pandas objects
- Unioning of overlapping DatetimeIndex objects with the same frequency is very fast (important for fast data alignment)
- Quick access to date fields via properties such as `year`, `month`, etc.
- Regularization functions like `snap` and very fast `asof` logic

http://pandas.pydata.org/pandas-docs/stable/timeseries.html

Pandas time-relates data types

Overview

Following table shows the type of time-related classes pandas can handle and how to create them.

Class	Remarks	How to create
`Timestamp`	Represents a single time stamp	`to_datetime`, `Timestamp`
`DatetimeIndex`	Index of `Timestamp`	`to_datetime`, `date_range`, `DatetimeIndex`
`Period`	Represents a single time span	`Period`
`PeriodIndex`	Index of `Period`	`period_range`, `PeriodIndex`

http://pandas.pydata.org/pandas-docs/stable/timeseries.html

Holidays / Holiday Calendars

Holidays and calendars provide a simple way to define holiday rules to be used with `CustomBusinessDay` or in other analysis that requires a predefined set of holidays. The `AbstractHolidayCalendar` class provides all the necessary methods to return a list of holidays and only `rules` need to be defined in a specific holiday calendar class. Further, `start_date` and `end_date` class attributes determine over what date range holidays are generated. These should be overwritten on the `AbstractHolidayCalendar` class to have the range apply to all calendar subclasses. `USFederalHolidayCalendar` is the only calendar that exists and primarily serves as an example for developing other calendars.

For holidays that occur on fixed dates (e.g., US Memorial Day or July 4th) an observance rule determines when that holiday is observed if it falls on a weekend or some other non-observed day. Defined observance rules are:

Rule	Description
nearest_workday	move Saturday to Friday and Sunday to Monday
sunday_to_monday	move Sunday to following Monday
next_monday_or_tuesday	move Saturday to Monday and Sunday/Monday to Tuesday
previous_friday	move Saturday and Sunday to previous Friday"
next_monday	move Saturday and Sunday to following Monday

An example of how holidays and holiday calendars are defined:

http://pandas.pydata.org/pandas-docs/stable/timeseries.html

Thorns in your side

pandas.DataFrame.asfreq

`DataFrame.`**`asfreq`**(*freq*, *method=None*, *how=None*, *normalize=False*)

Convert all TimeSeries inside to specified frequency using DateOffset objects. Optionally provide fill method to pad/backfill missing values.

Parameters:	**freq** : *DateOffset object, or string* **method** : *{'backfill', 'bfill', 'pad', 'ffill', None}* Method to use for filling holes in reindexed Series pad / ffill: propagate last valid observation forward to next valid backfill / bfill: use NEXT valid observation to fill method **how** : *{'start', 'end'}, default end* For PeriodIndex only, see PeriodIndex.asfreq **normalize** : *bool, default False* Whether to reset output index to midnight
Returns:	**converted** : *type of caller*

pandas.DataFrame.resample

`DataFrame.`**`resample`**(*rule*, *how=None*, *axis=0*, *fill_method=None*, *closed=None*, *label=None*, *convention='start'*, *kind=None*, *loffset=None*, *limit=None*, *base=0*)

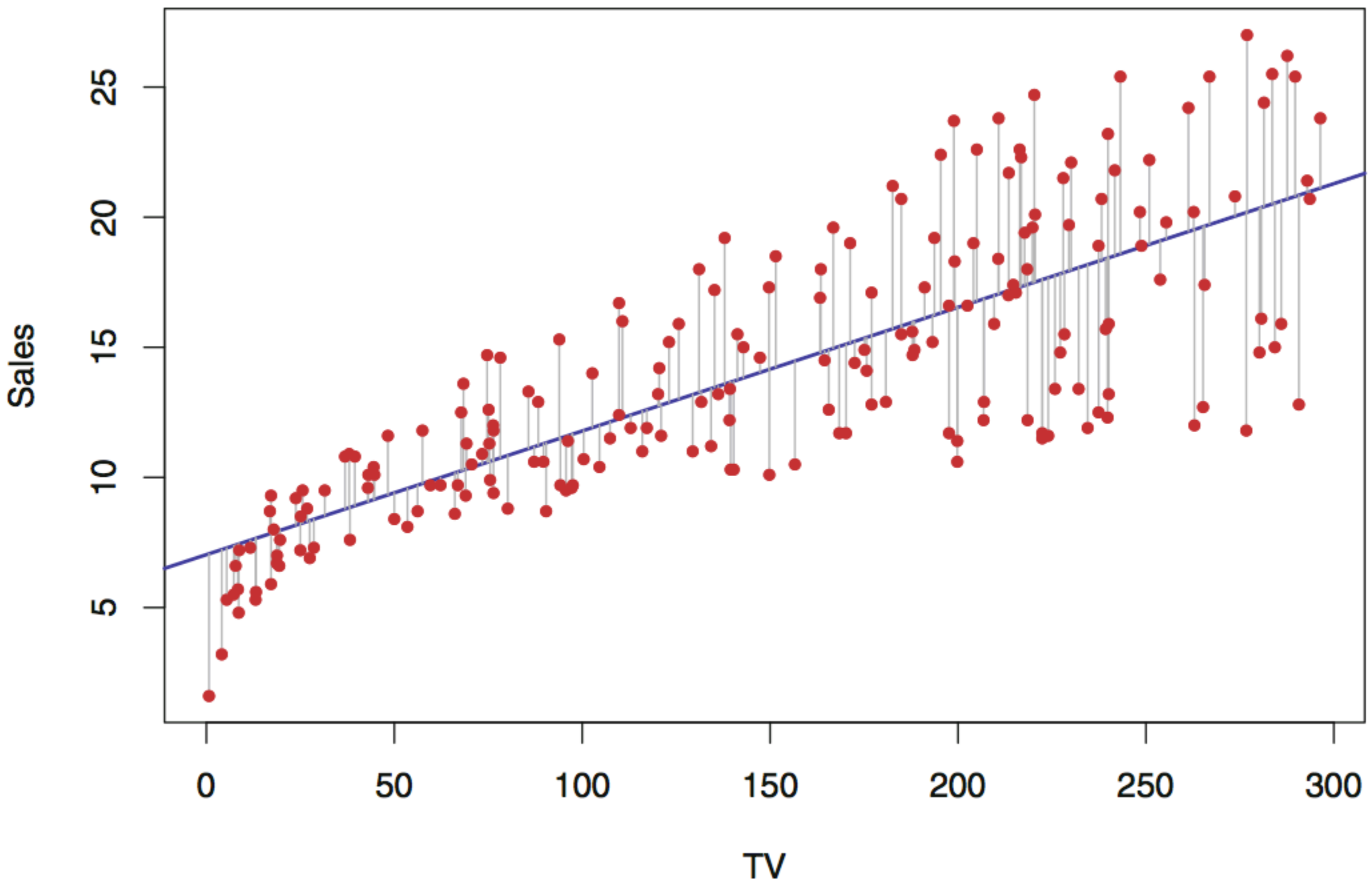
Convenience method for frequency conversion and resampling of regular time-series data.

