Time Series in Python & R

Technical Review

Author: Laurent Querella, PhD, MSc

Senior Data Scientist

Version : draft version 1.1 – Apr 2022

Contents

1 General

1.1 References

**[Ben2014] S. Ben Taieb, *Machine learning strategies for multi-step-ahead time series forecasting*, PhD thesis, ULB, Brussels, 2014

Contains methodology for GEFComm2012

***[Kit2010] Kitagawa, Introduction to time series modeling, 2010

***[Sch2011] Schumway, Time Series Analysis and its Applications, 2011

*[Deg2017] De Gooijer, Elements of Nonlinear Time Series Analysis, 2017

**[BDD2002] Brockwell, PJ and RA Davis, *Introduction to time series and forecasting*, 2nd. Taylor & Francis, 2002.

1.2 Links

1.2.1 Online resources

***[Hyndman] online book Hyndman : https://otexts.org/fpp2/

***[PennState]: lecture syllabus: https://onlinecourses.science.psu.edu/stat510/?q=node/70

*[NIST] standards: http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc4.htm

1.2.2 Data sources

https://fred.stlouisfed.org/

http://ec.europa.eu/eurostat/data/database

1.3 Very first questions

- Find patterns and important features
  - Unsupervised learning / time series clustering
  - Control the quality of a process (e.g. upset detection, predictive maintenance)
    - Use case: FCC TRA Stability
  - Outliers?
- Explain or Predict?
  - Root cause analysis: how the past affects the future? How times series interact?
  - Forecast
    - Short-term or Long-term forecast?
      - Short – energy demand forecasting
      - Long – strategic planning (prices)
    - One-step or multi-step ahead?
      - Forecast horizon to agree upon beforehand
      - Multi-step forecast requires to adopt a strategy:
        - Recursive
        - Direct
- Is there a trend? Is there a seasonality?

    - TS decomposition?
        - E.g. STL-LOESS
    - Transformation?
        - E.g. Box-Cox
- Is there a long-run cycle (period unrelated to seasonality factors)?
- Stationarity:
    - Is there constant variance over time?
        - Differencing
    - Are there structural changes to either the level (mean) of the series or the variance?
        - Use case: Catalyst cycles
- Time granularity?
- Quantity of data available
- Univariate or Multivariate problem?
    - Does the target variable influence predictors as well?

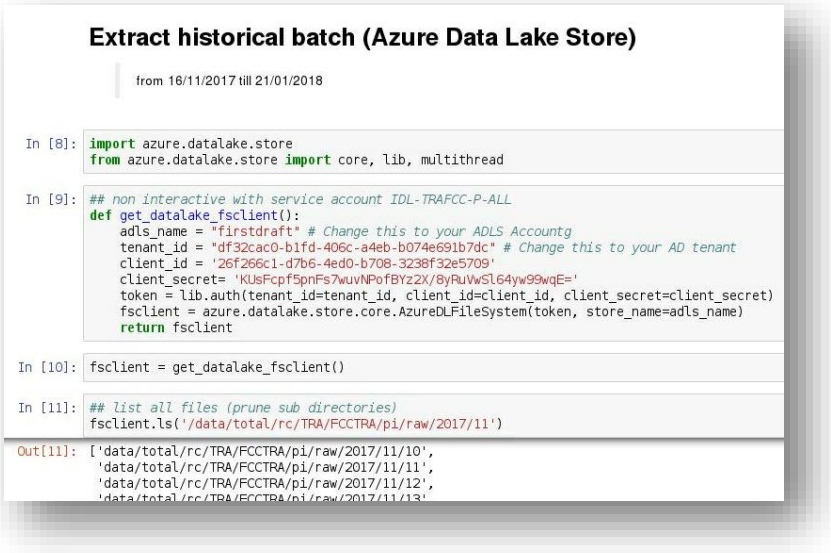
1.4 Recommended preliminary steps

1.4.1 Extraction

Data sources:

- Csv files (e.g. open data)
- Data Lake tables/files
- …

Example: (FCC-TRA)



Extract historical batch (Azure Data Lake Store)

from 16/11/2017 till 21/01/2018

```
In [8]: import azure.datalake.store
        from azure.datalake.store import core, lib, multithread

In [9]: ## non interactive with service account IDL-TRAFCC-P-ALL
        def get_datalake_fsclient():
            adls_name = "firstdraft" # Change this to your ADLS Accountg
            tenant_id = "df32cac0-b1fd-406c-a4eb-b074e691b7dc" # Change this to your AD tenant
            client_id = '26f266c1-d7b6-4ed0-b708-3238f32e5709'
            client_secret= 'KUsFcpf5pnFs7wuvNPofBYz2X/8yRuVwSl64yw99wqE='
            token = lib.auth(tenant_id=tenant_id, client_id=client_id, client_secret=client_secret)
            fsclient = azure.datalake.store.core.AzureDLFileSystem(token, store_name=adls_name)
            return fsclient

In [10]: fsclient = get_datalake_fsclient()

In [11]: ## list all files (prune sub directories)
         fsclient.ls('/data/total/rc/TRA/FCCTRA/pi/raw/2017/11')

Out[11]: ['data/total/rc/TRA/FCCTRA/pi/raw/2017/11/10',
          'data/total/rc/TRA/FCCTRA/pi/raw/2017/11/11',
          'data/total/rc/TRA/FCCTRA/pi/raw/2017/11/12',
          'data/total/rc/TRA/FCCTRA/pi/raw/2017/11/13'
```

1.4.2 Exploration and visualisation

1.4.2.1 Basic descriptive stats

```
tag_df[op_list].describe()
```

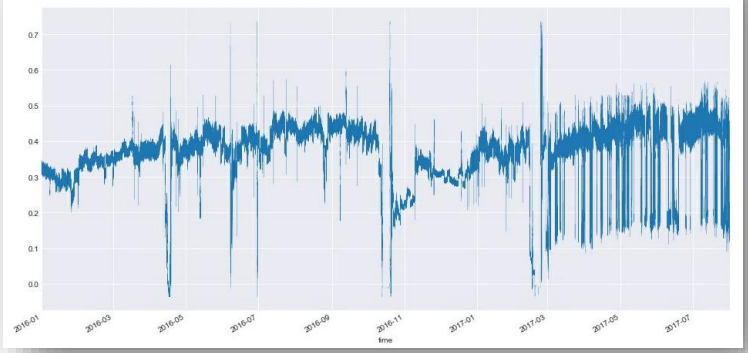
	FC66437_OP	FC670130_OP	FC670763_OP	FC671123_OP	FC670761_OP	FC670762_OP	FC670779_OP	FC670005_OP	FC6
count	832177.000000	644279.000000	286512.000000	164467.000000	503641.000000	463788.000000	801737.000000	801772.000000	175
mean	27.962471	32.486430	19.560070	48.181089	35.623782	36.930300	30.852917	69.839403	-4.6
std	13.591379	18.432604	26.923344	54.240788	29.450284	27.264057	12.473821	5.925569	6.7
min	-6.860000	-6.900000	-6.900000	-6.900000	-6.900000	-6.900000	-6.900000	-6.900000	-6.9
25%	20.885416	22.905075	-6.900000	-6.900000	7.099126	24.195628	21.183640	67.843930	-6.9
50%	32.249999	34.472440	16.976800	48.108640	37.816010	39.296550	36.277500	69.445270	-6.9
75%	34.620371	43.148310	35.715570	100.000000	61.189460	53.814907	38.883300	73.336265	-6.9
max	106.888885	106.900000	105.000000	106.900000	100.000000	105.000000	100.000000	100.000000	106.

1.4.2.2 Data visualisation

- Raw signals – time series plot

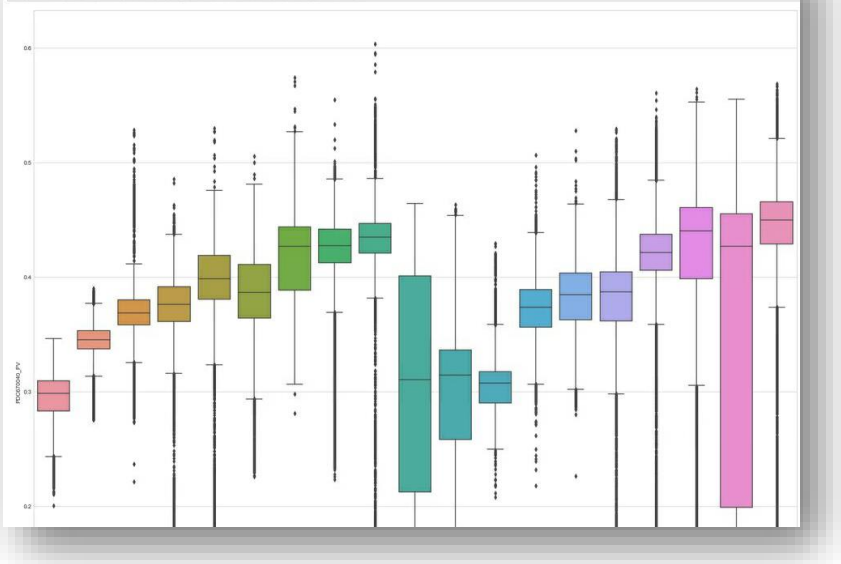
```
tag_df.PDC670040_PV.plot(figsize=(16, 8), lw=0.2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x9b7d438>
```

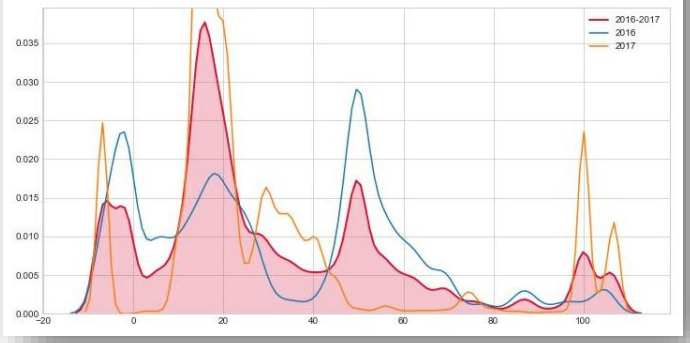


or other types of univariate plots:

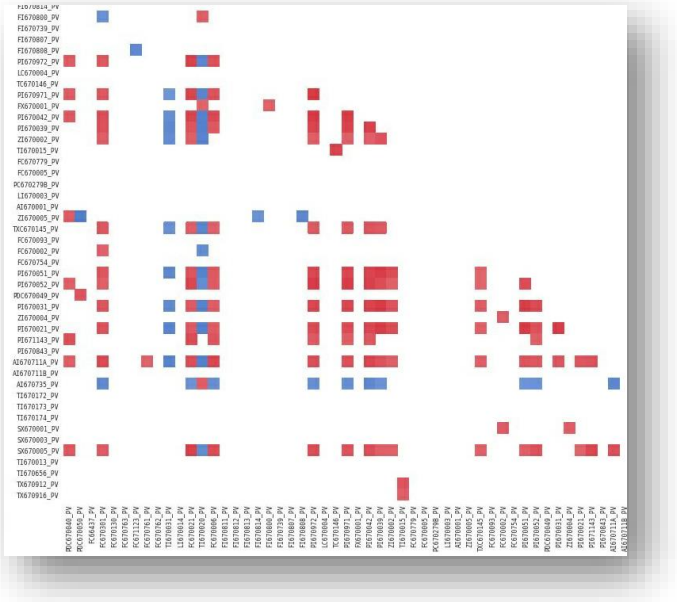
```
plt.rcParams['figure.figsize']=(24,24)
sb.set_style("whitegrid")
ax = sb.boxplot(x=test.yearmonth, y=test.PDC670040_PV)
```



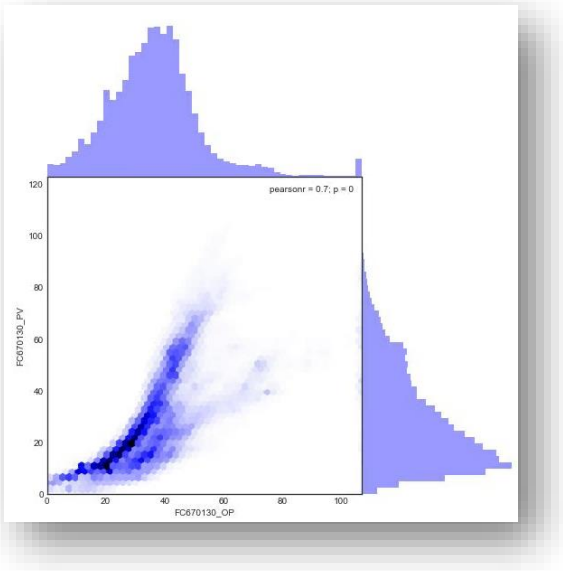
```
---------------------------------------------------------------
*** Tag:  FC670301_OP
          Sample 2016:  526980  Missing 2016:  195123   37.03 %
          Sample 2017:  305220  Missing 2017:  0   0.00 %
---------------------------------------------------------------
```



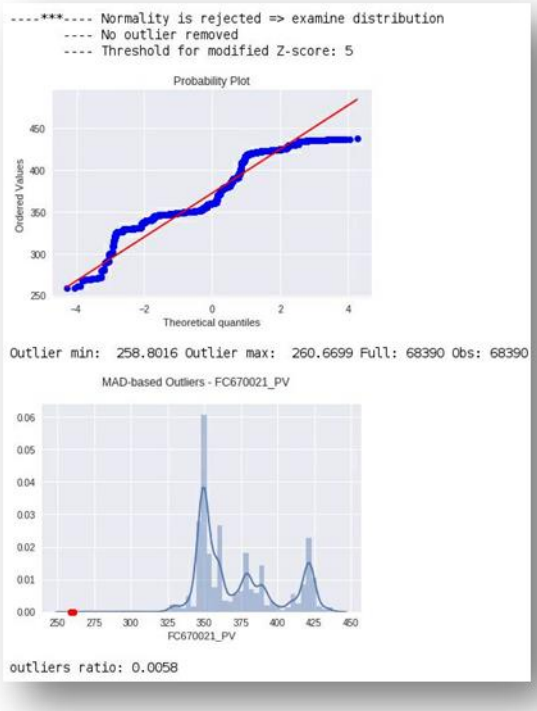
- Correlation matrix



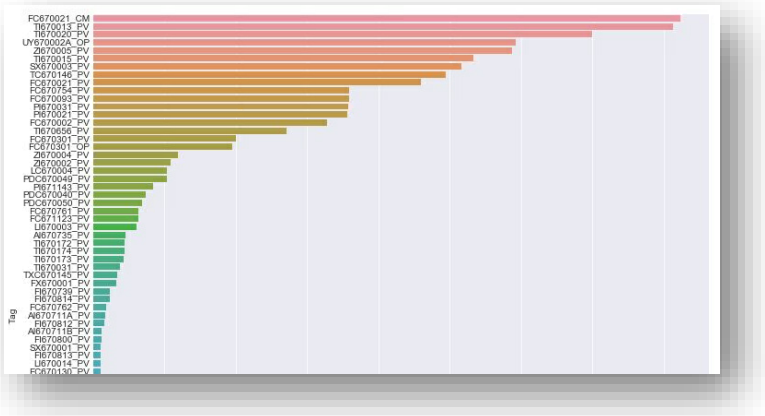
- Joint plot



1.4.2.3 Outlier detection

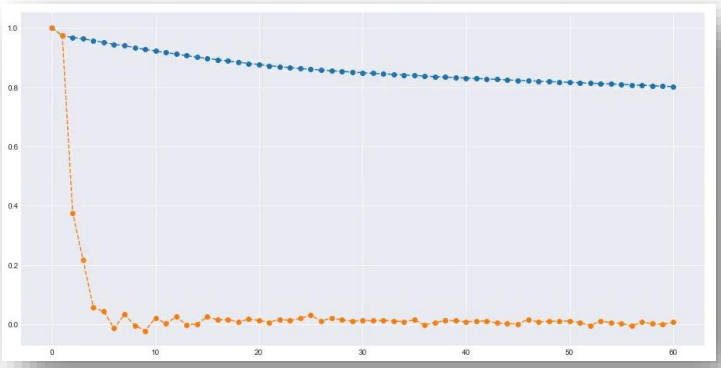


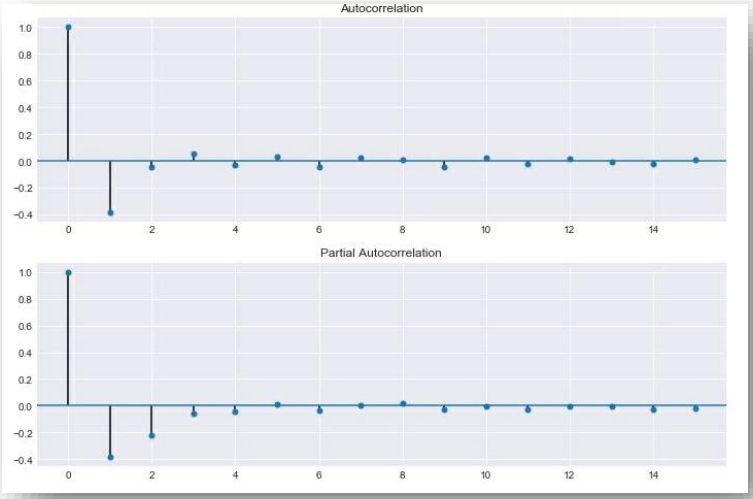
1.4.2.4 Missing values identification



1.4.2.5 Autocorrelation function

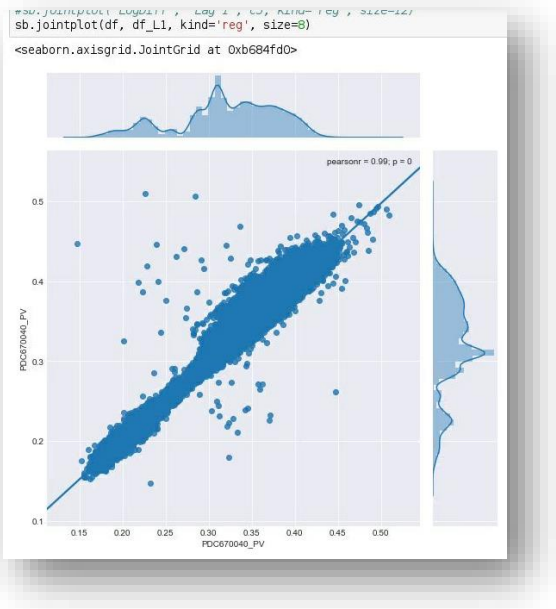
ACF/PCF for each series – CAUTION: must be weakly stationary