Parameters:	**rule** : *string* the offset string or object representing target conversion **axis** : *int, optional, default 0* **closed** : *{'right', 'left'}* Which side of bin interval is closed **label** : *{'right', 'left'}* Which bin edge label to label bucket with **convention** : *{'start', 'end', 's', 'e'}* **loffset** : *timedelta* Adjust the resampled time labels **base** : *int, default 0* For frequencies that evenly subdivide 1 day, the "origin" of the aggregated intervals. For example, for '5min' frequency, base could range from 0 through 4. Defaults to 0

Examples

Linear regression

Linear regression intuition



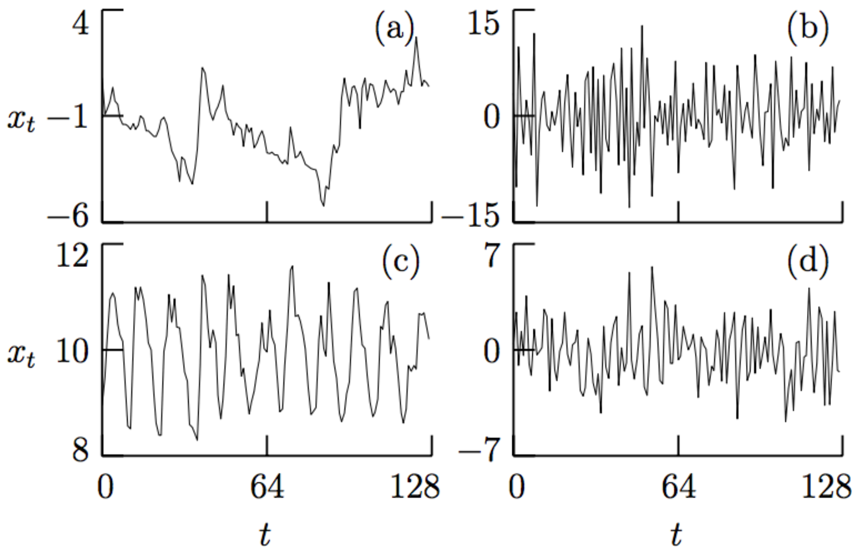
http://datascience.stackexchange.com/questions/6787/is-decision-tree-algorithm-a-linear-or-nonlinear-algorithm

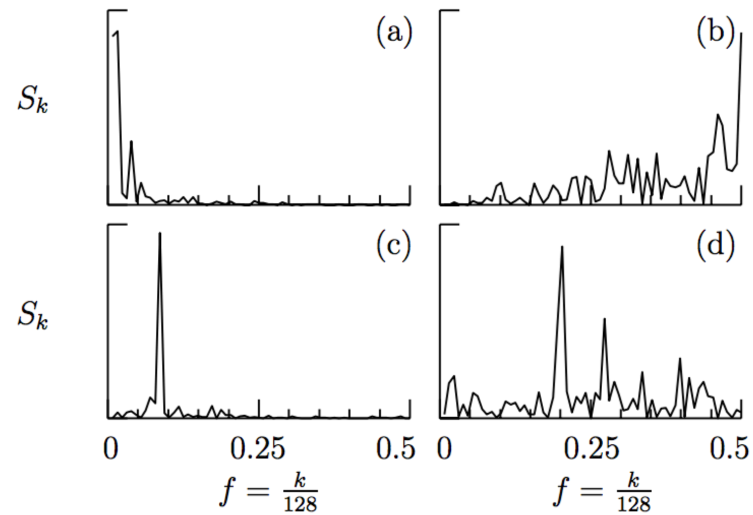
Spectral Analysis

Intuition

- Decompose a time series into a sum of many many sine or cosine functions
- The coefficients of these functions should have uncorrelated values
- Regression on sinusoids