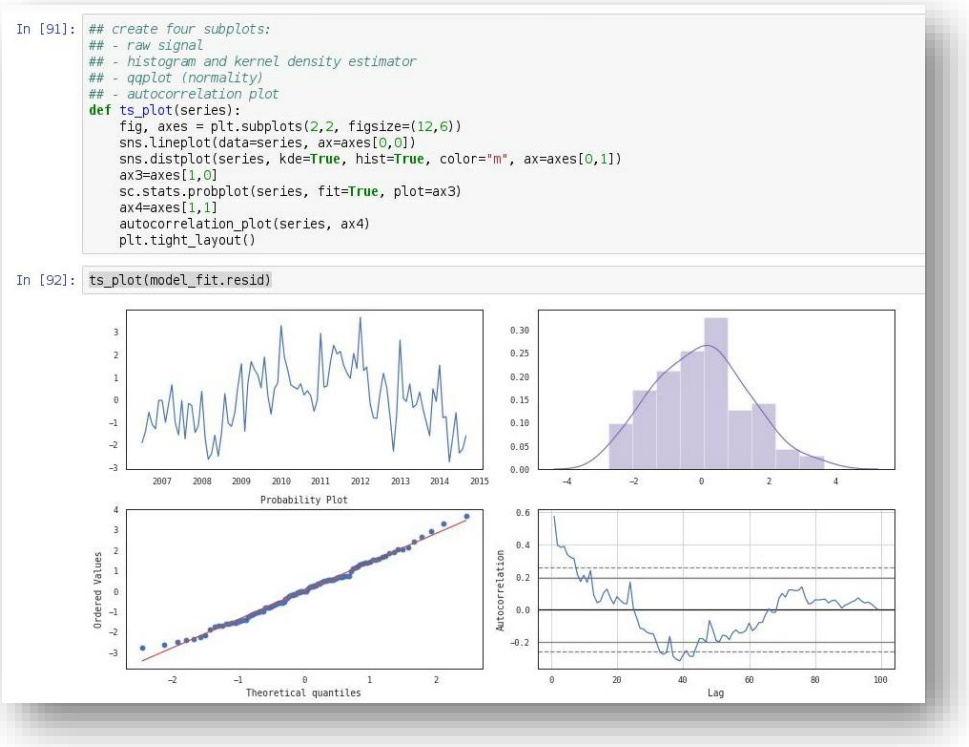


1.4.2.6 Possible clue on periodicity patterns

- Joint plot with lagged values



Looping on all ts can be done via function:



1.4.2.7 Cross-correlation

(…)

1.4.2.8 Power spectrum and Periodogram

[Kitagawa]

we can define the *Fourier transform* of C_k.

The function defined on the frequency $-1/2 \leq f \leq 1/2$,

$$p(f) = \sum_{k=-\infty}^{\infty} C_k e^{-2\pi i k f}, \tag{3.1}$$

is called the *power spectral density function* or simply the *power spectrum*.

- AR(1)

  $y_n = a y_{n-1} + w_n$

  - a>0

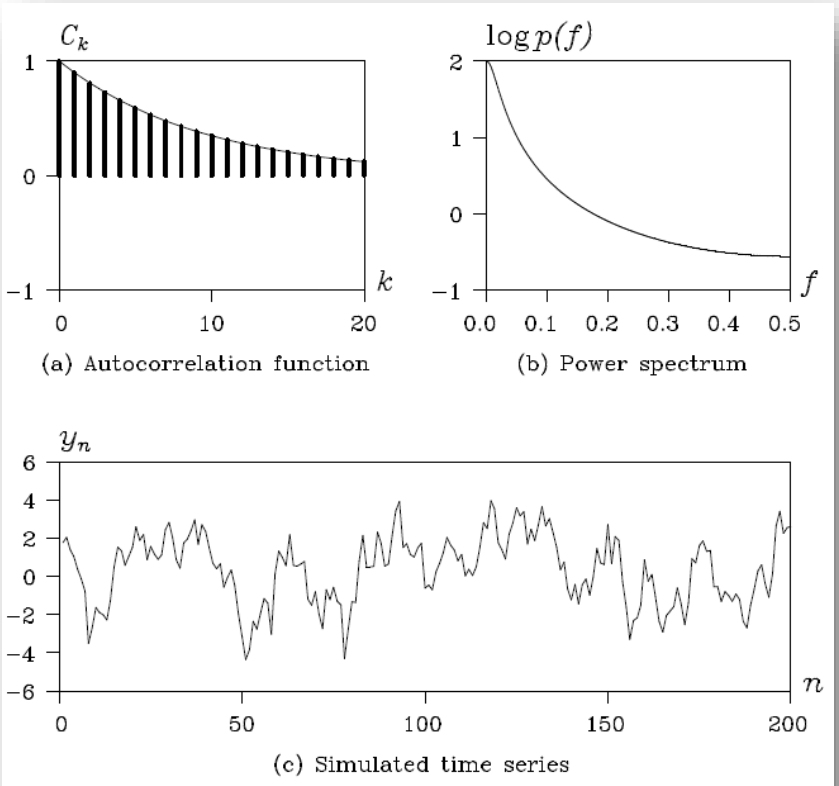


Figure 3.2 *Autocorrelation function, power spectrum, and realization of a first-order AR model with* $a = 0.9$.

- a<0

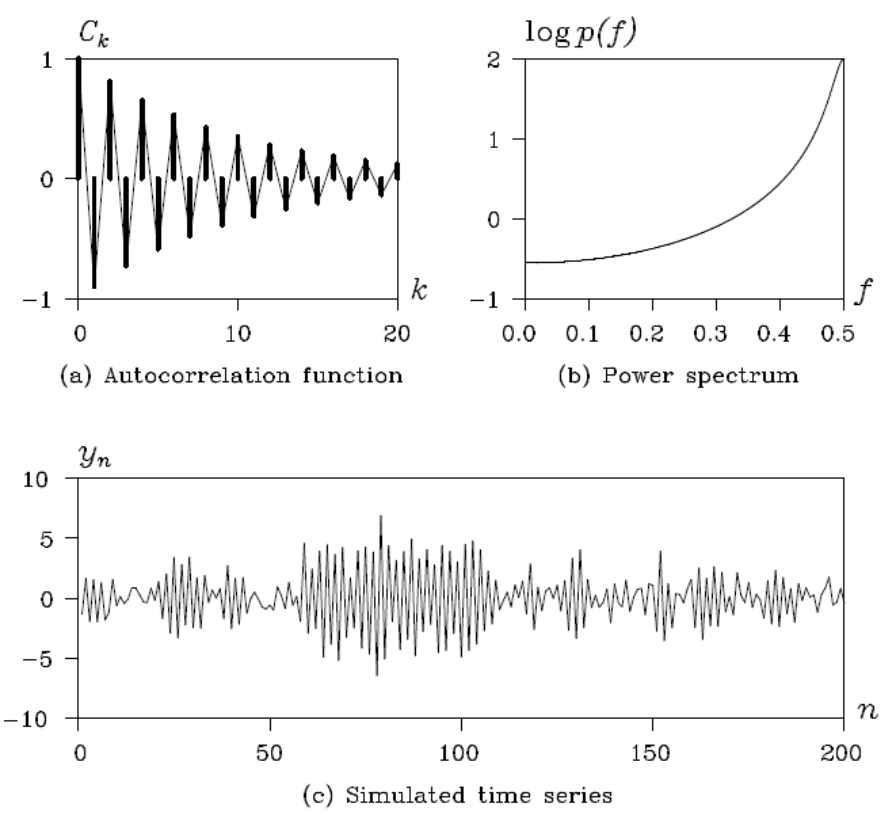


Figure 3.3 *Autocorrelation function, power spectrum, and realization of a first-order AR model with* $a = -0.9$.

- AR(2)

  $y_n = a_1 y_{n-1} + a_2 y_{n-2} + w_n$

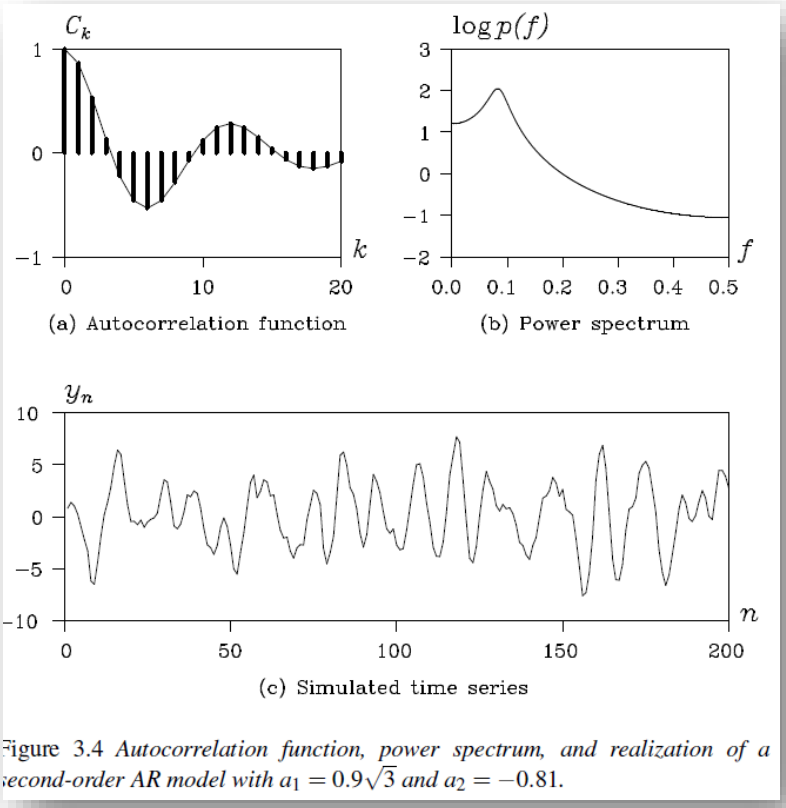


Figure 3.4 *Autocorrelation function, power spectrum, and realization of a second-order AR model with* $a_1 = 0.9\sqrt{3}$ *and* $a_2 = -0.81$.

Given a time series $y_1, \cdots, y_N$, the *periodogram* is defined by

$$p_j = \sum_{k=-N+1}^{N-1} \hat{C}_k e^{-2\pi i k f_j} = \hat{C}_0 + 2\sum_{k=1}^{N-1} \hat{C}_k \cos 2\pi k f_j, \qquad (3.8)$$

where the sample autocovariance function $\hat{C}_k$ is substituted for the autocovariance function C_k of equations (3.1) and (3.2).

1.4.2.9 Residuals

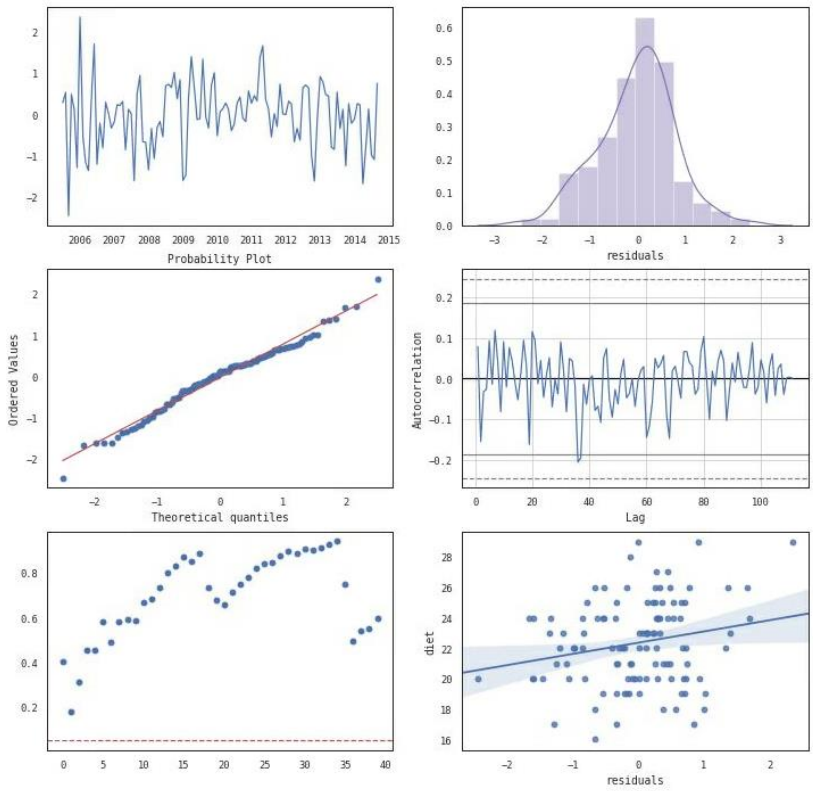
NB: Residuals usually are theoretically assumed to have an ACF that has correlation = 0 for all lags.

Several visualisations can be done to check the absence of correlations in residuals.

Cf. own function in python:

- Line plot of residuals (not standardized)
- Histogram and kde (normality)
- Qqplot (normality)
- ACF
- P-values of the Ljung–Box(-Pierce) statistic
    - We reject the null of no autocorrelation if the p-value is small.

- Joint plot (residuals, original time series with possible lags)



1.4.3 Preprocessing

1.4.3.1 Data sanitizing

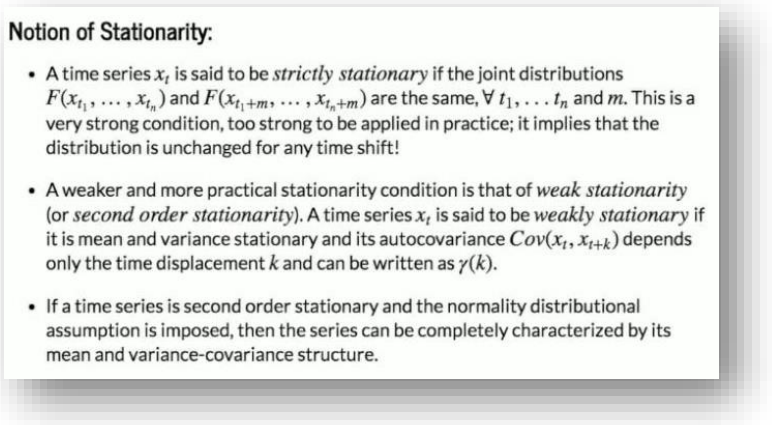
- Time granularity
  - E.g. convert date to DateTimeIndex
- Handling outliers
  - Cf. references in https://www.researchgate.net/profile/Guy_Melard/post/How_do_I_build_an_ARIMA_model_based_on_MM-Estimator/attachment/59d61f3c79197b807797dad1/AS:283716472590337@1444654686770/download/BSS2014_poster%26minipaper_v2.pdf
- Imputing missing values
  - Cf. references in https://www.researchgate.net/profile/Guy_Melard/post/How_do_I_build_an_ARIMA_model_based_on_MM-Estimator/attachment/59d61f3c79197b807797dad1/AS:283716472590337@1444654686770/download/BSS2014_poster%26minipaper_v2.pdf
- Resampling
  - Upsampling
    - Time series is resampled from low frequency to high frequency(Monthly to daily frequency). It involves filling or interpolating missing data
  - Downsampling

        - Time series is resampled from high frequency to low frequency(Weekly to monthly frequency). It involves aggregation of existing data.
- Normalising (or not)
    - Depending on the method
- Feature engineering
    - Rolling windows with aggregation
    - OHLC
        - open, high, low and close price of a certain time period
- Correlation
    - Autocorrelation
        - The autocorrelation function (ACF) measures how a series is correlated with itself at different lags.
    - Partial Autocorrelation
        - The partial autocorrelation function can be interpreted as a regression of the series against its past lags. The terms can be interpreted the same way as a standard linear regression, that is the contribution of a change in that particular lag while holding others constant.

1.4.3.2 Stationarity

1.4.3.2.1 Definition

Check https://otexts.org/fpp2/stationarity.html



Time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times.

1.4.3.2.2 Statistical tests (Unit root tests)

Statistical hypothesis tests of stationarity that are designed for determining whether differencing is required.

A number of unit root tests are available, which are based on different assumptions and may lead to conflicting answers.

1.4.3.2.2.1 Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test

http://www.statisticshowto.com/kpss-test/

In this test, the null hypothesis is that the data are stationary, and we look for evidence that the null hypothesis is false. Consequently, small p-values (e.g., less than 0.05) suggest that differencing is required.

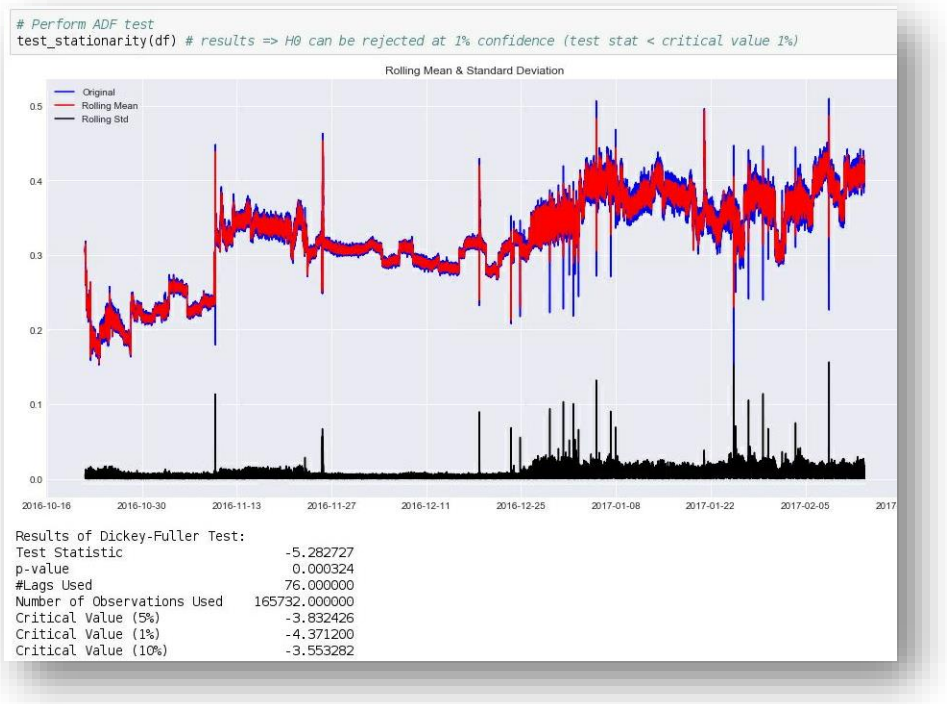
Python: https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.kpss.html

R: urca

1.4.3.2.2.2 Augmented Dickey-Fuller (ADF) test

http://www.statisticshowto.com/adf-augmented-dickey-fuller-test/

The more negative the DF test statistic, the stronger the evidence for rejecting the null hypothesis of a unit root.



1.4.3.2.3 Stationarity transformations

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

[Kitagawa]/[Hyndman]

Transformations such as logarithms can help to stabilise the variance of a time series.

Differencing can help stabilise the mean of a time series by removing changes in the level of a time series, and therefore eliminating (or reducing) trend and seasonality.

As well as looking at the time plot of the data, the ACF plot is also useful for identifying non-stationary time series. For a stationary time series, the ACF will drop to zero relatively quickly, while the ACF of non-stationary data decreases slowly.

1.4.3.2.3.1 Transformation of variables

Transformation

Applying first differencing or seasonal differencing to the log of the series should make the-above two series stationary, a condition that is required by the VAR model

Specifically, try the following transformation:

$$log(y_t) - log(y_{t-l}) = log(\frac{y_t}{y_{t-l}})$$

where l is some lag.

For the UMCSENT series, $l = 1$, whereas $l = 12$ for the beer series.

More generally: [Kitagawa]

formation (Box and Cox (1964)), which includes the log-transformation as a special case

$$z_n = \begin{cases} \lambda^{-1}(y_n^\lambda - 1), & \text{if } \lambda \neq 0 \\ \log y_n, & \text{if } \lambda = 0. \end{cases} \quad (4.52)$$

Ignoring a constant term, the Box-Cox transformation yields the logarithm of the original series for $\lambda = 0$, the inverse for $\lambda = -1$ and the square root for $\lambda = 0.5$. In addition, it agrees with the original data for $\lambda = 1$. Applying the information criterion AIC to the Box-Cox transformation, we can determine the best parameter λ of the Box-Cox transformation (Konishi and Kitagawa (2008)). On the assumption that the

To avoid unless the transformed variables have a clear interpretation (e.g. rates, …)

1.4.3.2.3.2 Differencing

When a time series y_n contains a trend as seen in Figures 1.1(c), (e) and (g), we might study the differenced series z_n defined by (Box and Jenkins (1970))

$$z_n = \Delta y_n = y_n - y_{n-1}. \quad (1.2)$$

This is motivated by the fact that, when y_n is a straight line expressed as $y_n = a + bn$, then the differenced series z_n becomes a constant as

$$z_n = \Delta y_n = b, \quad (1.3)$$

and the slope of the straight line can be removed.

Moreover, if y_n is a parabola and expressed by $y_n = a + bn + cn^2$, then the difference of z_n becomes a constant and a and b are removed as follows

$$\Delta z_n = z_n - z_{n-1}$$

Unit root tests are hypothesis tests of stationarity that can be used to determine whether differencing is required. Different unit root tests are based on different assumptions and have different null hypotheses. Two popular test are the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test.

1.4.3.3 Trend – Seasonality

1.4.3.3.1 ST Decomposition – Smoothing/Filtering – Local Regression

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

This is an important technique for all types of time series analysis, especially for seasonal adjustment.[2] It seek each of these has a certain characteristic or type of behaviour. For example, time series are usually decompos

- T_t, the trend component at time t, which reflects the long-term progression of the series (secular variation).
- C_t, the cyclical component at time t, which reflects repeated but non-periodic fluctuations. The duration of t
- S_t, the seasonal component at time t, reflecting seasonality (seasonal variation). A seasonal pattern exists week).[1]
- I_t, the irregular component (or "noise") at time t, which describes random, irregular influences. It represent

Hence a time series using an additive model can be thought of as

$$y_t = T_t + C_t + S_t + I_t,$$

whereas a multiplicative model would be

$$y_t = T_t \times C_t \times S_t \times I_t.$$

- Moving Average
- Exponential Smoothing
- Chow's Adaptive Control
- Winter's Linear and Seasonal Exponential Smoothing
- Hodrick-Prescott Filter technique
  - Caution: do not use the Hodrick-Prescott Filter
- STL – LOESS/LOWESS

- Only additive decomposition

Steps: also see [Kitagawa] Chap 12/13

- Forecast $A_t = T_t + eps_t$
- Forecast $S_t = y_t - A_t$
- $\hat{y}_t = \hat{A}_t + \hat{S}_t$

1.4.3.3.2 Trend

[Sch2011] Section 2.4

1.4.3.3.2.1 Moving average

(…)

1.4.3.3.2.2 Polynomial and periodic regression smoothers

(…)

1.4.3.3.2.3 Kernel smoothing

Usually with a Gaussian kernel – choice of bandwidth is key

1.4.3.3.2.4 Lowess and nearest neighbor regression

Different windows (how local it is) to estimate the trend and seasonal components resp.

1.4.3.3.2.5 Smoothing splines

(…)

1.4.3.3.2.6 Smoothing one series as a function of another

(…)

1.4.3.3.3 Seasonality

Identifying a Seasonal Model

- Step 1:
  Do a time series plot of the data.
  Examine it for features such as trend and seasonality.
- Step 2:
  Do any necessary differencing.
  The general guidelines are:
  - If there is seasonality and no trend, then take a difference of lag S. For instance, take a 12th difference for monthly data with seasonality. Seasonality will appear in the ACF by tapering slowly at multiples of S.
  - If there is linear trend and no obvious seasonality, then take a first difference. If there is a curved trend, consider a transformation of the data before differencing.
  - If there is both trend and seasonality, apply a seasonal difference to the data and then re-evaluate the trend. If a trend remains, then take first differences.
  - If there is neither obvious trend nor seasonality, don't take any differences.
- Step 3:
  Examine the ACF and PACF of the differenced data (if differencing is necessary).

We're using this information to determine possible models.
This can be tricky going involving some (educated) guessing. Some basic guidance:

- Non-seasonal terms:
  Examine the early lags (1, 2, 3, …) to judge non-seasonal terms.
  - Spikes in the ACF (at low lags) indicate non-seasonal MA terms.
  - Spikes in the PACF (at low lags) indicated possible non-seasonal AR terms.
- Seasonal terms:
  Examine the patterns across lags that are multiples of S.
  For example, for monthly data, look at lags 12, 24, 36, and so on.
  Judge the ACF and PACF at the seasonal lags in the same way you do for the earlier lags.

- Step 4:
  Estimate the model(s) that might be reasonable on the basis of Step 3.
  Don't forget to include any differencing that you did before looking at the ACF and PACF.
  In the software, specify the original series as the data and then indicate the desired differencing when specifying parameters in the arima command that you're using.
- Step 5:
  Examine the residuals (with ACF, Box-Pierce, and any other means) to see if the model seems good.
  Compare AIC or BIC values if you tried several models.
  If things don't look good here, it's back to Step 3 (or maybe even Step 2).

1.4.3.4 Feature engineering tricks

1.4.3.4.1 Natural features

Features can be numeric or Boolean (categorical).

- Lags of time series – up to a certain order
  - Order determined by arbitrary choice of cut-off or according to domain expertise (e.g. dynamic of a process – elapsed time)
- Window statistics
  - Max/Min/Mean/Median/Variance/Count values within a time window
- DateTime:
  - Day of week, hour of day, …
- Calendar
  - Holidays, special days (e.g. when a manufacturing unit is stopped for maintenance)
    - Indicator variables

1.4.3.4.2 Target encoding

https://medium.com/datadriveninvestor/improve-your-classification-models-using-mean-target-encoding-a3d573df31e8

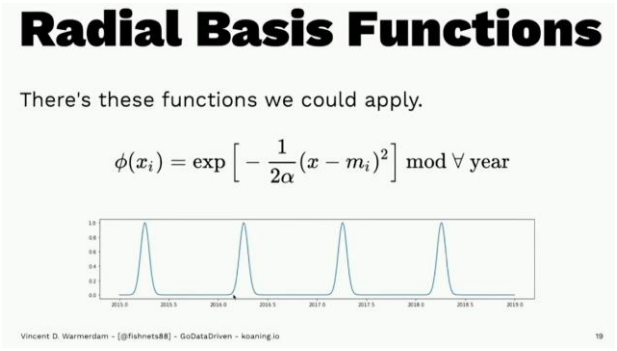
Must be used with regularization to avoid overfitting!

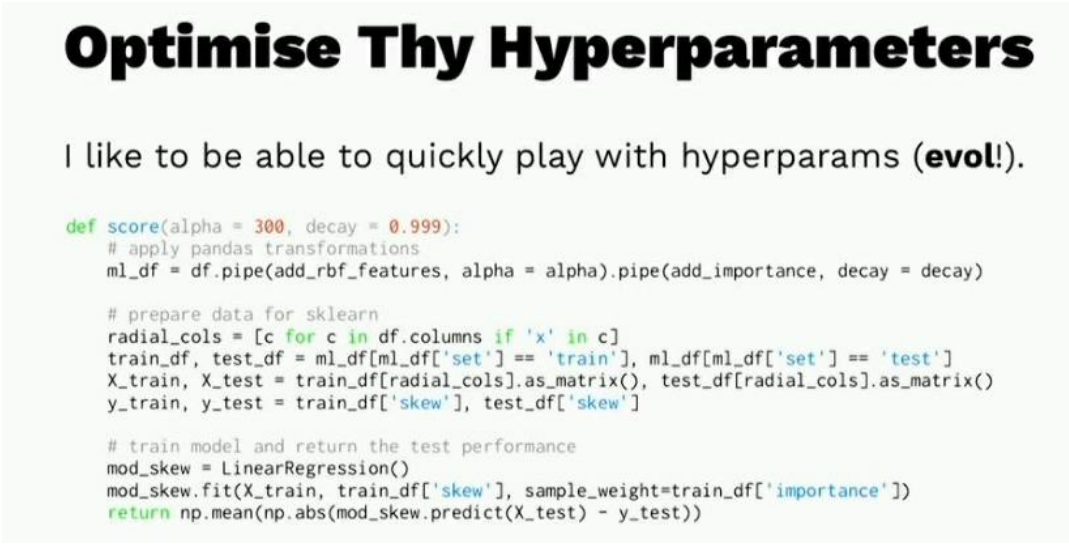
1.4.3.4.3 Fourier terms

Fourier decomposition

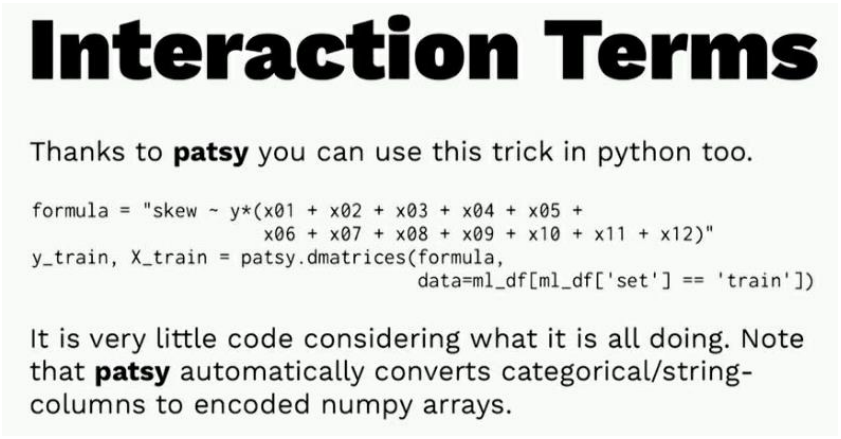
1.4.3.4.4 Radial Basis Functions

https://www.youtube.com/watch?v=68ABAU_V8qI





1.4.3.4.5 Interaction terms



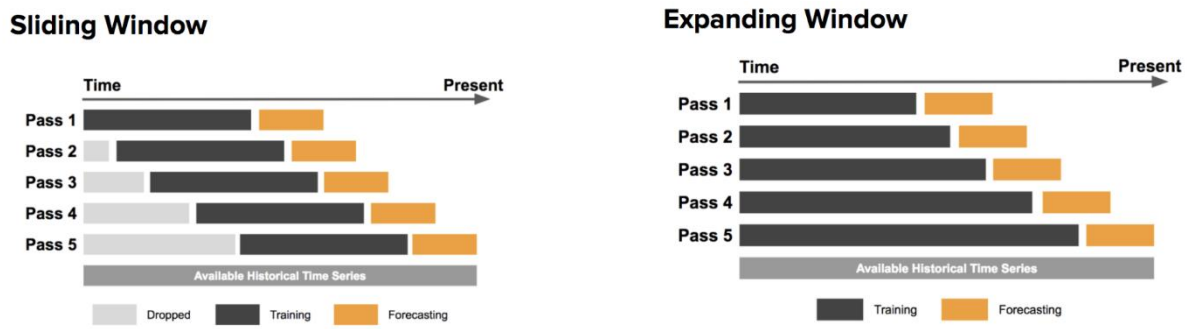
1.4.4 Train-Test split – Cross-Validation strategy

CAUTION! – information leakage

https://cran.r-project.org/web/packages/dataPreparation/vignettes/train_test_prep.html

cf. UBER:

To avoid introducing a bias in test using train-data, the train-test split should be performed before (most) data preparation steps.



There are two major approaches, outlined in Figure 4, above: the **sliding window** approach and the **expanding window** approach. In the sliding window approach, one uses a fixed size window, shown here in black, for training. Subsequently, the method is tested against the data shown in orange.

On the other hand, the expanding window approach uses more and more training data, while keeping the testing window size fixed. The latter approach is particularly useful if there is a limited amount of data to work with.

It is also possible, and often best, to marry the two methods: start with the expanding window method and, when the window grows sufficiently large, switch to the sliding window method.

Backtesting – Omphalos

https://eng.uber.com/omphalos/

Sliding window

The sliding window method requires three hyperparameters: training window size, forecasting window size (horizon), and sliding steps, detailed below:

- Training window size: the number of data points included in a training pass
- Forecasting window size: the number of data points to include in forecasting
- Sliding steps: the number of data points skipped from one pass to another

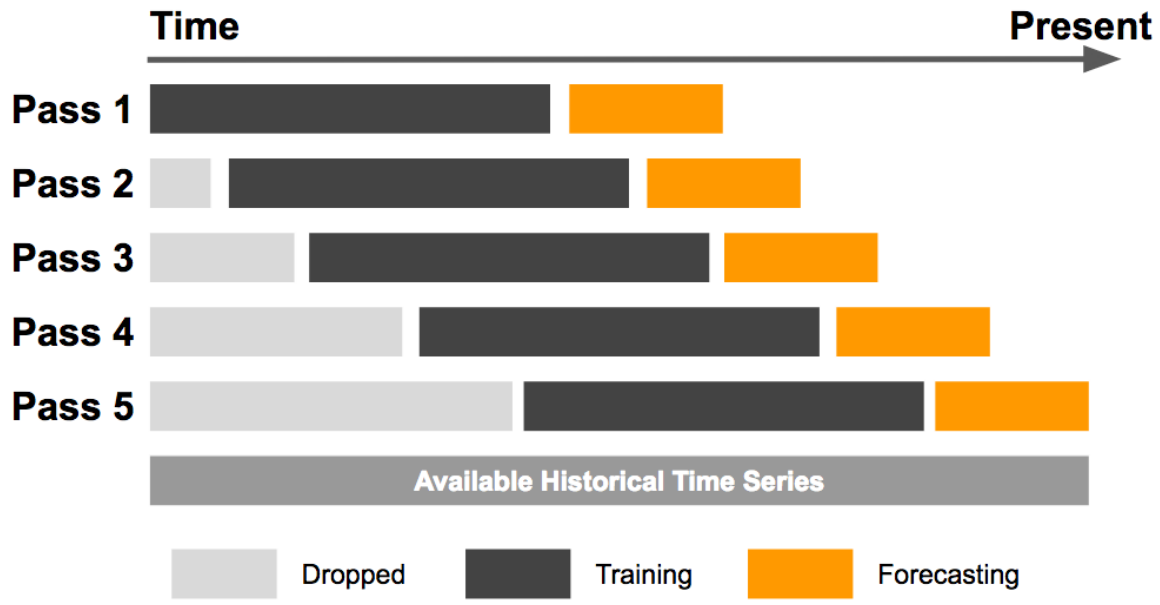


Figure 1: In the sliding window backtesting model, a fixed-size training window (in black) slides over the entire history of a time series and is repeatedly tested against a forecasting window (in orange) with older data points dropped.

Expanding window

The expanding window form requires four hyperparameters: starting window size, ending window size, forecasting window size, and expanding steps, outlined below:

- Starting window size: the number of data points included in the first training pass
- Ending window size: the number of data points included in the last training pass
- Forecasting window size: number of data points included for forecasting
- Expanding steps: the number of data points added to the training time series from one pass to another

Figure 2: In the expanding window backtesting model, a training window (in black) expands over the entire history of a time series and is repeatedly tested against forecasting window (in orange) without dropping older data points.

1.4.5 Evaluation metrics

Many evaluation metrics have been proposed in this space, including absolute errors and percentage errors, which have a few drawbacks. One particularly useful approach is to compare model performance against the **naive forecast**. In the case of a non-seasonal series, a naive forecast is when the last value is assumed to be equal to the next value. For a periodic time series, the forecast estimate is equal to the previous seasonal value (e.g., for an hourly time series with weekly periodicity the naive forecast assumes the next value is at the current hour one week ago).

To make choosing the right forecasting method easier for our teams, the Forecasting Platform team at Uber built a parallel, language-extensible backtesting framework called Omphalos to provide rapid iterations and comparisons of forecasting methodologies.

1.4.6 Uncertainty estimation

The importance of uncertainty estimation.