Examples

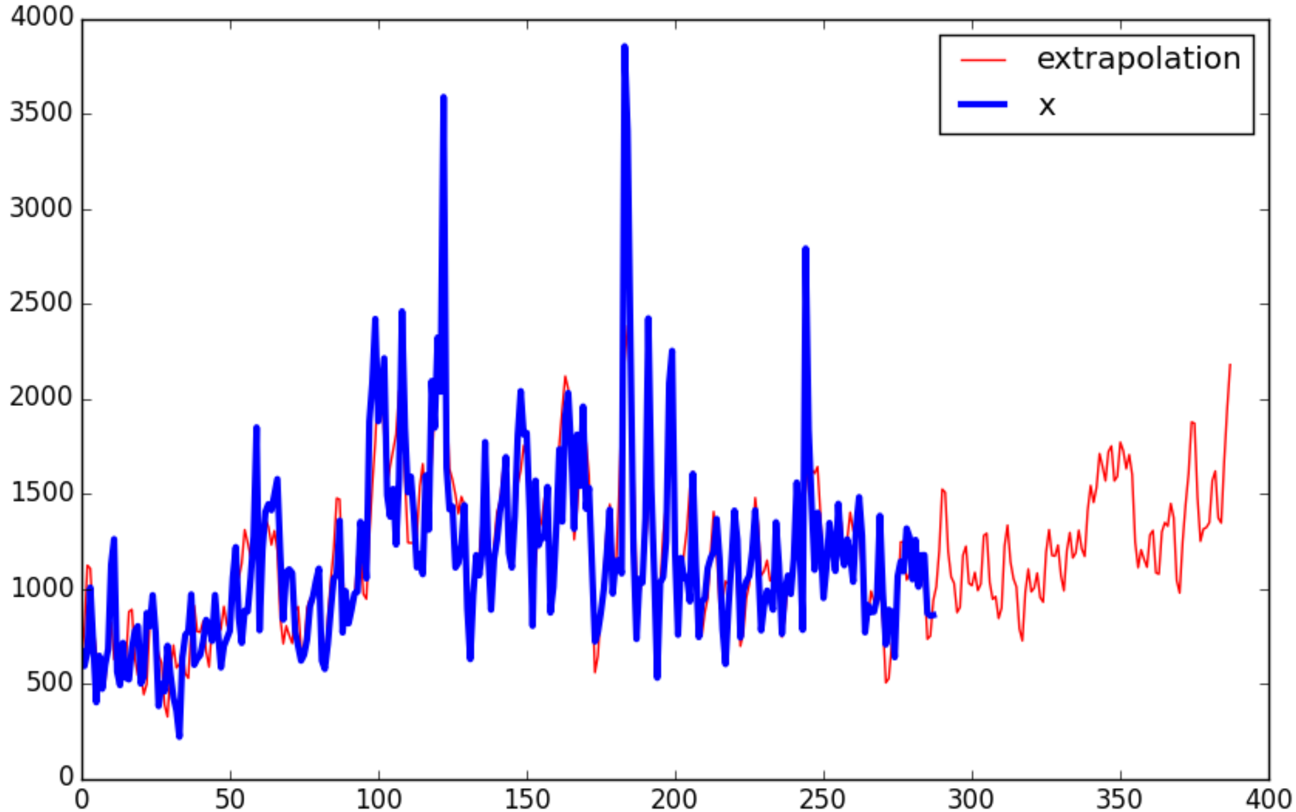




1. What are the advantages?
2. Potential disadvantages?
3. When would this provide useful information?
4. When would this *not* provide useful information?

https://faculty.washington.edu/dbp/PDFFILES/GHS-AP-Stat-talk.pdf

Fit can be surprisingly good



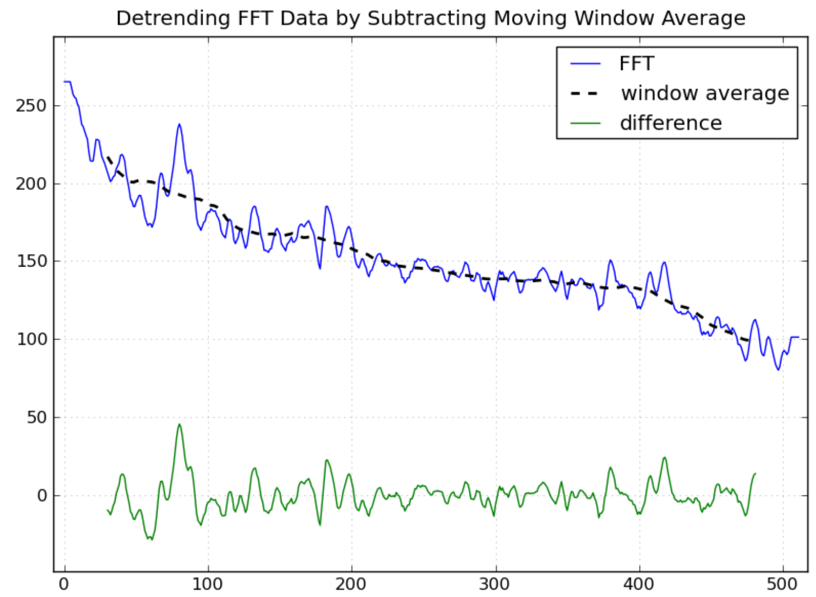
Pre-Prediction Munging

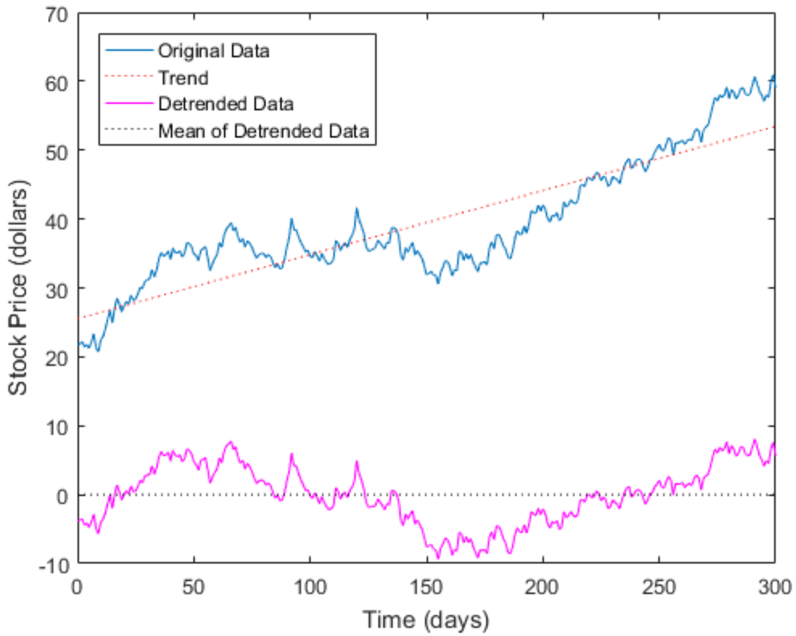
You need to remove the trend and seasonal elements before forecasting

- Most (interesting) data in the real world will show
  - Trends
  - Seasonality
- Most models require data that shows neither of these properties to say something interesting

De-trend your data

Use local smoothing or a linear regression



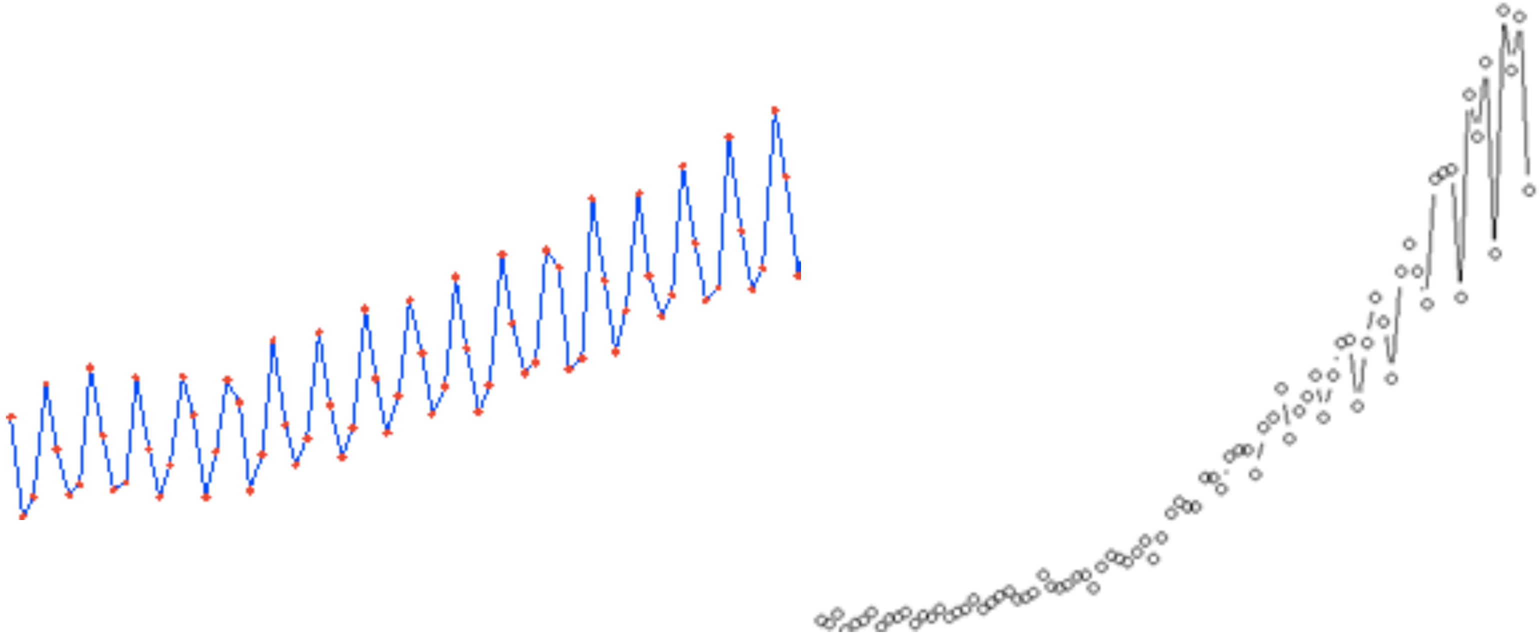


Remove seasonality

Look at power spectrum **after** removing trend data

- Simplest: average de-trended values for specific season
- More common: use ‘loess’ method (‘locally weighted scatterplot smoothing’)
  - Window of specified width is placed over the data
  - A weighted regression line or curve is fitted to the data, with points closest to center of curve having greatest weight
  - Weighting is reduced on points farthest from regression line/curve and calculation is rerun several times.
  - This yields 1 point on loess curve
  - Helps reduce impact of outlier points