Determining the best forecasting method for a given use case is only one half of the equation. We also need to estimate prediction intervals. The prediction intervals are upper and lower forecast values that the actual value is expected to fall between with some (usually high) probability, e.g. 0.9. We highlight how prediction intervals work in Figure 5, below:

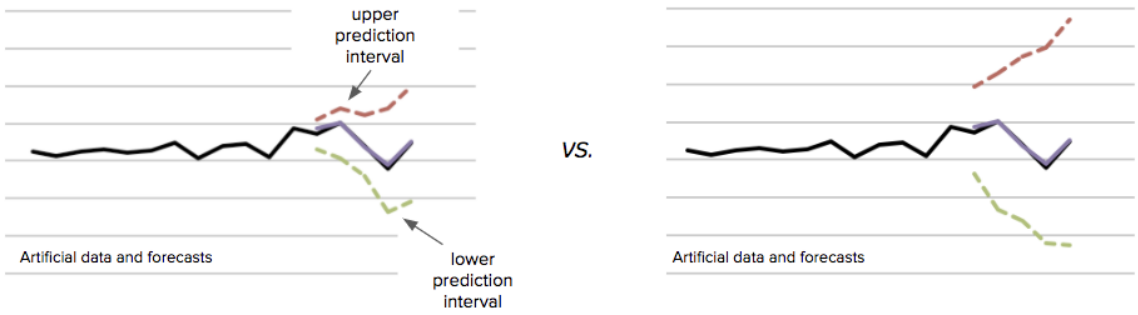


Figure 5: Prediction intervals are critical to informed decision making. Although point forecasts may be the same, their prediction intervals may be significantly different.

In Figure 5, the point forecasts shown in purple are exactly the same. However, the prediction intervals in the the left chart are considerably narrower than in the right chart. The difference in prediction intervals results in two very different forecasts, especially in the context of capacity planning: the second forecast calls for much higher capacity reserves to allow for the possibility of a large increase in demand.

Prediction intervals are just as important as the point forecast itself and should always be included in your forecasts. Prediction intervals are typically a function of how much data we have, how much variation is in this data, how far out we are forecasting, and which forecasting approach is used

1.5 Modeling

1.5.1 Basic models in time domain

There are two basic types of "time domain" models:

- Models that relate the present value of a series to past values and past prediction errors - these are called ARIMA models (for Autoregressive Integrated Moving Average).
- Ordinary regression models that use time indices as x-variables.

Key property to discriminate models: Is stationarity mandatory?

1.5.1.1 Statistical modeling

[Kitagawa]: K-L Information; Log-likelihood; MLE; AIC

1.5.1.2 Stationarity required

[Kitagawa] Chap 6-7-8

1.5.1.2.1 Autoregressive models

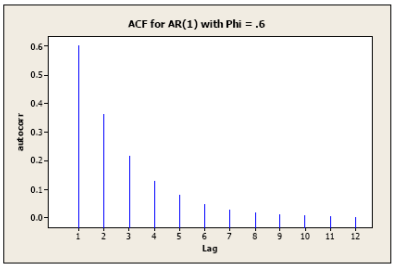
1.5.1.2.1.1 ARIMA (or Box-Jenkins models)

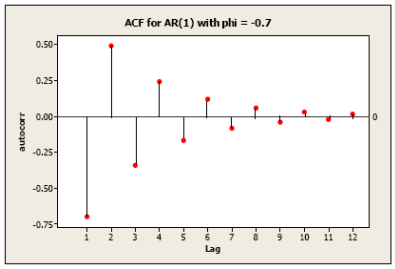
Univariate autoregressive models (AR), moving average (MA) and univariate autoregressive moving average models (ARMA) are special cases of ARIMA models.

ARIMA(p,d,q):

- p = AR order: lag dependency
  - Jointplot y_t y_t-1 or lagplot

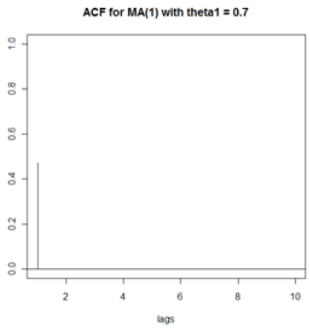
- Ar(1) with phi > 0 and phi <0 resp. : $x_t = \delta + \phi_1 x_{t-1} + w$

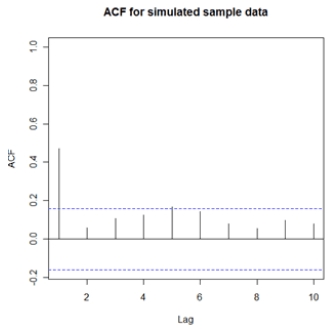




  - Identification of an AR model is often best done with the PACF because drops after the order p, i.e. the number of non-zero partial autocorrelations gives the order of the AR model.

- d = Differencing order: how many differencing operations are necessary to make the ts stationary
- q = MA order: error dependency in previous lags (past errors)
  - MA(1): $x_t = \mu + w_t + \theta_1 w_{t-1}$
    - Important: theta values should be < 1 (invertibility requirement)
    - ACF drops to zero as of lag 2 (theory – simulated data)





  - MA(q): ACF drops after lag q
    - Invertibility ➔ unit roots of the polynomial equation outside the unit circle
  - Identification of an MA model is often best done with the ACF rather than the PACF: the theoretical PACF does not shut off, but instead tapers toward 0 in some manner. A clearer pattern for an MA model is in the ACF. The ACF will have non-zero autocorrelations only at lags involved in the model.

[Hyndman]: process to handle ARIMA modeling

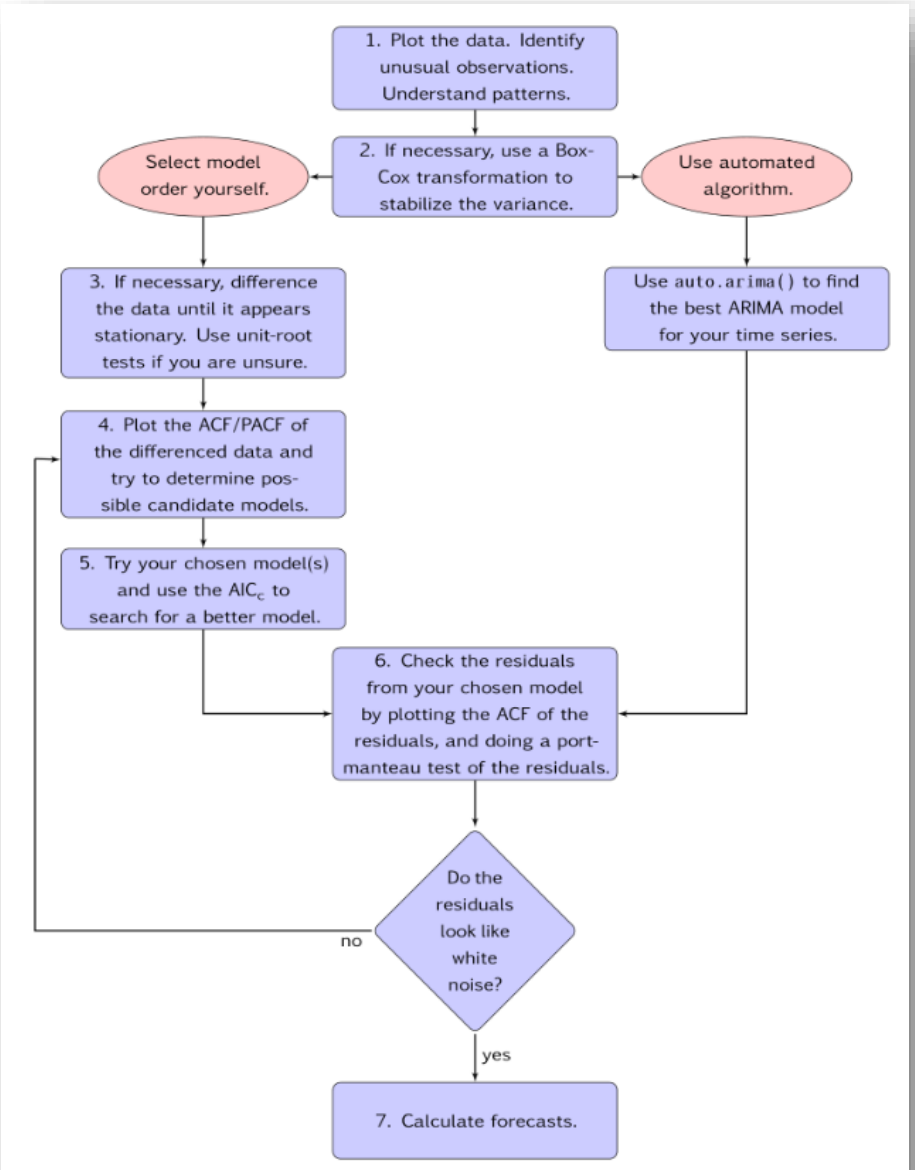


Figure 8.11: General process for forecasting using an ARIMA model.

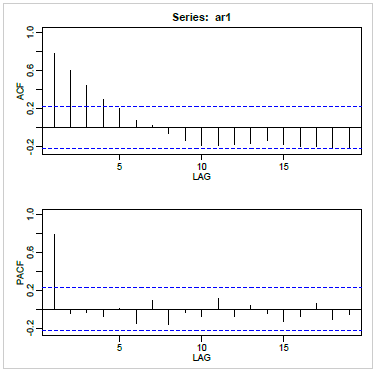
[Pennstate] complimentary information:

<u>Identifying a Possible Model</u>

1. Time series plot (raw signal)
    a. Obvious upward or downward linear trend
        i. ➔ First diff
    b. Quadratic trend
        i. ➔ Second order diff
    c. Higher-order trend
        i. ➔ other techniques like smoothing/splines/etc.
    d. Curved upward trend + increasing variance
        i. ➔ Transformation (cf. log, square root, Cox-Box)

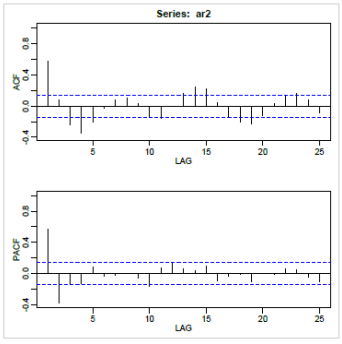
e. Nonconstant variance without trend
    i. ARCH model

2. ACF and PACF
    a. AR models have theoretical PACFs with non-zero values at the AR terms in the model and zero values elsewhere. The ACF will taper to zero in some fashion
        i. AR(1):



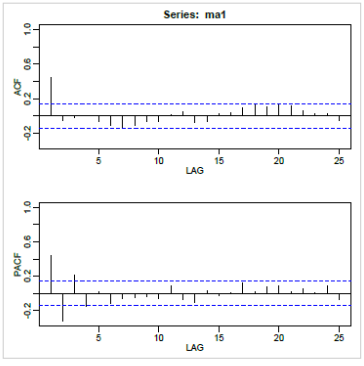
AR(1) Example Close or Esc Key

        ii. An AR(2) has a sinusoidal ACF that converges to 0



AR(2) Example Close or Esc Key

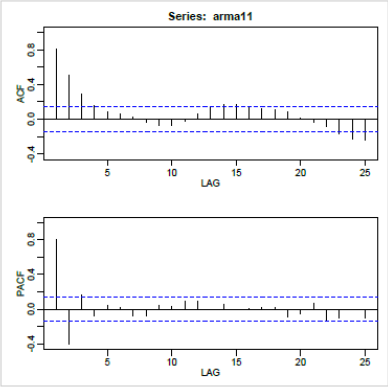
    b. MA models have theoretical ACFs with non-zero values at the MA terms in the model and zero values elsewhere
        i. MA(1)



MA(1) Example Close or Esc Key

    c. ARMA models (including both AR and MA terms) have ACFs and PACFs that both tail off to 0. These are the trickiest because the order will not be particularly obvious. Basically you just have to guess that one or two terms of each type may be needed and then see what happens when you estimate the model

i. ARMA(1,1):



d. If the ACF and PACF do not tail off, but instead have values that stay close to 1 over many lags, the series is non-stationary and differencing will be needed. Try a first difference and then look at the ACF and PACF of the differenced data.
e. If all autocorrelations are non-significant, then the series is random (white noise; the ordering matters, but the data are independent and identically distributed.) You're done at that point.
f. If you have taken first differences and all autocorrelations are non-significant, then the series is called a random walk and you are done. (A possible model for a random walk is $x_t = \delta + x_{t-1} + w_t$. The data are dependent and are not identically distributed; in fact both the mean and variance are increasing through time.)

3. (Optional) lagplots (x_t vs x_{t-h})

<u>Estimating and Diagnosing a Possible Model</u>

1. Estimate the coefficients (python/R). Common method is MLE (maximum likelihood estimation)
2. Look at the significance of the coefficients.
   a. P-values if provided by the algorithm implementation (compare the p-value to the standard 0.05 cut-off).
   b. Calculate a t-statistic: (…) When n is large, you may compare estimated coeff. / std. error of coeff to 1.96.
3. Look at the ACF of the residuals.
   a. For a good model, all autocorrelations for the residual series should be non-significant. If this isn't the case, you need to try a different model.
4. Look at Box-Pierce (Ljung) tests for possible residual autocorrelation at various lags
5. If non-constant variance is a concern, look at a plot of residuals versus fits and/or a time series plot of the residuals.

If something looks wrong, you'll have to revise your guess at what the model might be. This might involve adding parameters or re-interpreting the original ACF and PACF to possibly move in a different direction.

<u>What if More Than One Model Looks Okay?</u>

Sometimes more than one model can seem to work for the same dataset. When that's the case, some things you can do to decide between the models are:

1. Possibly choose the model with the fewest parameters.
2. Examine standard errors of forecast values.
    a. Pick the model with the generally lowest standard errors for predictions of the future.
3. Compare models with regard to statistics such as the MSE (the estimate of the variance of the wt), AIC, AICc, and SIC (also called BIC). Lower values of these statistics are desirable. The statistics combine the estimate of the variance with values of the sample size and number of parameters in the model.

One reason that two models may seem to give about the same results is that, with the certain coefficient values, two different models can sometimes be nearly equivalent when they are each converted to an infinite order MA model. [Every ARIMA model can be converted to an infinite order MA – this is useful for some theoretical work, including the determination of standard errors for forecast errors.]

1.5.1.2.1.2 SARIMA

Include a seasonal pattern

SARIMA: https://onlinecourses.science.psu.edu/stat510/node/67/

The seasonal ARIMA model incorporates both non-seasonal and seasonal factors in a multiplicative model:

ARIMA(p, d, q) × (P, D, Q)S

SARIMAX:

https://www.statsmodels.org/dev/examples/notebooks/generated/statespace_sarimax_stata.html

NB: to handle multiple seasonality, other techniques are required (msts in R)

https://otexts.org/fpp2/complexseasonality.html

example: using Dynamic harmonic regression with multiple seasonal periods by means of Fourier terms (use in Python scipy.fftpack).

1.5.1.2.1.3 AUTO (S)ARIMA

https://medium.com/@josemarcialportilla/using-python-and-auto-arima-to-forecast-seasonal-time-series-90877adff03c

https://www.analyticsvidhya.com/blog/2018/08/auto-arima-time-series-modeling-python-r/

https://www.alkaline-ml.com/pyramid/modules/classes.html

Automatic selection of parameters.

There are many ways to choose these values statistically, such as looking at auto-correlation plots, correlation plots, domain experience, etc.

One simple approach is to perform a grid search over multiple values of p, d, q, P, D, and Q using some sort of performance criteria. The Akaike information criterion (AIC) is an estimator of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. The AIC value will allow us to compare how well a model fits the data and takes into account the complexity of a model, so models that have a better fit while using fewer features will receive a better (lower) AIC score than similar models that utilize more features.

The pyramid-arima library for Python allows us to quickly perform this grid search and even creates a model object that you can fit to the training data. This library contains an auto_arima function that allows us to set a range of p, d, q, P, D, and Q values and then fit models for all the possible combinations. Then the model will keep the combination that reported back the best AIC value.

1.5.1.3 Exponential smoothing methods

Not based on an autoregression and produce forecasts by taking a weighted average of past observations with the weights decaying exponentially as the observations get older.

Smoothing: symmetric ➔ not appropriate for forecasts

Exp smoothing: asymmetric

[Kitagawa] Section 9.3

1.5.1.3.1 Basic exponential smoothing

$$\hat{y}_t = \alpha \cdot y_t + (1-\alpha) \cdot \hat{y}_{t-1}$$

Here the model value is a weighted average between the current true value and the previous model values. The α weight is called a smoothing factor. It defines how quickly we will "forget" the last available true observation. The smaller α is, the more influence the previous observations have and the smoother the series is.

Exponentiality is hidden in the recursiveness of the function -- we multiply by $(1-\alpha)$ each time, which already contains a multiplication by $(1-\alpha)$ of previous model values.

Smoothing parameter alpha should be tuned.

1.5.1.3.2 Double (or second-order) exponential smoothing

Basic exponential smoothing does not work well when there is a trend.

Series decomposition will help us -- we obtain two components: intercept (i.e. level) ℓ and slope (i.e. trend) b. We have learnt to predict intercept (or expected series value) with our previous methods; now, we will apply the same exponential smoothing to the trend by assuming that the future direction of the time series changes depends on the previous weighted changes. As a result, we get the following set of functions:

$$\ell_x = \alpha y_x + (1-\alpha)(\ell_{x-1} + b_{x-1})$$

$$b_x = \beta(\ell_x - \ell_{x-1}) + (1-\beta) b_{x-1}$$

$$\hat{y}_{x+1} = \ell_x + b_x$$

The first one describes the intercept, which, as before, depends on the current value of the series. The second term is now split into previous values of the level and of the trend. The second function describes the trend, which depends on the level changes at the current step and on the previous value of the trend. In this case, the β coefficient is a weight for exponential smoothing. The final prediction is the sum of the model values of the intercept and trend.

Alpha and beta should be tuned.

1.5.1.3.3 Triple exponential smoothing a.k.a. Holt-Winters

Adding seasonality!

As you could have guessed, the idea is to add a third component - seasonality. This means that we should not use this method if our time series is not expected to have seasonality. Seasonal components in the model will explain repeated variations around intercept and trend, and it will be specified by the length of the season, in other words by the period after which the variations repeat. For each observation in the season, there is a separate component; for example, if the length of the season is 7 days (a weekly seasonality), we will have 7 seasonal components, one for each day of the week.