Additive
vs.
multiplicative seasonality

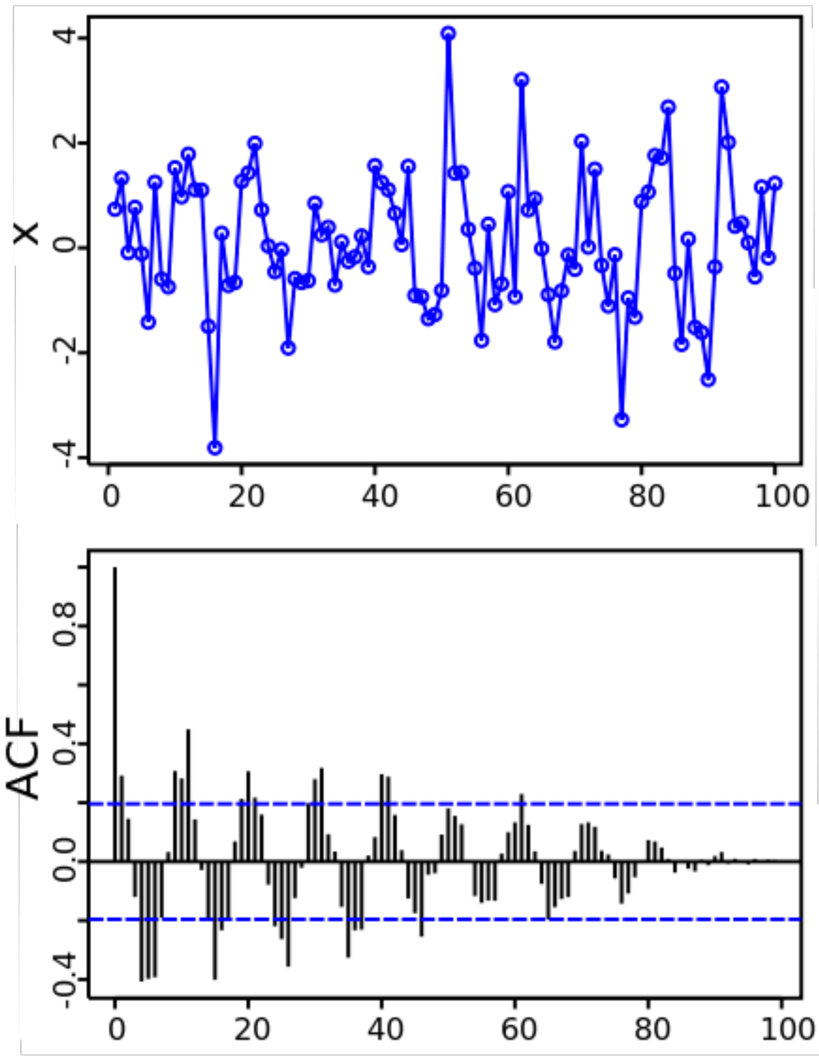


https://onlinecourses.science.psu.edu/stat510/node/57

Prediction

Autocorrelation function

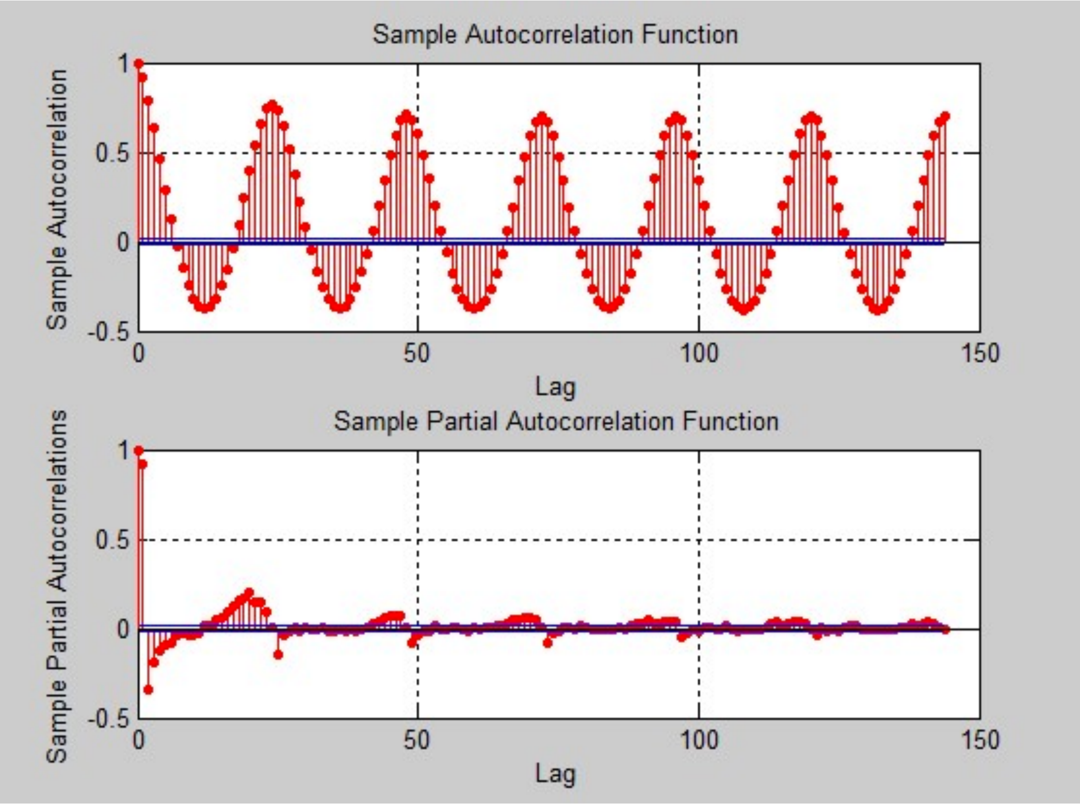
- Used to help identify possible structures of time series data
- Gives a sense of how different points in time relate to each other in a way explained by temporal distance



https://en.wikipedia.org/wiki/Autocorrelation

Partial autocorrelation function

- "gives the partial correlation of a time series with its own lagged values, controlling for the values of time series at all shorter lags"
- Why would this be useful?



https://en.wikipedia.org/wiki/Partial_autocorrelation_function &
https://www.mathworks.com/matlabcentral/answers/160904-interpretation-of-autocorrelation-function-to-determine-number-of-lags-in-ar-model?requestedDomain=www.mathworks.com

ARIMA model (a.k.a. Box-Jenkins)

- AR = autoregressive terms
- I = differencing
- MA = moving average
- Hence specified as (autoregressive terms, differencing terms, moving average terms)

ARIMA mode: ‘the most general class of models for forecasting a time series which can be **made to be ’stationary’**

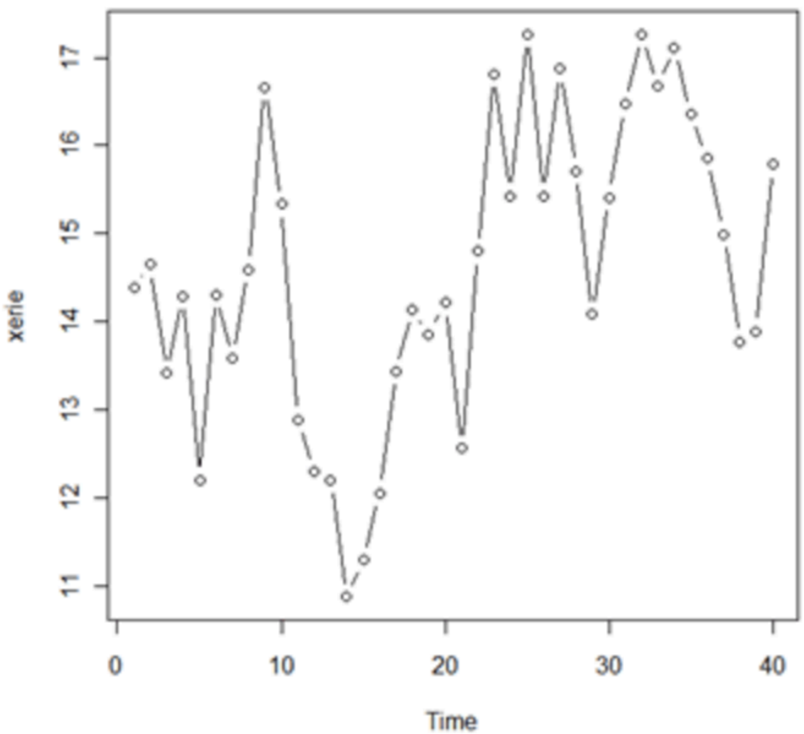
- Statistical properties (mean, variance) constant overt time
- ‘its short-term random time patterns always look the same in a statistical sense’
- Autocorrelation function & power spectrum remain constant over time
- Ok to do non-linear transformations to get there
- ARIMA model can be viewed as a combination of signal ad noise
- Extrapolate the signal to obtain forecasts