With this, let's write out a new system of equations:

$$\ell_x = \alpha(y_x - s_{x-L}) + (1-\alpha)(\ell_{x-1} + b_{x-1})$$

$$b_x = \beta(\ell_x - \ell_{x-1}) + (1-\beta)b_{x-1}$$

$$s_x = \gamma(y_x - \ell_x) + (1-\gamma)s_{x-L}$$

$$\hat{y}_{x+m} = \ell_x + mb_x + s_{x-L+1+(m-1)modL}$$

The intercept now depends on the current value of the series minus any corresponding seasonal component. Trend remains unchanged, and the seasonal component depends on the current value of the series minus the intercept and on the previous value of the component. Take into account that the component is smoothed through all the available seasons; for example, if we have a Monday component, then it will only be averaged with other Mondays. You can read more on how averaging works and how the initial approximation of the trend and seasonal components is done here. Now that we have the seasonal component, we can predict not just one or two steps ahead but an arbitrary m future steps ahead, which is very encouraging.

Below is the code for a triple exponential smoothing model, which is also known by the last names of its creators, Charles Holt and his student Peter Winters. Additionally, the Brutlag method was included in the model to produce confidence intervals:

$$\hat{y}_{max_x} = \ell_{x-1} + b_{x-1} + s_{x-T} + m \cdot d_{t-T}$$

$$\hat{y}_{min_x} = \ell_{x-1} + b_{x-1} + s_{x-T} - m \cdot d_{t-T}$$

$$d_t = \gamma \mid y_t - \hat{y}_t \mid + (1-\gamma)d_{t-T},$$

where T is the length of the season, d is the predicted deviation. Other parameters were taken from triple exponential smoothing. You can read more about the method and its applicability to anomaly detection in time series here.

1.5.1.4 State-Space model

[Kitagawa] chap 9

[Sch2011] chap 6

9.1 The State-Space Model

It is assumed that y_n is an ℓ-variate time series. The following model for the time series is called a *state-space model*.

$$x_n = F_n x_{n-1} + G_n v_n, \quad \text{(system model)} \quad (9.1)$$
$$y_n = H_n x_n + w_n, \quad \text{(observation model)}, \quad (9.2)$$

where x_n is a k-dimensional unobservable vector, referred to as the *state* (Anderson and Moore (1979)). v_n is a system noise or a state noise, that is, an m-dimensional white noise with mean vector zero and variance-covariance matrix Q_n. On the other hand, w_n is called observation noise; it is assumed to be an ℓ-dimensional Gaussian white noise with mean vector zero and the variance-covariance matrix R_n. F_n, G_n and H_n are $k \times k$, $k \times m$ and $\ell \times k$ matrices, respectively. Many linear models used in time series analysis are expressible in terms of state-space models.

https://www.statsmodels.org/dev/examples/notebooks/generated/statespace_sarimax_stata.html

`statsmodels.tsa.statespace` contains classes and functions that are useful for time series analysis using state space methods.

A general state space model is of the form

$$y_t = Z_t \alpha_t + d_t + \varepsilon_t$$
$$\alpha_t = T_t \alpha_{t-1} + c_t + R_t \eta_t$$

where y_t refers to the observation vector at time t, α_t refers to the (unobserved) state vector at time t, and where the irregular components are defined as

$$\varepsilon_t \sim N(0, H_t)$$
$$\eta_t \sim N(0, Q_t)$$

The remaining variables $(Z_t, d_t, H_t, T_t, c_t, R_t, Q_t)$ in the equations are matrices describing the process. Their variable names and dimensions are as follows

Z : *design* $(k\_endog \times k\_states \times nobs)$

d : *obs_intercept* $(k\_endog \times nobs)$

H : *obs_cov* $(k\_endog \times k\_endog \times nobs)$

T : *transition* $(k\_states \times k\_states \times nobs)$

c : *state_intercept* $(k\_states \times nobs)$

R : *selection* $(k\_states \times k\_posdef \times nobs)$

Q : *state_cov* $(k\_posdef \times k\_posdef \times nobs)$

In the case that one of the matrices is time-invariant (so that, for example, $Z_t = Z_{t+1} \; \forall \; t$), its last dimension may be of size 1 rather than size *nobs*.

This generic form encapsulates many of the most popular linear time series models (see below) and is very flexible, allowing estimation with missing observations, forecasting, impulse response functions, and much more.

1.5.1.4.1 SARIMAX

The SARIMAX class is an example of a fully-fledged model created using the state space backend for estimation.

1.5.1.4.2 VARMAX

(…)

1.5.1.4.3 Kalman filter

[Kitagawa] Section 9.2

A very computationally efficient procedure for obtaining the joint conditional distribution of the state has been developed by means of a recursive computational algorithm. This algorithm is known as the Kalman filter (Kalman (1960), Anderson and Moore (1976)).

1.5.2 Advanced models

1.5.2.1 Dynamic regression models

Also with lagged predictors

[Hyndman]

https://onlinecourses.science.psu.edu/stat510/node/72/

1.5.2.2 Lagged regression & Transfer function modeling

[Sch2011] Section 5.7

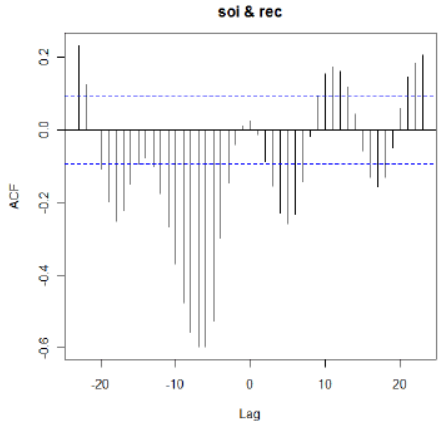
https://onlinecourses.science.psu.edu/stat510/node/74/

1.5.2.3 Cross-correlation function

CCF to identify the dependency in lags:

The goal may be to identify which variable is leading and which is lagging. In many problems we consider, though, we'll examine the x-variable(s) to be a leading variable of the y-variable because we will want to use values of the x-variable to predict future values of y. Thus, we'll usually be looking at what's happening at the negative values of h on the CCF plot.

Example (https://onlinecourses.science.psu.edu/stat510/node/74/): SOI could be a predictor for rec because leads rec (correlation in negative lags)



Transfer Function:

Y and x lags are considered and ts structure of predictors is also modeled.

1.5.2.4 Pre-whitening

https://onlinecourses.science.psu.edu/stat510/node/75/

http://hosting.astro.cornell.edu/~cordes/A6523/Prewhitening.pdf

One difficulty is that the CCF is affected by the time series structure of the x-variable and any "in common" trends the x and y series may have over time.

One strategy for dealing with this difficulty is called "pre-whitening." The steps are:

- Determine a time series model for the x-variable and store the residuals from this model
- Filter the y-variable series using the x-variable model (using the estimated coefficients from step 1).
    - In this step we find differences between observed y-values and "estimated" y-values based on the x-variable model
- Examine the CCF between the residuals from Step 1 and the filtered y-values from Step 2.
    - This CCF can be used to identify the possible terms for a lagged regression.

This strategy stems from the fact that when the input series (say, wt) is "white noise" the patterns of the CCF between wt and zt, a linear combination of lags of the wt, are easily identifiable (and easily derived). Step 1 above creates a "white noise" series as the input. Conceptually, Step 2 above arises from a starting point that y-series = linear combination of x-series. If we "transform" the x-series to white noise (residuals from its ARIMA model) then we should apply the transformation to both sides of the equation to preserve an equality of sorts.

It's not absolutely crucial that we find the model for x exactly. We just want to get close to the "white noise" input situation. For simplicity, some analysts only consider AR models (with differencing possible) for x because it's much easier to filter the y-variable with AR coefficients than with MA coefficients.

Pre-whitening is just used to help us identify which lags of x may predict y. After identifying possible model from the CCF, we work with the original variables to estimate the lagged regression.

Alternative strategies to pre-whitening include:

- Looking at the CCF for the original variables – this sometimes works
- De-trending the series using either first differences or linear regressions with time as a predictor

NB: This approach is valid if you want to pick the lags of x / this is tedious though when the number of variables is high; in that case, I prefer Regularized regression/boosted trees

1.5.2.5 ARCH/GARCH models (variance)

https://onlinecourses.science.psu.edu/stat510/node/85/

if y_t appears to be white noise and y_{2t}yt2 appears to be AR(1), then an ARCH(1) model for the variance is suggested. If the PACF of the y_{2t}yt2 suggests AR(m), then ARCH(m) may work.

1.5.2.6 Nonlinear models

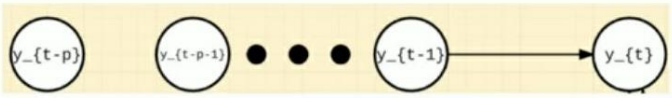
Non-linear models include Markov switching dynamic regression and autoregression.

[DeG2017]

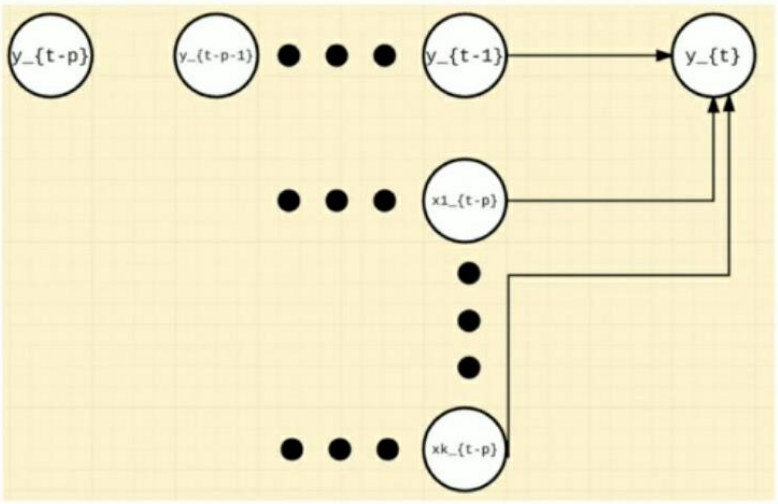
1.5.3 Univariate vs Multivariate

1.5.3.1 Univariate

In univariate time series forecasting, the statistical relationship is **unidirectional** in that the forecast variable is influenced by its own lags or the lags of other the predictor variables (or features)



+ exogeneous variables:



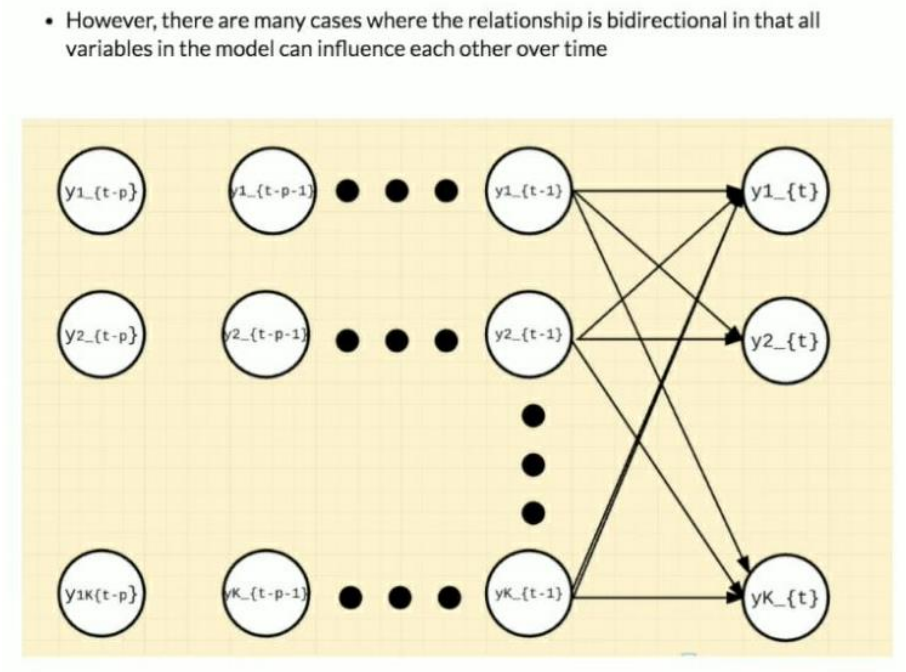
One such example is the family of ARMAX models:

$$y_t = a + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \omega_t + \theta_1 \omega_{t-1} + \cdots + \theta_q \omega_{t-q} + \sum_{m=1}^{M} \beta_m \mathbf{X}_{m,t}$$

where

$\omega_t \sim N(0, \sigma_\omega^2) \quad \forall t$

1.5.3.2 Multivariate



ARMAX models [Sch2011] Section 5.8

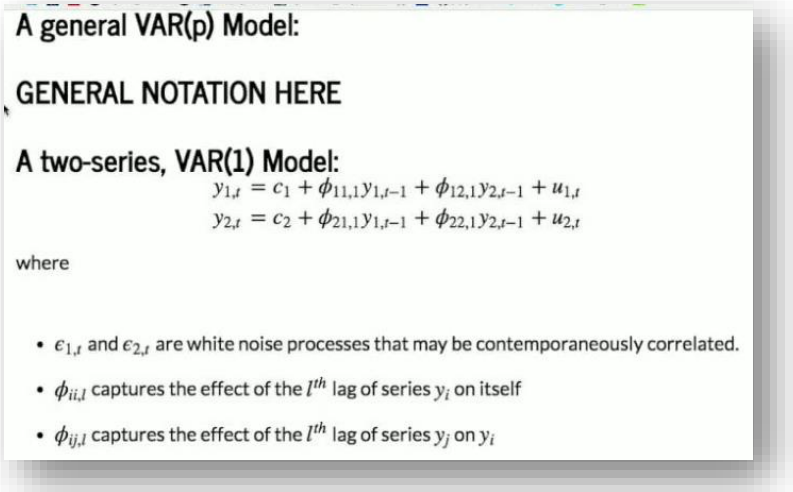
1.5.3.2.1 Example of model – VAR(p) models

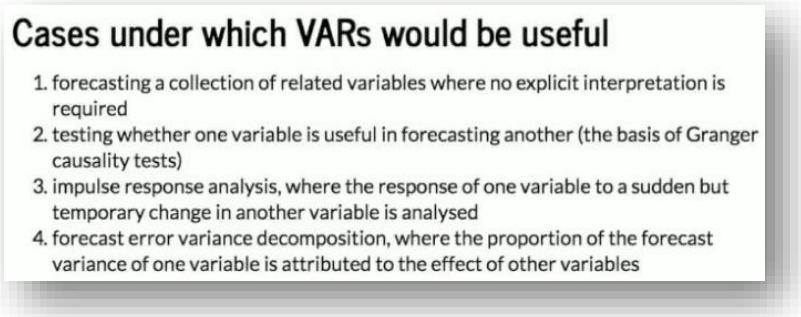
https://onlinecourses.science.psu.edu/stat510/node/79/

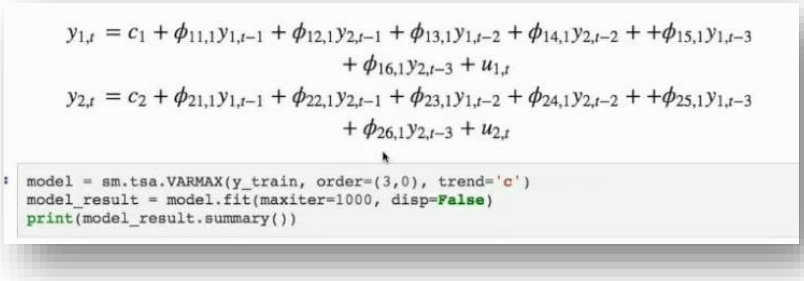
vector autoregressive models (VAR) or VARMA if MA is incorporated

These types of models require the series to be stationary! ➔ transform

If co-integration ➔ use other types of model
https://en.wikipedia.org/wiki/Cointegration







1.5.3.2.2 Model selection

Why picking order 3 in VAR model? ➔ select among different options using an **information criterion**

Model Selection (or Order Selection in the case of VAR(p))

```
aic = []
for i in range(5):
    i += 1
    model = sm.tsa.VARMAX(y_train, order=(i,0), trend='c')
    model_result = model.fit(maxiter=1000, disp=False)
    print('Order =', i)
    print('AIC: ', model_result.aic)
    print('BIC: ', model_result.bic)
    print('HQIC: ', model_result.hqic)
#     info_criteria = pd.concat([model_result.aic, model_result.bic],axis=1)
#     info_criteria = pd.concat([info_criteria   , model_result.hqic],axis=1)
#     #info_criteria.append(model_result.aic)
#     info_criteria.append(info_criteria)
```

```
Order = 1
AIC:  -2139.30980052
BIC:  -2106.15758262
HQIC:  -2126.03333758
```

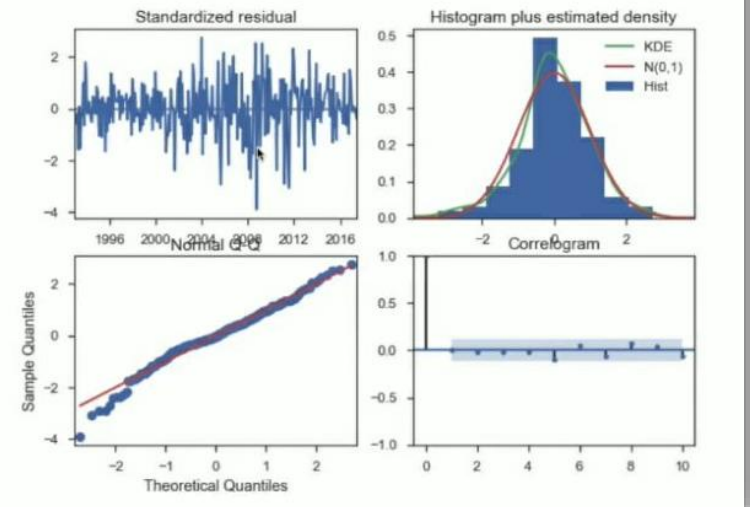
the lag order d is often selected by model-dependent penalties such as AIC or BIC

Alternative to model selection is data-driven validation (e.g. CV)

1.5.3.2.3 Model diagnosis

Residual plot ➔ white noise (stationatity/homeodasticity/…)

```
model = sm.tsa.VARMAX(y_train, order=(3,0), trend='c')
model_result = model.fit(maxiter=1000, disp=False)
model_result.plot_diagnostics()
```



1.5.3.2.4 Prediction (forecast)

Reverting first to original time series if transformations have been applied

Computing a metric (RMSE/MAPE/…)

1.5.4 Imputing missing values

[Kitagawa] Section 9.7

Cf. TRCRPM: A nonparametric Bayesian method for clustering, imputation, and forecasting in multivariate time series data.

https://github.com/probcomp/trcrpm

1.5.5 Clustering

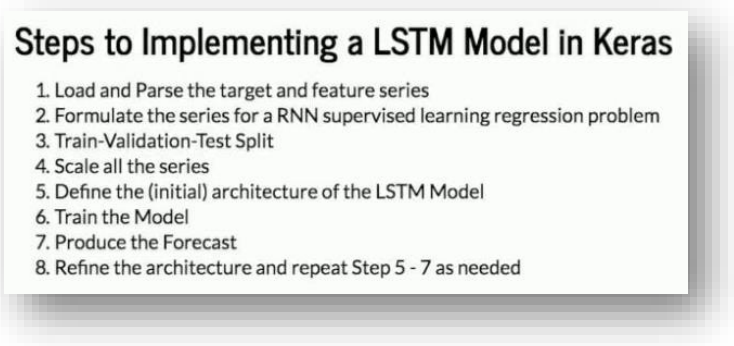
https://www.evernote.com/shard/s208/res/da5c50f1-91f3-4599-bbbd-43ea3402185d/stageverslag-roelofsen_tcm235-882304.pdf

Review of state-of-the-art methods:

https://arxiv.org/pdf/1808.10594.pdf

with spectacular improvement on scalability.

1.5.6 Neural Networks



1.5.7 Gaussian Processes

https://www.researchgate.net/post/How_to_use_Gaussian_processes_for_time_series_prediction

http://mc-stan.org/users/documentation/

GPR (the full GPR) has the computational complexity of O(N^3) for model training. If you training sample reaches few thousands. The inference can be way slow.

1.5.8 Frequency domain

1.5.8.1 Discrete Fourier Transform

(…)

1.6 Singular Spectrum Analysis (SSA)

https://www.kaggle.com/jdarcy/introducing-ssa-for-time-series-decomposition

Python package = PySSA: https://github.com/aj-cloete/pySSA

https://github.com/dmarienko/chaos

R:

http://karthur.org/2017/learning-for-time-series-ssa-vs-pca.html

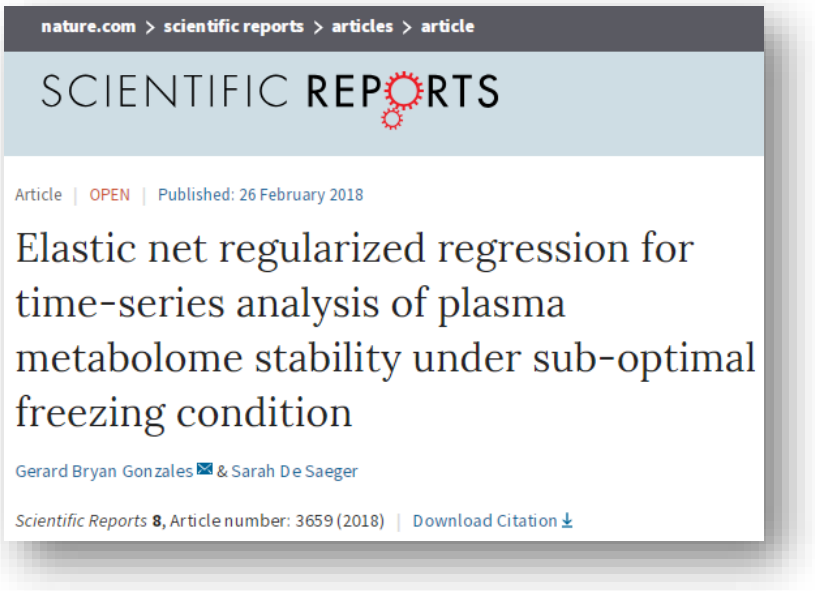
1.7 Regularized regression/classification

Cf. TOTAL approach DHC/FCC TRA

[TO ELABORATE]

Use case example:

https://www.nature.com/articles/s41598-018-21851-7



1.8 Bayesian Structural Time Series Models

http://www.unofficialgoogledatascience.com/2017/07/fitting-bayesian-structural-time-series.html

https://multithreaded.stitchfix.com/blog/2016/04/21/forget-arima/

https://medium.com/@chawannut157/using-bayesian-structural-modelling-to-analyze-cryptocurrencies-d08b6069284c

1.9 Forecasting

1.9.1 Links

https://labs.eleks.com/2016/10/combined-different-methods-create-advanced-time-series-prediction.html

https://towardsdatascience.com/forecasting-fundamentals-you-should-know-before-building-predictive-models-299a18c2093b

https://eng.uber.com/forecasting-introduction/

1.9.2 Fundamentals

Forecasting types in terms of **time horizon** and **data availability**

Will forecasts be required for a few minutes in advance, for 6 months, or for ten years? There are 3 types of forecasting in terms of time horizon:

- Short-term forecasts: a normal range between one and three months
- Medium-term forecasts: the time period is normally one year
- Long-term forecasts: predict results over periods greater than two years

In most forecasting situations, the uncertainty associated with the thing we are forecasting will decrease as the event approaches. In other words, the closer ahead we forecast, the more accurate we are.

Forecasting can be categorized into two types in terms of data availability.

- Qualitative Forecasts: If there are no data available, or if the data available are not relevant to the forecasts.
- Quantitative Forecasts (Time Series Forecasting): If numerical information about the past is available; and the past patterns will continue into the future.

Data for Quantitative Forecasts (Time Series Forecasting) are often observed at regular intervals of time, e.g., hourly, daily, weekly, monthly, quarterly, annually. The goal is to estimate how the sequence of historical observations will continue into the future.

1.9.3 Prominent forecasting approaches (UBER's view)

Apart from qualitative methods, quantitative forecasting approaches can be grouped as follows:

- model-based or causal classical
- statistical methods
- machine learning approaches.

Model-based forecasting is the strongest choice when the underlying mechanism, or physics, of the problem is known, and as such it is the right choice in many scientific and engineering situations at Uber. It is also the usual approach in econometrics, with a broad range of models following different theories.

When the underlying mechanisms are not known or are too complicated, e.g., the stock market, or not fully known, e.g., retail sales, it is usually better to apply a simple statistical model. Popular classical methods that belong to this category include **ARIMA** (autoregressive integrated moving average), exponential smoothing methods, such as **Holt-Winters**, and the **Theta method**, which is less widely used, but performs very well. In fact, the Theta method won the M3 Forecasting Competition, and we also have found it to work well on Uber's time series (moreover, it is computationally cheap).

In recent years, **machine learning approaches**, including quantile regression forests (QRF), the cousins of the well-known random forest, have become part of the forecaster's toolkit. **Recurrent neural networks** (RNNs) have also been shown to be very useful if sufficient data, especially exogenous regressors, are available. Typically, these machine learning models are of a black-box type and are used when interpretability is not a requirement. Below, we offer a high level overview of popular classical and machine learning forecasting methods:

Classical & Statistical	Machine Learning
• Autoregressive integrated moving average (ARIMA) • Exponential smoothing methods (e.g. Holt-Winters) • Theta	• Recurrent neural networks (RNN) • Quantile regression forest (QRF) • Gradient boosting trees (GBM) • Support vector regression (SVR) • Gaussian Process regression (GP)

Choosing the right forecasting method for a given use case is a function of many factors, including how much historical data is available, if exogenous variables (e.g., weather, concerts, etc.) play a big role, and the business needs (for example, does the model need to be interpretable?). The bottom line, however, is that we cannot know for sure which approach will result in the best performance and so it becomes necessary to compare model performance across multiple approaches.

1.9.4 Hierarchical TS forecasting

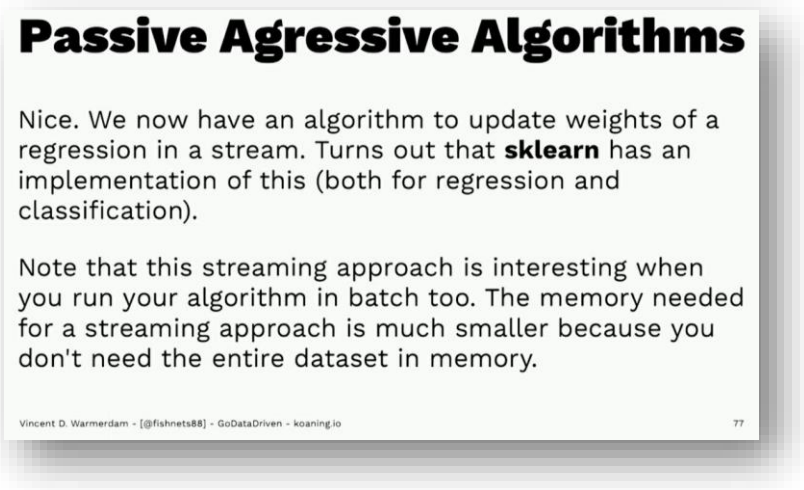
https://robjhyndman.com/tags/hierarchical-time-series/

2 Streaming – Online Machine Learning

2.1 Influx db

https://www.influxdata.com/

2.2 Passive aggressive algorithms



http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PassiveAggressiveClassifier.html

2.3 Vowpal Wabbit

https://github.com/JohnLangford/vowpal_wabbit

3 Anomaly Detection

3.1 References

https://rd.springer.com/chapter/10.1007/978-3-319-47578-3_9

https://www.datascience.com/blog/python-anomaly-detection

3.1.1 R

https://datascienceplus.com/anomaly-detection-in-r-the-tidy-way/

3.1.2 Twitter

https://blog.twitter.com/engineering/en_us/a/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series.html

https://github.com/twitter/AnomalyDetection

https://anomaly.io/anomaly-detection-using-twitter-breakout/

algorithm a Seasonal Hybrid ESD (S-H-ESD)

https://arxiv.org/pdf/1704.07706.pdf

3.1.3 Microsoft

https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/time-series-anomaly-detection

exchangeability martingales (EM)

3.1.4 Anodot

https://www.anodot.com/product/

Automated Anomaly Detection

- Hybrid semi-supervised

Design principles:

- Timeliness/Scale ➔ type of ML algo to use
  - Real-time ➔ online learning: ANODOT
  - Long-term: e.g. predictive maintenance ➔ batch ML
- Rate of change
  - High frequency of change ➔ Highly adaptive algo: ANODOT
- Conciseness
  - Univariate
  - Multivariate: use of a proxy ts encompassing n original ts
  - Hybrid: ANODOT
    - Learn what is normal for each one of the ts by themselves

        - Combine detected anomalies in groups and give an interpretation to that group == combine n anomalies into a single incident if ts are related
- Definition of incidents
    - Well-defined incidents are not always known ➔ supervised vs unsupervised (latter can detect known and unknown types of incident)
    - What is normal ➔ deviation from normality
    - Semi-supervised: ANODOT

Learning normal behaviour:

- Statistical model of normal behavior + statistical test on new samples (are they explained by the model?) ➔ flag as anomaly if test is not passed
    - Example: assuming Gaussian distribution and 3sigma rule: 2 parameters mu/sigma → not general // high bias
    - ➔ find the statistical distribution that best fits the data? Not one type of distribution that fits all possible ts
    - ➔ Classify ts and map the class to a distribution type: CLASSIFICATION PHASE (ANODOT)
        - Dictionary of model types: stationary, multi-modal, discrete, irregular sampling, sparse, step, etc.
    - TS behavior can change overnight + high number of ts ➔ adaptive is required ➔ dynamic classification
- Modeling seasonality
    - Only 14% of ts have seasonal patterns
    - Algo to detect seasonal patterns: ANODOT
        - Fourier – power spectrum/periodogram: not working well for multiple seasonal patterns – Not suited for real-time filtering
        - ACF: compute expensive
        - ANODOT: VIVALDI algo
            - ACF + smart subsampling technique (computing a small subset of ACF coefficients) + applied on multiple filtered versions of the original ts (to detect multiple seasonal patterns)
- Online adaptive learning: ANODOT
    - Impact of learning rate ➔ Auto-tune learning rate: ANODOT
        - Adapt the learning rate by assigning weights to the validity of the data points
            - Point detected as anomaly => very low weight when adapting the model parameters
        - Balance between how fast we learn versus how adaptive we are

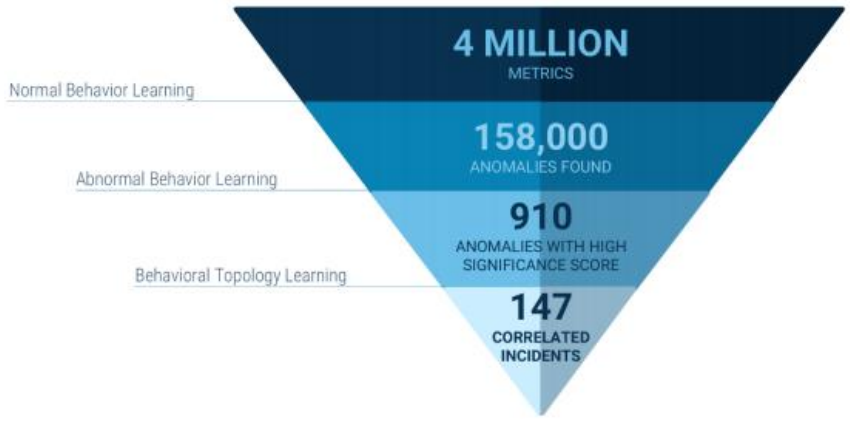
Correlating abnormal behaviour:

- ANODOT: ranking/scoring anomalies + correlating ts together: differentiator ➔ min FP/FN
- Abnormal behavior learning and scoring
    - Notions of deviation and duration of detected anomalies
        - Statistical model:

        - Input = set of statistics related to each anomaly
        - Output = score of how significant the anomaly is
    - ➔ filter anomalies based on their significance
- Behavioral Topology Learning
    - Combining anomalies into a concise story requires an understanding of which ts are related
    - Methods to learn actual relationships among different metrics
        - (1) Abnormal based similarity
            - Clustering – LDA with enhancements : ANODOT (soft clustering)
        - (2) Name similarity
            - On ts names
        - (3) Normal behavior similarity
            - Linear correlation (Pearson correlation coefficient) – de-trending and de-season is required → FP!
            - Pattern dictionary approach (typical shapes assigned to segments) + TF-IDF – creation of the dictionary with stacked auto-encoders
        - (4) User input
            - Direct/indirect

Scaling

- Many ts ➔ group them with LSH algo
- Online learning on the data stream! Cf. architecture
- + offline learning of behavioral topology/seasonality
- Funnel



Human experience - dashboard

3.1.5 Anomaly.io

https://anomaly.io/detect-anomaly/

4 Structural Change Detection

5 R

5.1 Useful links

https://cran.r-project.org/web/views/TimeSeries.html

https://www.otexts.org/fpp/

*https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/

http://www.kdnuggets.com/2016/11/combining-different-methods-create-advanced-time-series-prediction.html

http://www.kdnuggets.com/2017/01/time-series-analysis-primer.html

http://www.kdnuggets.com/2017/01/datascience-introduction-forecasting-arima-r.html

http://robjhyndman.com/hyndsight/tscvexample/

http://www.kdnuggets.com/2017/04/time-series-analysis-generalized-additive-models.html

https://thuijskens.github.io/2016/08/03/time-series-forecasting/

https://stats.stackexchange.com/questions/60939/classification-in-time-series-svms-neural-networks-random-forests-or-non-para

5.1.1 Datetime objects

http://neondataskills.org/R/time-series-convert-date-time-class-POSIX/

5.1.2 Time series and forecasting

http://www.statmethods.net/advstats/timeseries.html

https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/

https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/

http://www.stat.pitt.edu/stoffer/tsa4/R_toot.htm

5.1.3 Packages

5.1.3.1 Xts package

https://www.datacamp.com/community/blog/r-xts-cheat-sheet#gs.O71NQjY

https://www.rdocumentation.org/packages/xts/versions/0.10-0

https://github.com/joshuaulrich/xts

http://joshuaulrich.github.io/xts/

5.1.3.2 Forecastxgb

https://github.com/ellisp/forecastxgb-r-package

5.2 Read csv with timestamps

5.2.1 Compare read.csv with read.zoo

Example of csv file with timestamps (PI tags)

```
"time","PI:PDC670040.PV","PI:PDC670050.PV"
2010-01-01T00:00:00.0000000,,0.28241298
2010-01-01T00:01:00.0000000,0.21244586,0.28
2010-01-01T00:02:00.0000000,0.21572177,0.28
2010-01-01T00:03:00.0000000,0.22079334,0.28
2010-01-01T00:04:00.0000000,0.2050449,0.288
```

Read.csv proceeds in two steps:

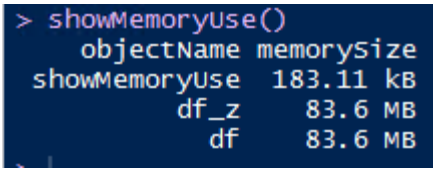
- Reading
- Setting date format (timestamp)

Read.zoo directly sets the format as specified.

```
> setwd("S:/Projects/xx-FCC TRA Stability Troubleshooting/pi_data")
> system.time({
+ # Get data (read csv)
+ df <- read.csv("./pitags_part3.csv", stringsAsFactors=FALSE, fileEncoding="utf-8")
+ # str(df$time)
+ df$time <- as.POSIXct(df$time, format = "%Y-%m-%dT%H:%M:%OS", tz = "GMT")
+ })
   user  system elapsed
  47.96    0.11   49.44
> system.time({
+   df_z <- read.zoo("./pitags_part3.csv", header=TRUE, sep=',',
+                    index = "time",
+                    format = "%Y-%m-%dT%H:%M:%OS", tz = "GMT")
+ })
   user  system elapsed
  42.93    0.11   45.17
```

```
Data
df                          3652385 obs. of 3 variables
   time : POSIXct, format: "2010-01-01 00:00:00" "2010-01-01 00:01:00" "20
   PI.PDC670040.PV: num NA 0.212 0.216 0.221 0.205 ...
   PI.PDC670050.PV: num 0.282 0.288 0.282 0.286 0.288 ...
Values
df_z                        Large zoo (7304770 elements, 83.6 Mb)
   Data: num [1:3652385, 1:2] NA 0.212 0.216 0.221 0.205 ...
   attr(*, "dimnames")=List of 2
   ..$ : NULL
   ..$ : chr [1:2] "PI.PDC670040.PV" "PI.PDC670050.PV"
   Index: POSIXct[1:3652385], format: "2010-01-01 00:00:00" "2010-01-01 00
```