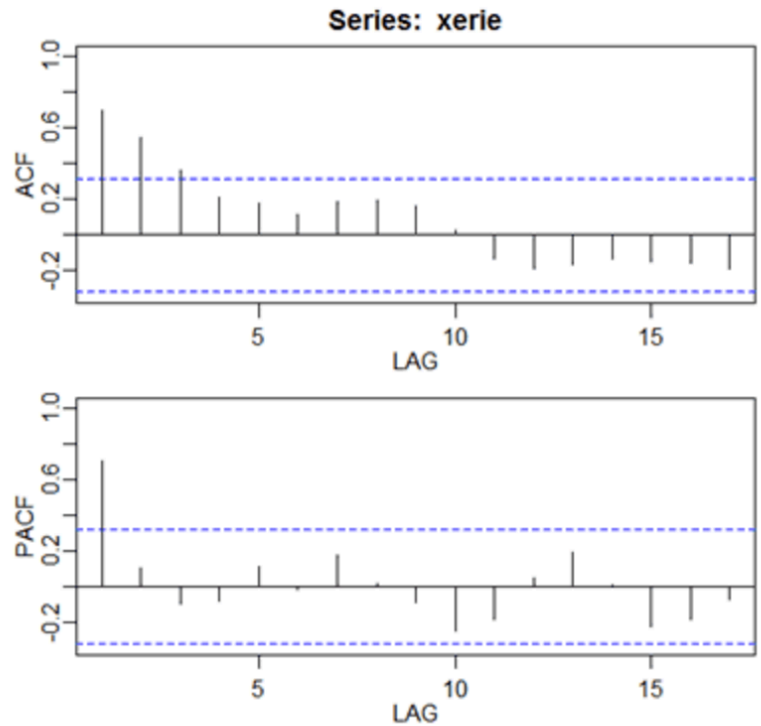
http://people.duke.edu/~rnau/411arim.htm

Applying the appropriate ARIMA model

- Need to determine what ARIMA model to use
- Use plot of the data, the ACF, and the PACF
- With the plot of the data: look for trend (linear or otherwise) & determine whether to transform data
- Most software will use a maximum likelihood estimation to determine appropriate ARIMA parameters

Example – Annual Lake Erie depth

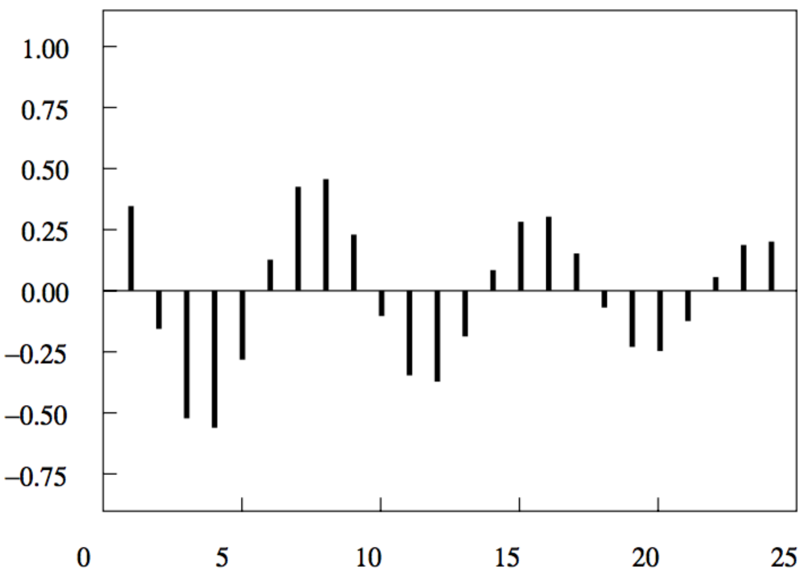


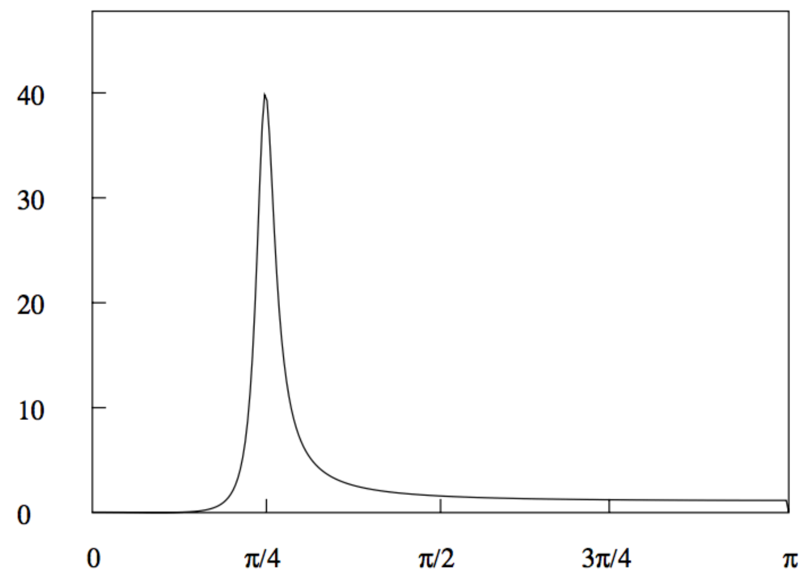


https://faculty.washington.edu/dbp/PDFFILES/GHS-AP-Stat-talk.pdf

Spectral analysis is helpful for ARIMA modeling

Here we see an ARMA(2,2) process and its spectral decomposition

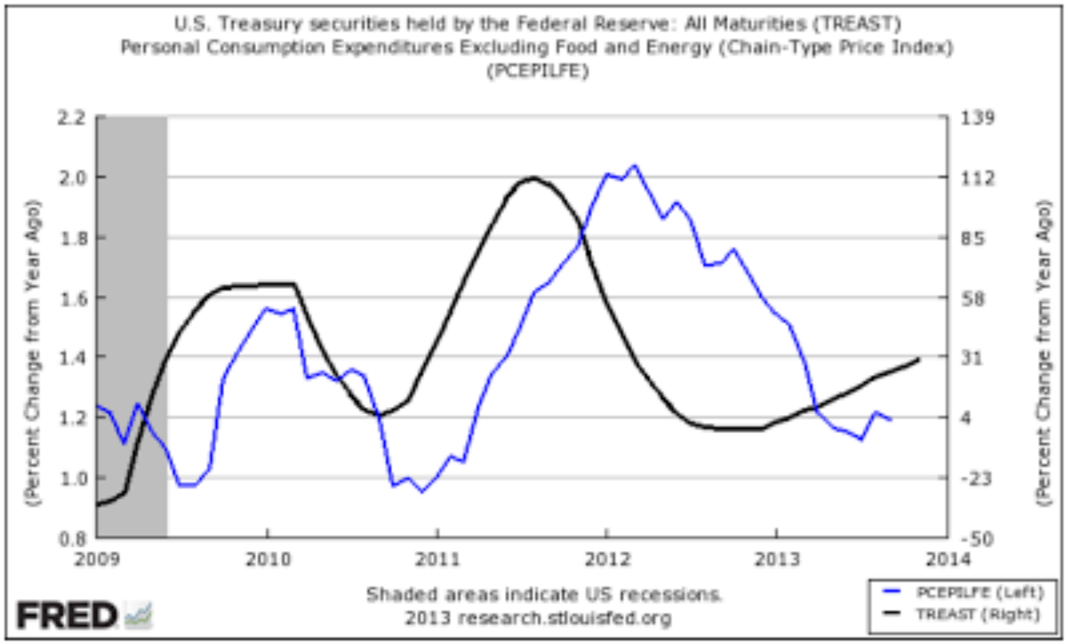




https://faculty.washington.edu/dbp/PDFFILES/GHS-AP-Stat-talk.pdf

Learn more…

Vector autoregression works similarly for cases of multivariate time series



$$x_{t,1} = \alpha_1 + \phi_{11} x_{t-1,1} + \phi_{12} x_{t-1,2} + \phi_{13} x_{t-1,3} + w_{t,1}$$

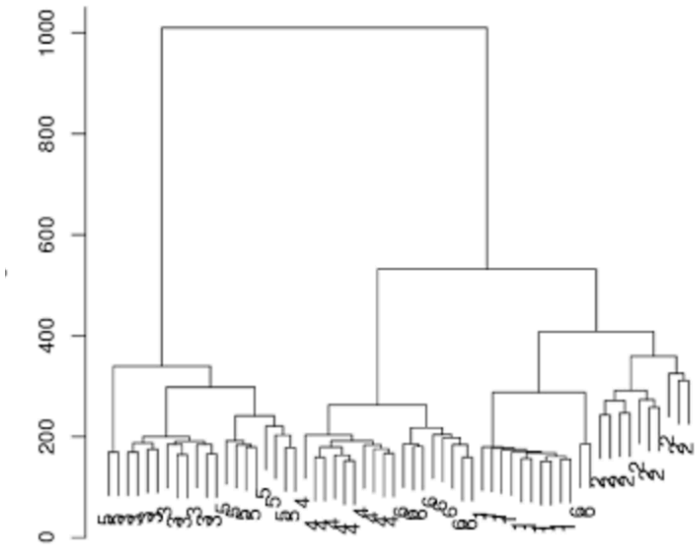
$$x_{t,2} = \alpha_2 + \phi_{21} x_{t-1,1} + \phi_{22} x_{t-1,2} + \phi_{23} x_{t-1,3} + w_{t,2}$$

$$x_{t,3} = \alpha_3 + \phi_{31} x_{t-1,1} + \phi_{32} x_{t-1,2} + \phi_{33} x_{t-1,3} + w_{t,3}$$

http://macromarketmusings.blogspot.com/2013/12/taking-model-to-data.html

Clustering & Classification
(yet another route to prediction)

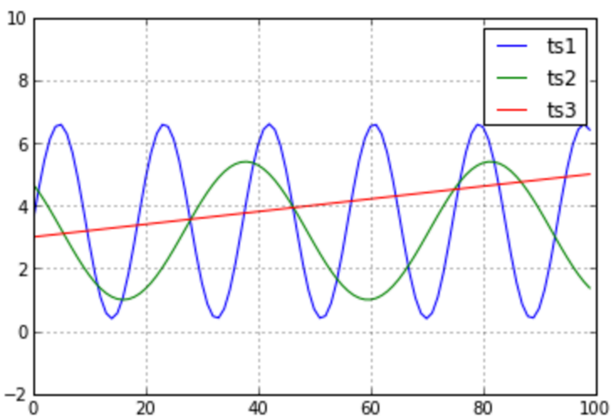
Clustering

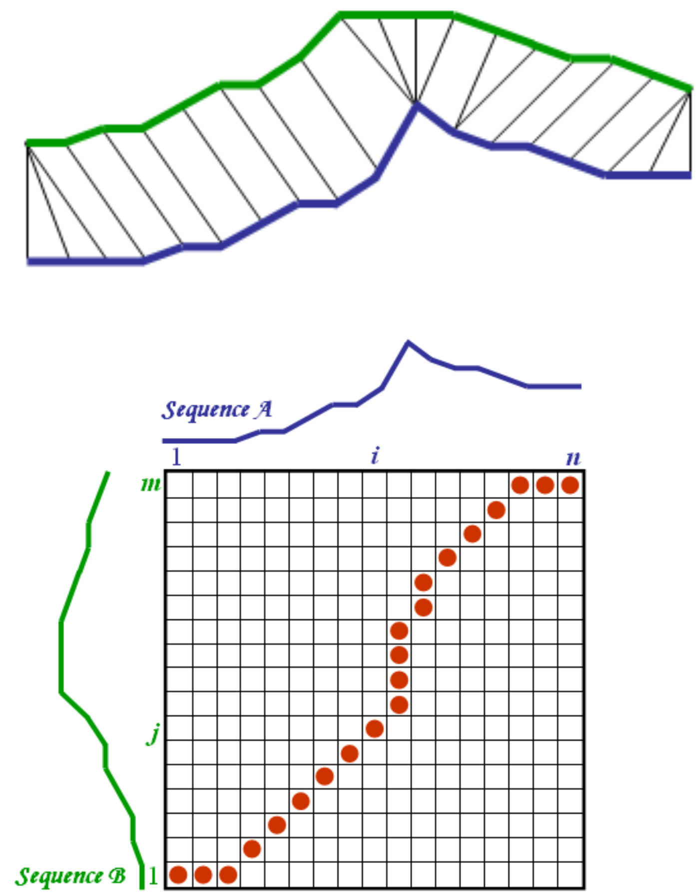


Choose an appropriate clustering technique based on what you know about the distance measure you use

http://www.rdatamining.com/examples/time-series-clustering-classification

Need to think carefully about distance metric





http://www.psb.ugent.be/cbd/papers/gentxwarper/DTWalgorithm.htm &
http://alexminnaar.com/time-series-classification-and-clustering-with-python.html

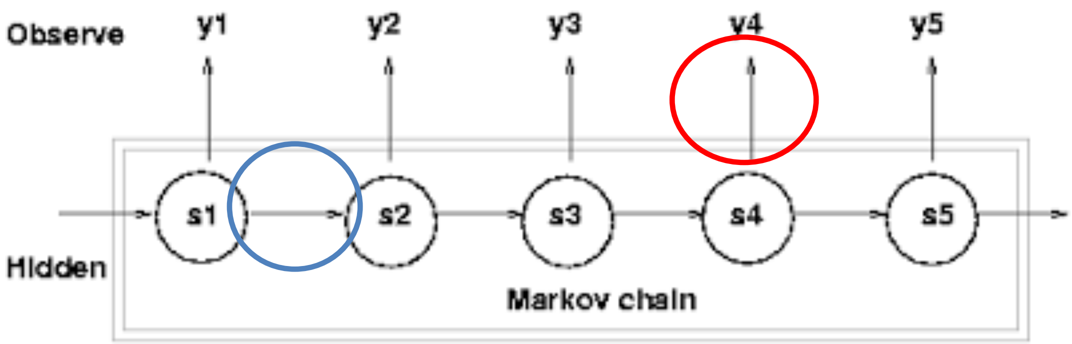
DTW-based Classification

DTW-based decision trees

DTW-based nearest neighbor

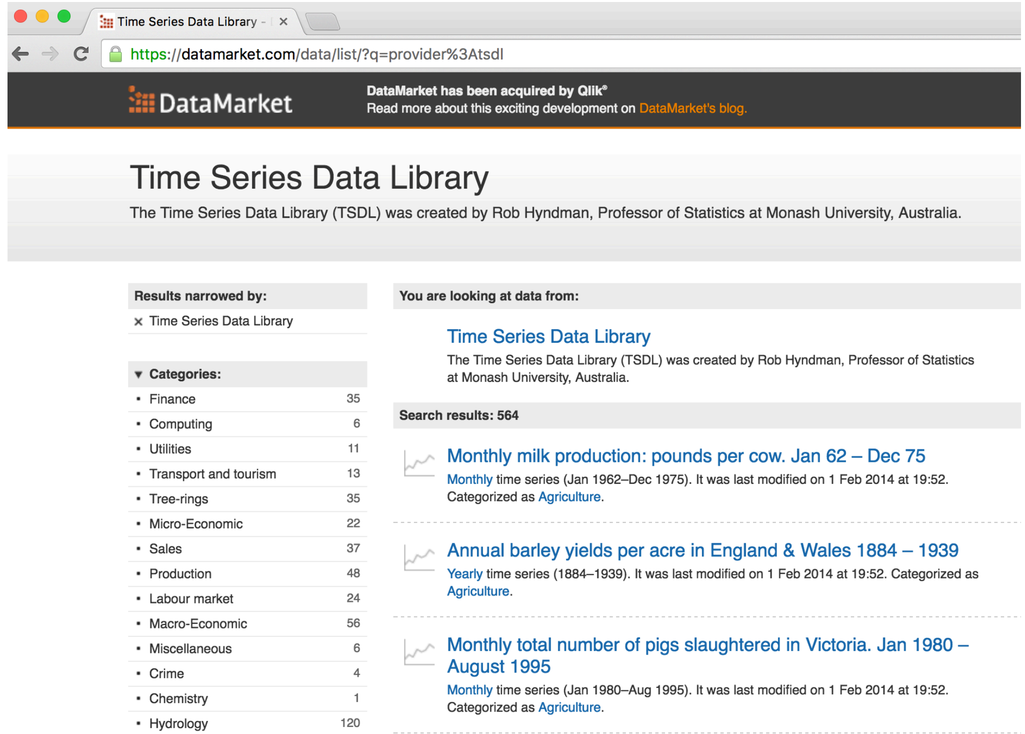
Learn more…

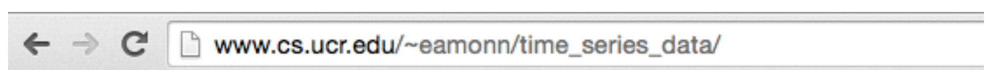
Hidden Markov Models are another way of thinking about time series classification.



https://en.wikipedia.org/wiki/Hidden_Markov_model

Get More Practice





UCR Time Series Classification Archive