Memory in both cases:


> showMemoryUse()
objectName memorySize
showMemoryUse 183.11 kB
df_z 83.6 MB
df 83.6 MB

6 Python

6.1 Useful links

***https://medium.com/open-machine-learning-course/open-machine-learning-course-topic-9-time-series-analysis-in-python-a270cb05e0b3

**https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/

https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

**https://www.kaggle.com/thebrownviking20/everything-you-can-do-with-a-time-series/notebook

**https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/

***https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/

**https://www.datacamp.com/community/tutorials/time-series-analysis-tutorial

https://www.statsmodels.org/dev/examples/notebooks/generated/statespace_sarimax_stata.html

https://towardsdatascience.com/time-series-analysis-in-python-an-introduction-70d5a5b1d52a

https://www.elenacuoco.com/2016/09/22/power-spectral-density/

http://pbpython.com/pandas-grouper-agg.html

https://tomaugspurger.github.io/modern-7-timeseries.html

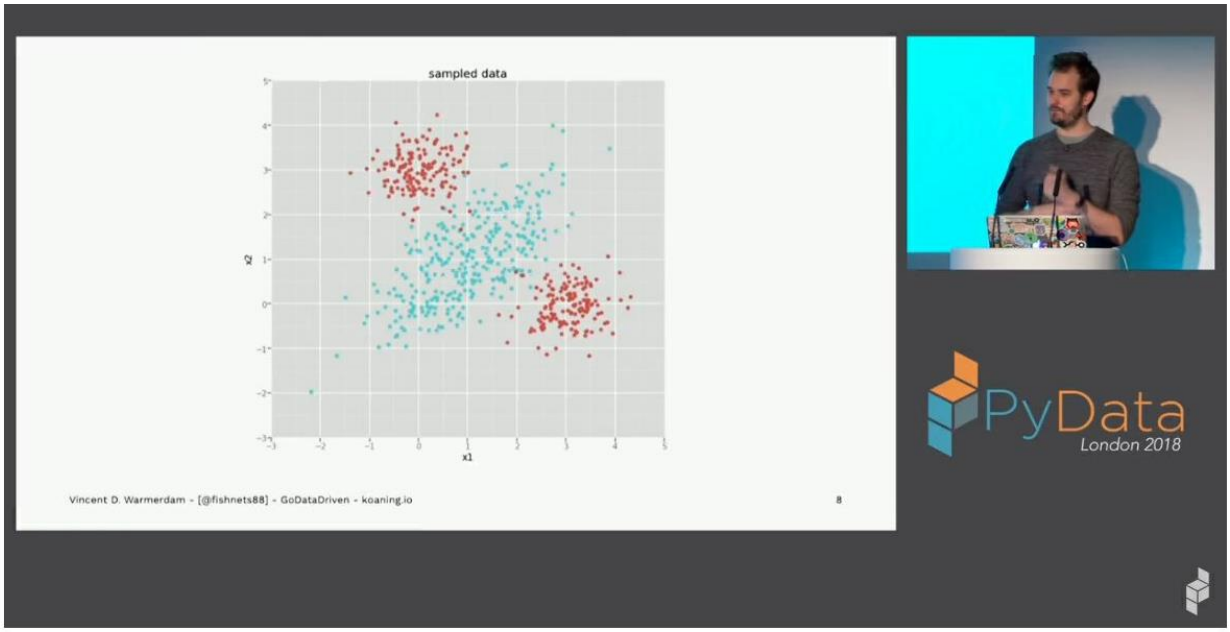
https://www.analyticsvidhya.com/blog/2016/02/hand-learn-time-series-3-hours-mini-datahack/

https://conference.scipy.org/scipy2011/slides/mckinney_time_series.pdf

**https://labs.eleks.com/2016/10/combined-different-methods-create-advanced-time-series-prediction.html

- Decomposition in Trend + Seas + Res
    - Trend : Hodrick-Prescott Filter
    - Seas: STL (Loess)
- Fit trend
    - Enet
    - Forecast with Fourier extrapolation (discrete Fourier transform)
- NOT CONVINCING

***https://www.youtube.com/watch?v=68ABAU_V8qI



Winning with Simple, even Linear, Models - Vincent D. Warmerdam

Offset Aliases

A number of string aliases are given to useful common time series f
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
SM	semi-month end frequency (15th and end of month)
BM	business month end frequency
CBM	custom business month end frequency
MS	month start frequency
SMS	semi-month start frequency (1st and 15th)
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

6.2 Packages

6.2.1 Statsmodels

https://www.statsmodels.org/stable/tsa.html

6.2.2 Prophet (also R)

Univariate forecasting!

Paper: https://peerj.com/preprints/3190/

IMPORTANT:

- Business Time Series
  - Trend + Seasonality + Holidays
- Forecasting as a curve-fitting exercise
  - ➔ no root cause analysis / no influence of exogeneous variables

https://github.com/facebook/prophet

https://facebook.github.io/prophet/

https://www.youtube.com/watch?v=95-HMzxsghY

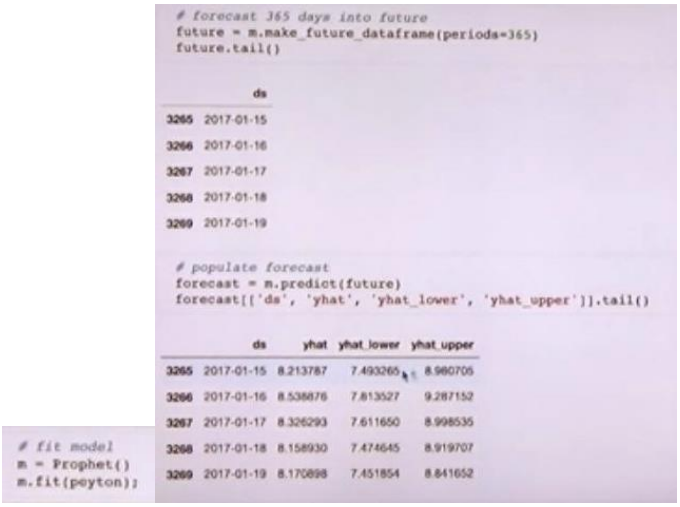
https://towardsdatascience.com/time-series-analysis-in-python-an-introduction-70d5a5b1d52a

Based on GAM:

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

http://environmentalcomputing.net/intro-to-gams/

Simple as that: (sklearn)



```
# fit model
m = Prophet()
m.fit(peyton);
```

```
# forecast 365 days into future
future = m.make_future_dataframe(periods=365)
future.tail()
```

	ds
3265	2017-01-15
3266	2017-01-16
3267	2017-01-17
3268	2017-01-18
3269	2017-01-19

```
# populate forecast
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
3265	2017-01-15	8.213787	7.493265	8.980705
3266	2017-01-16	8.538876	7.813527	9.287152
3267	2017-01-17	8.326293	7.611650	8.998535
3268	2017-01-18	8.158930	7.474645	8.919707
3269	2017-01-19	8.170898	7.451854	8.841652

Restrictions:

- Works best on a daily level
-

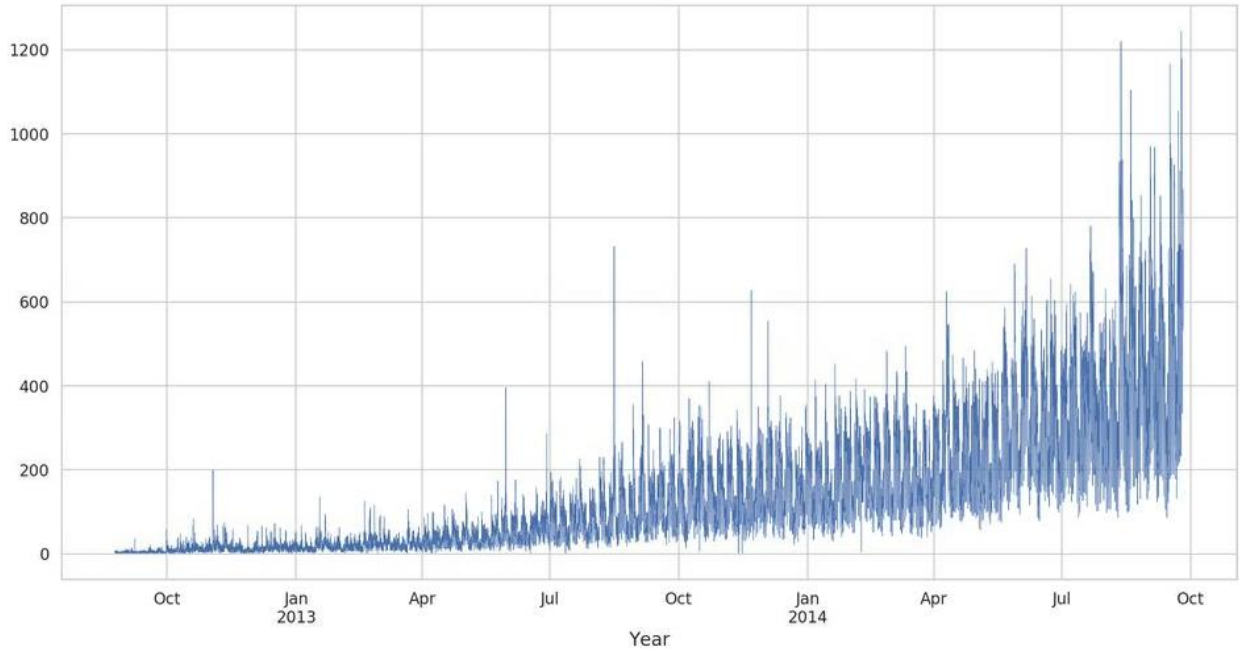
6.3 Misc

6.3.1 Forecast stock exchange

https://www.quantinsti.com/blog/forecasting-markets-using-extreme-gradient-boosting-xgboost/

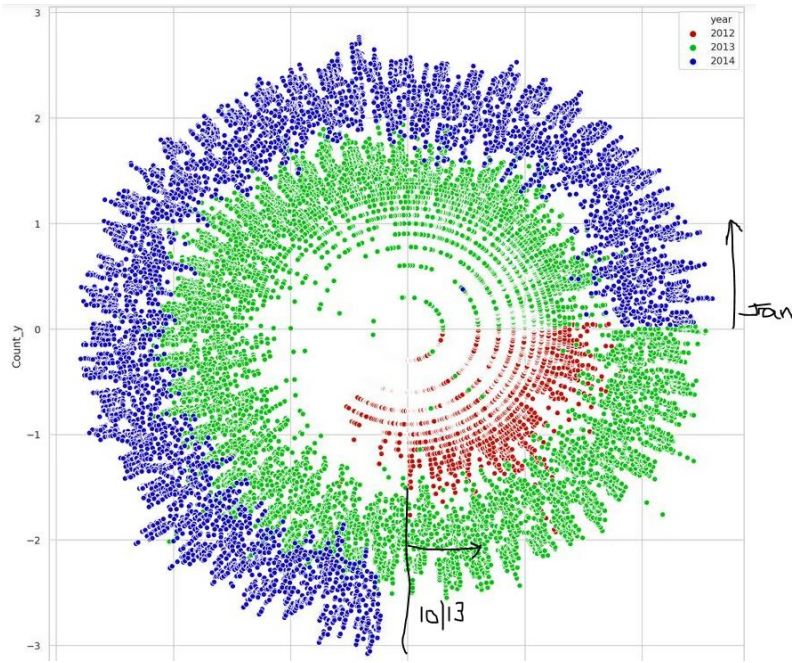
Classification problem (up/down)

6.4 Use case

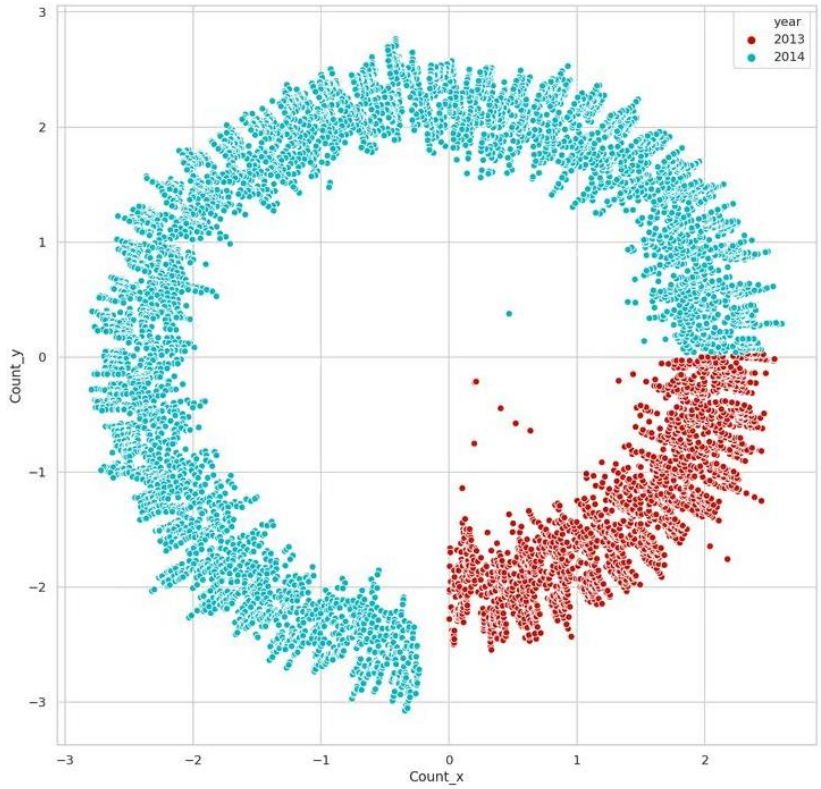


Non-stationary (non-linear trend) and increasing variance

Also seen in polar corrdinates



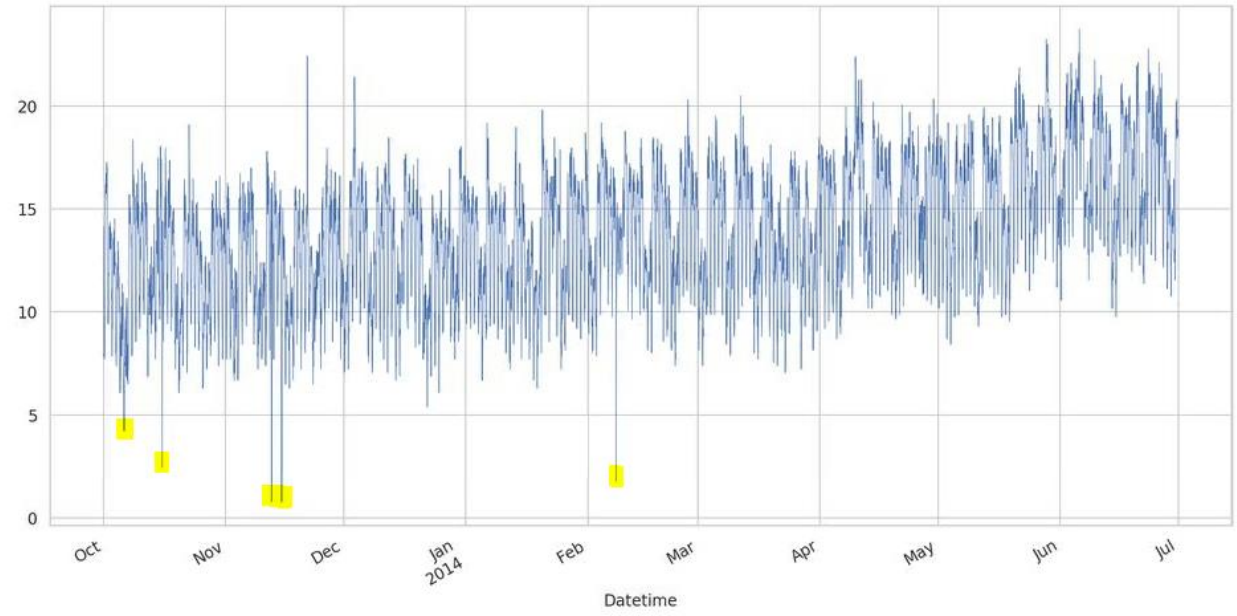
⇨ Certainly discard 2012 (red) and at least 3 quarters of 2013



We could replace outliers (cf above plot) by a moving average/smoother

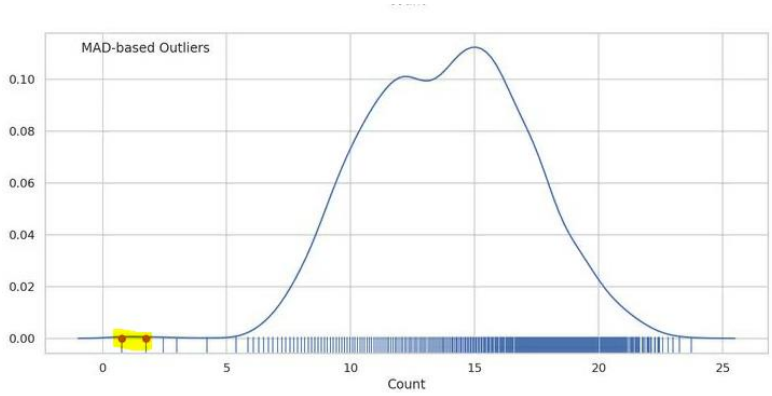
We split train till 06.2014 – test as of 07.2014

We stabilize the increasing variance by a Box-Cox transformation -> lambda = 1/3 ➔ cubic square root



Determine lambda on train and apply on test

Outlier test (MAD-based) on the training set:




```
df_train_box[mad_based_outlier(df_train_box.Count,thresh=3.0)]
```

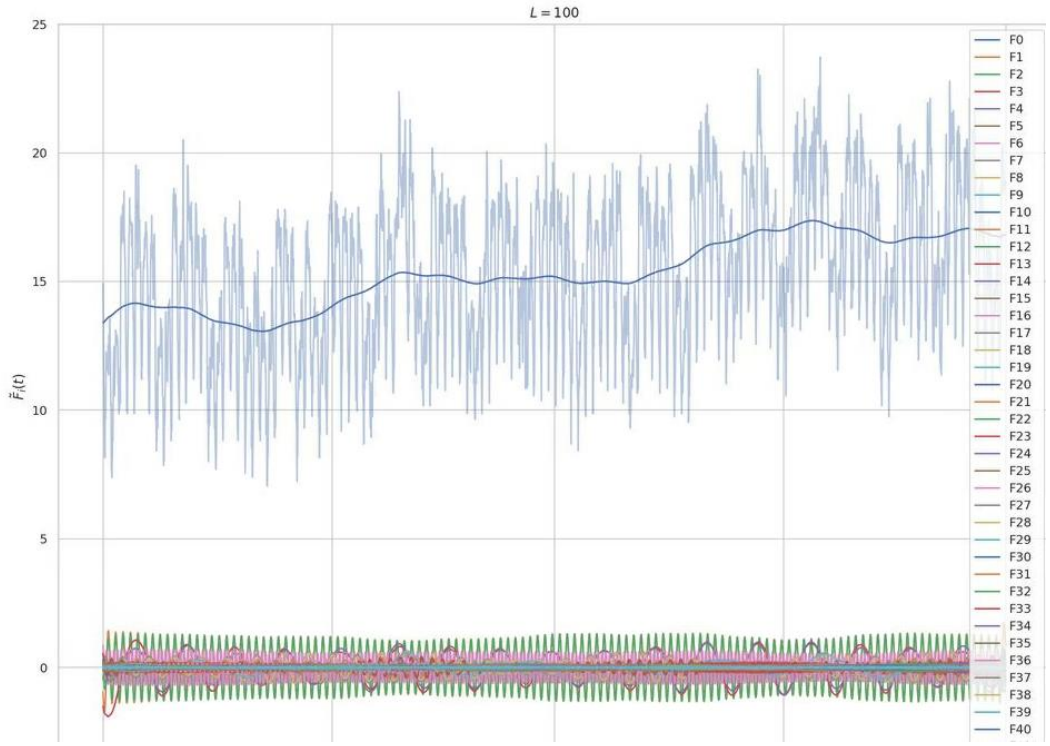
	Count	hour	day	weekday	dayofyear	week	month	quarter	year
Datetime									
2013-10-15 21:00:00	2.445370	21	15	1	288	42	10	4	2013
2013-11-12 15:00:00	2.445370	15	12	1	316	46	11	4	2013
2013-11-12 16:00:00	0.779066	16	12	1	316	46	11	4	2013
2013-11-12 17:00:00	1.758937	17	12	1	316	46	11	4	2013
2013-11-12 18:00:00	1.758937	18	12	1	316	46	11	4	2013
2013-11-15 04:00:00	0.779066	4	15	4	319	46	11	4	2013
2013-11-15 05:00:00	0.779066	5	15	4	319	46	11	4	2013
2013-11-15 06:00:00	0.779066	6	15	4	319	46	11	4	2013
2013-11-15 07:00:00	0.779066	7	15	4	319	46	11	4	2013
2013-11-15 08:00:00	2.991371	8	15	4	319	46	11	4	2013
2014-02-08 05:00:00	1.758937	5	8	5	39	6	2	1	2014

Replace count by value same day/hour week before

Due to multiple seasonality: daily, weekly for sure, standard methods are not that accurate.

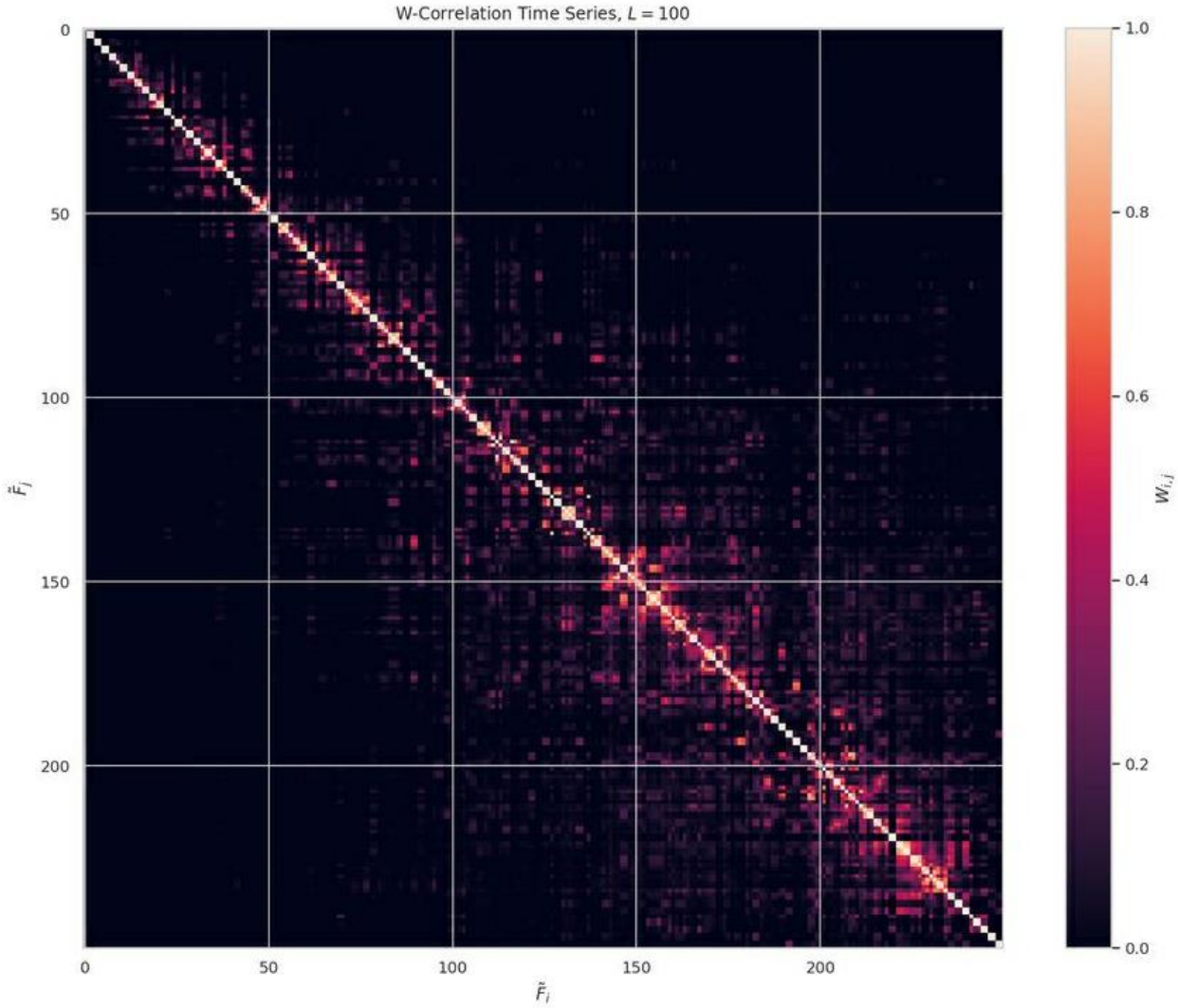
Testing SSA decomposition

250 components:

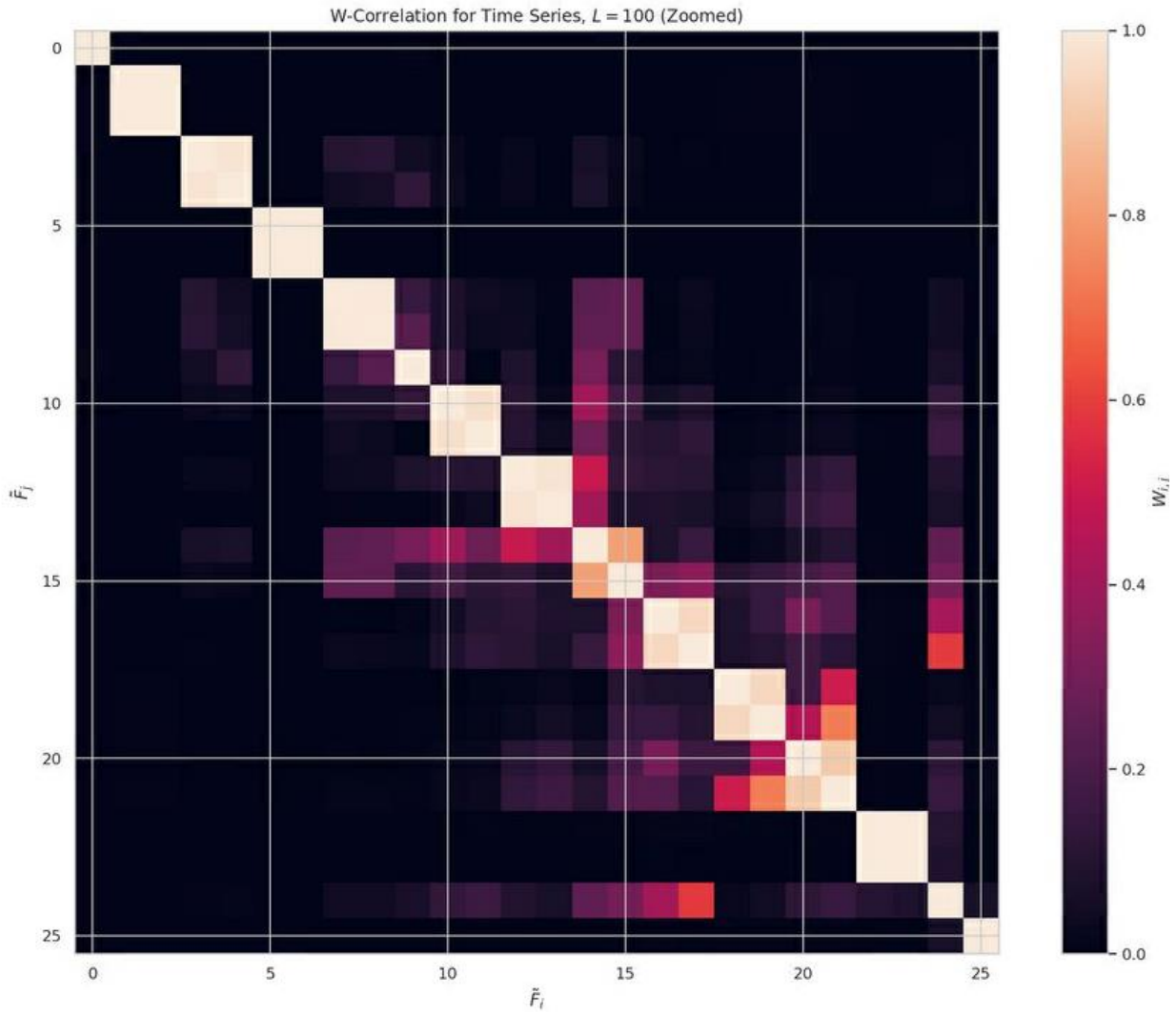


Trend is clearly separated from the periodic components

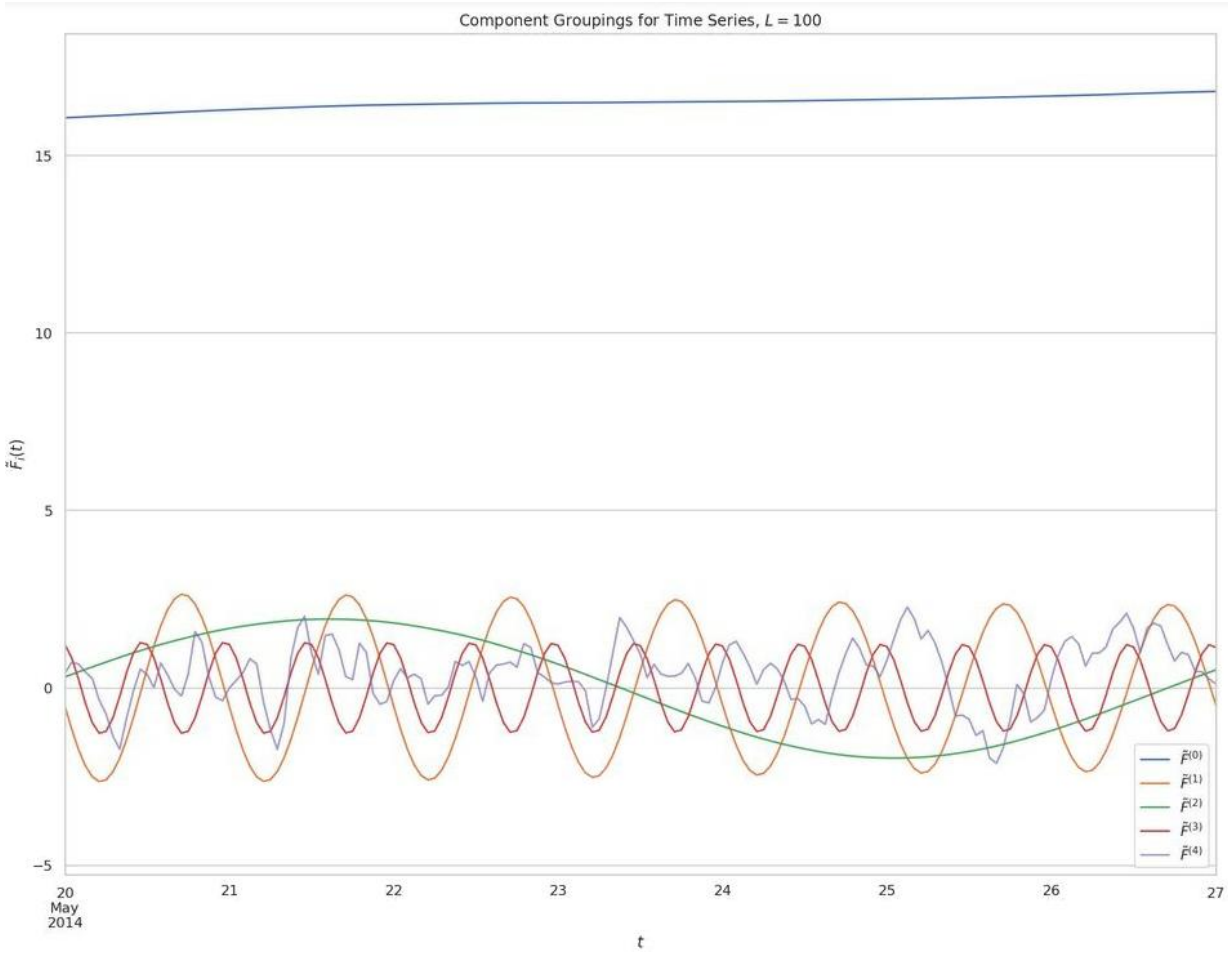
Correlation of elementary matrices:



Zoom on 25 first:



Shows that F1+F2, F5+F6, and F3+F4 should be grouped resp. (although small correlations between 3,4 and > 7



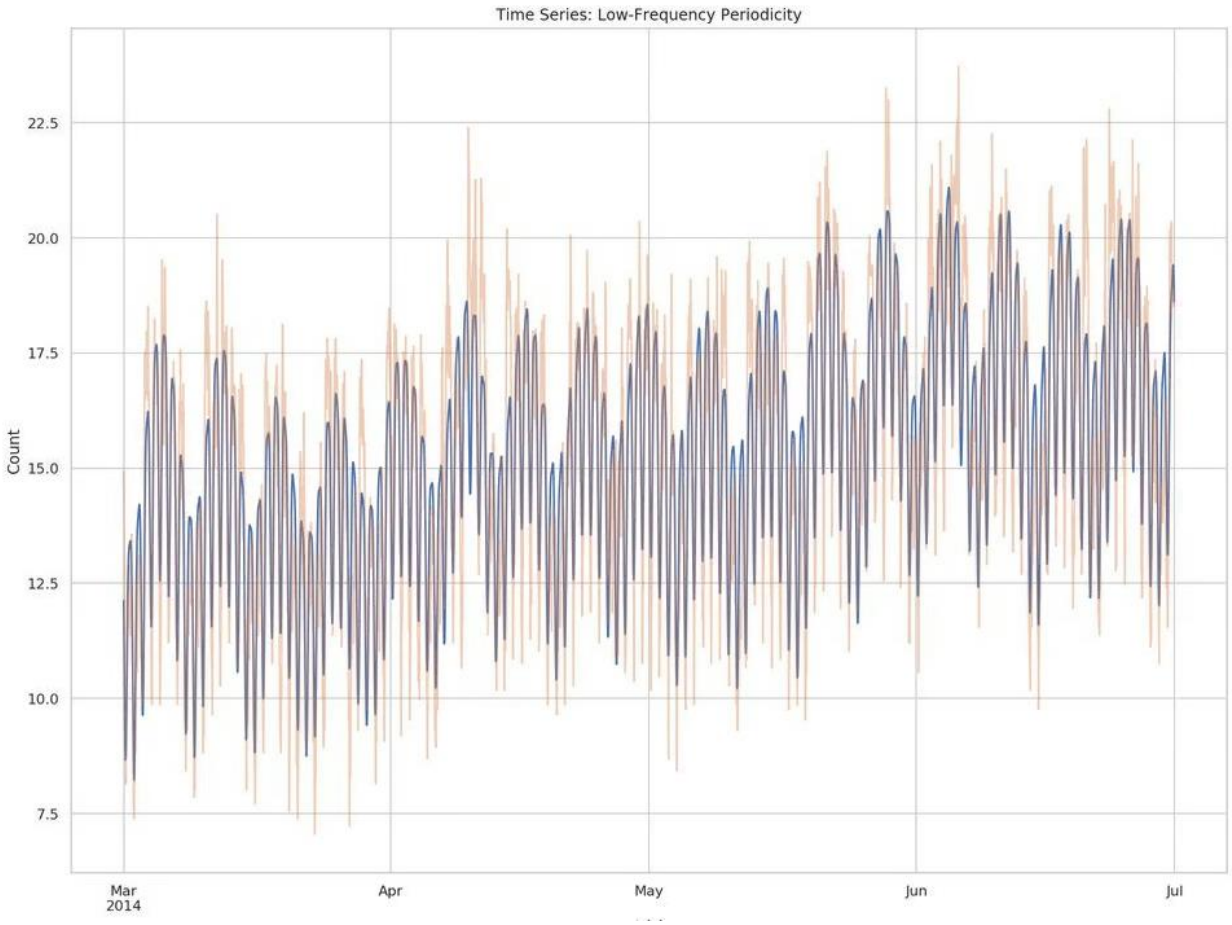
F1 = first grouping (daily – period = 24)

F2 = second grouping (weekly)

F3 = third grouping (subdaily period=12)

F4 = remainders = periodic smaller components + noise

Reconstructing signal from 6 first components:



One sees that the ssa captures the main variability up to noise uncertainties.

Zoom: