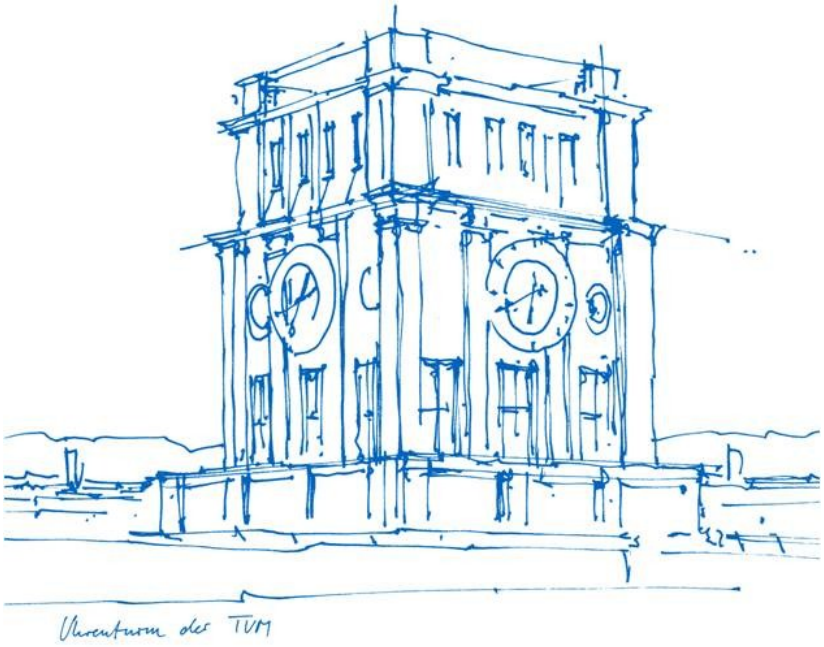
Forecasting the Students Count

Mohak Chadha



About Me

- **Education**
  - B.E. (Hons.) Computer Science and M.Sc. (Hons.) Mathematics from BITS – Pilani, India – August 2017
  - M.Sc. Informatik from TUM – April 2020
  - PhD Candidate at TUM since July 2020
- **Research Areas**
  - Systems for ML using FaaS
  - Extending FaaS to heterogenous computing
  - Performance optimization and analysis across the FaaS stack.
  - Edge Computing
- **Contact**
  - https://kky-fury.github.io
  - https://www.ce.cit.tum.de/caps/theses/open/#c47732

Forecasting Models:
- Linear regression
- ARIMA
- other models

Background: Time Series

Time series is a discrete time-ordered sequence of numerical values

Example – temperature average for 5-minutes intervals

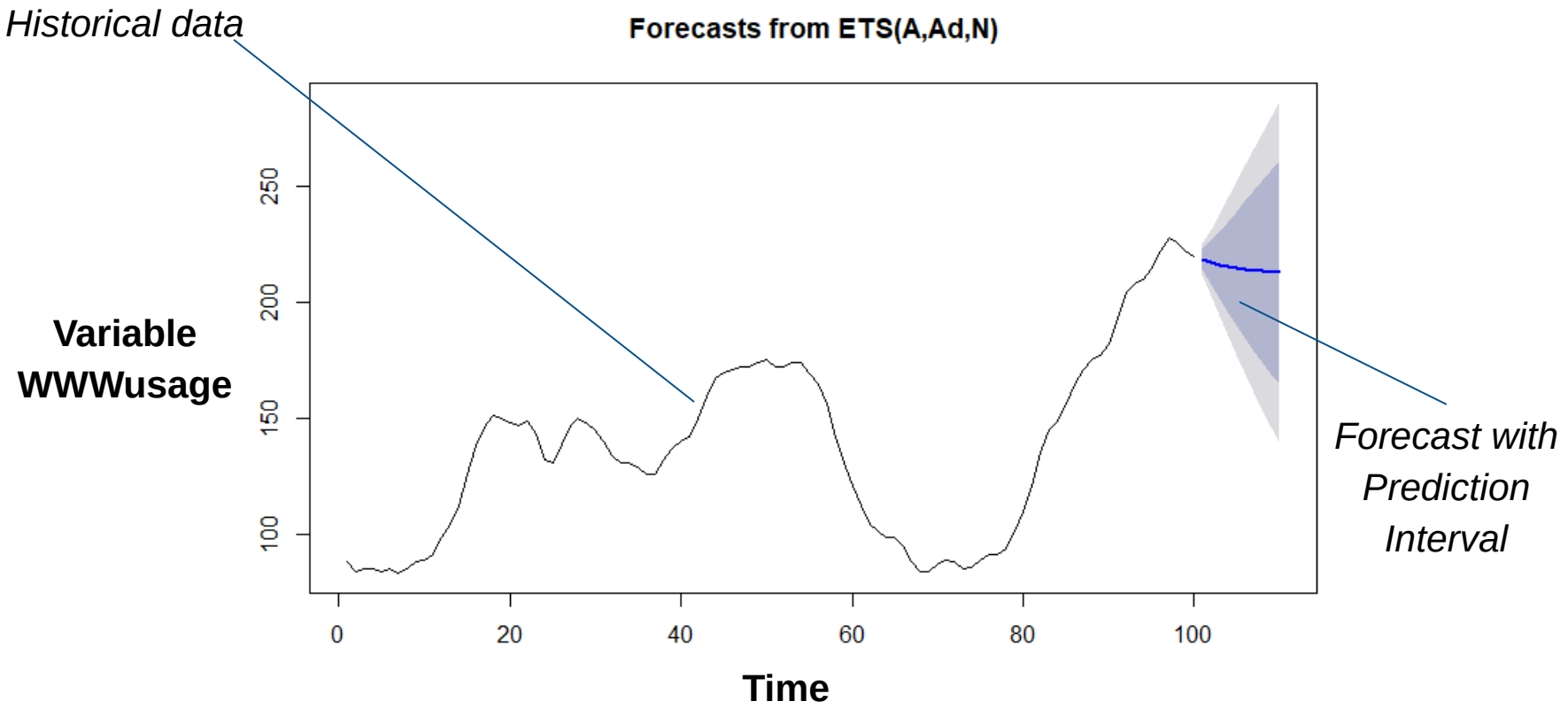
Time	12:00:00	12:05:00	12:10:00	12:15:00	…
Temperature	25.7	24.8	25.5	25.2	…

A time series can be acquired by measuring the values of some time-dependent variable at specific points in time or with some frequency:

...

Background: Forecasting for Time Series

Forecasting for time series leverages the previous measured values of some variable to provide an estimate of the future values of the same variable.
Although the model is important, the quantity and the quality of the data is more important



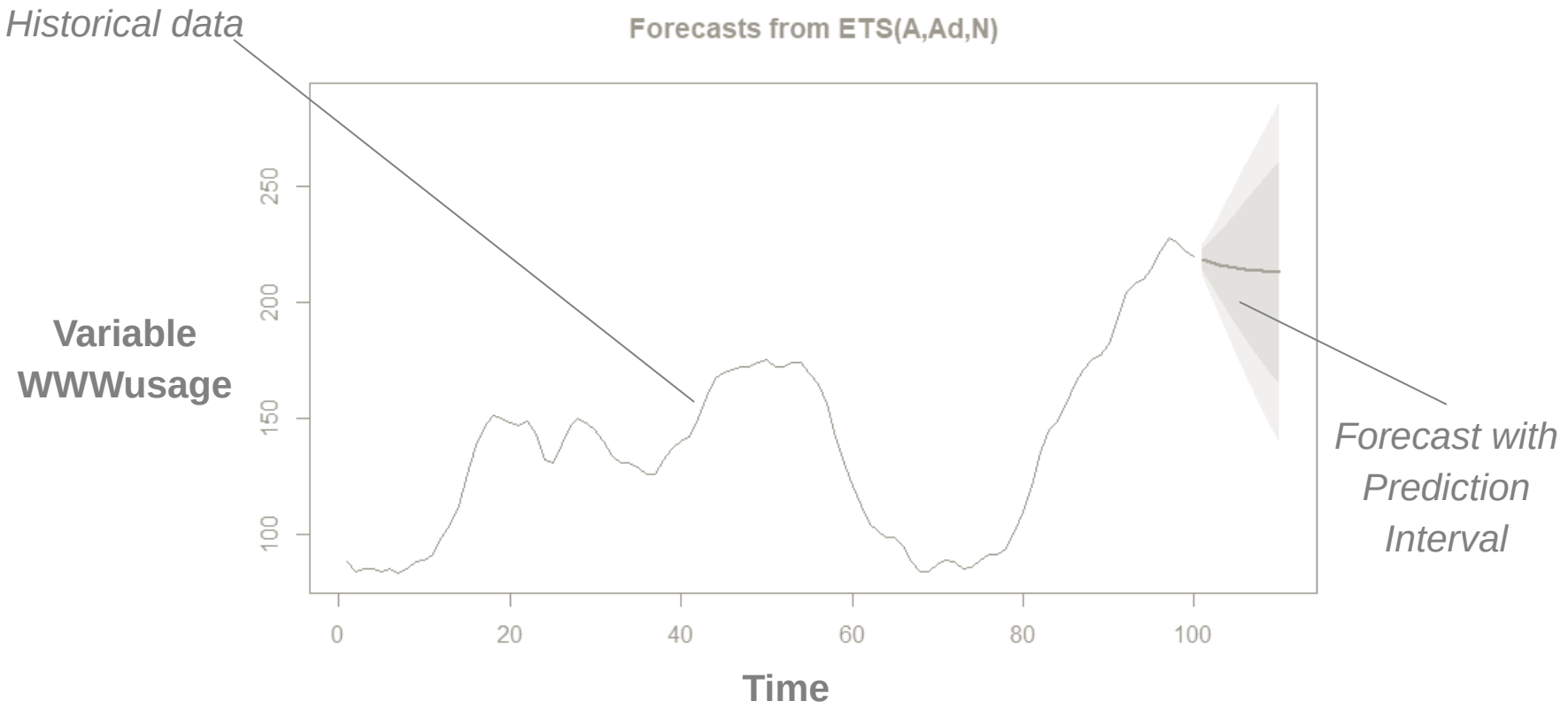
[produced by R package *forecast* version 8.2]

Background: Forecasting for Time Series

Forecasting for time series leverages the previous measured values of some variable to provide an estimate of the future values of the same variable.

Although the model is important, the quantity and the quality of the data is more important

Note: we are dealing with the univariate fixed-step time series forecasting!



[produced by R package *forecast* version 8.2]

Background: Forecasting Models

Forecasting Models have different mathematical apparatus and are based on various assumptions about the time series.

Following, some of these models are briefly introduced.

Linear Regression

Linear regression assumes the *linear relation* between the *dependent variable* (e.g. count of people) and the vector of *independent variables,* or regressors (e.g. values of the same variable at the previous timestamps). Sample formula:

$$x = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Linear regression need not be linear in regressors, but coefficients should be → we're perfectly fine with $x^2, x^3, \ldots, x^n$ being our features if needed, also *dummy variables*.

Linear Regression

Linear regression assumes the *linear relation* between the *dependent variable* (e.g. count of people) and the vector of *independent variables,* or regressors (e.g. values of the same variable at the previous timestamps). Sample formula:

$$x = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Linear regression need not be linear in regressors, but coefficients should be → we're perfectly fine with $x^2, x^3, \ldots, x^n$ being our features if needed, also *dummy variables*.

Assumptions to apply the linear regression:

Mean of the dependent (predicted) variable is the *linear combination* of the parameters (betas) with the features (x)

predictor variables are error-free, measurements are perfect

constant variance in the errors of the dependent variable; how to check – look at the *residuals* – if they systematically diverge when plotted against the predictive variables, then the variance is not constant

errors of dependent variables are uncorrelated

lack of perfect multicollinearity, i.e. number of parameters to estimate should be smaller than the available data, no independent variables should be correlated

Linear Regression: When/How to Apply

Linear regression is a working horse of the analyst. If the assumptions to its application at least *roughly hold*, then it can give some good insights about the behavior of the response variable. Otherwise, it can be misleading.
This actually applies to any model, if you do not check its assumptions/limitations.

If the linear regression gives a good result, leave the more sophisticated models aside! They should only be used if they give some added value.

Linear Regression: When/How to Apply

Linear regression is a working horse of the analyst. If the assumptions to its application at least *roughly hold*, then it can give some good insights about the behavior of the response variable. Otherwise, it can be misleading.
This actually applies to any model, if you do not check its assumptions/limitations.

If the linear regression gives a good result, leave the more sophisticated models aside! They should only be used if they give some added value.

How to apply to time series:

- plot your time series and spot the patterns
- check whether the assumptions hold for your TS
- if some of them don't, then try to adjust your TS

(e.g. via taking a logarithm to get better variance)

- plot your adjusted time series again
- track visually how observations relate to ones that

are *just behind them* and the ones that are *way behind*

- assemble a meaningful set of *time lags* to check,

e.g. values of the variable 1 step back, 12 steps back…

Linear Regression: When/How to Apply

Linear regression is a working horse of the analyst. If the assumptions to its application at least *roughly hold*, then it can give some good insights about the behavior of the response variable. Otherwise, it can be misleading.
This actually applies to any model, if you do not check its assumptions/limitations.

If the linear regression gives a good result, leave the more sophisticated models aside! They should only be used if they give some added value.

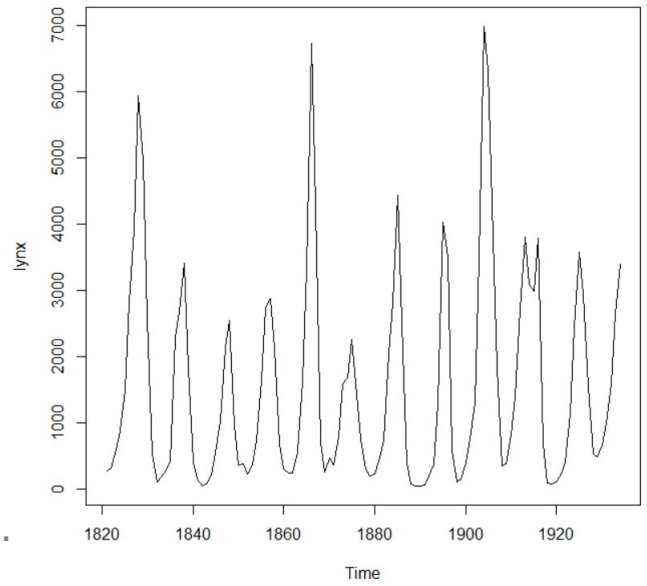
How to apply to time series:

- plot your time series and spot the patterns
- check whether the assumptions hold for your TS
- if some of them don't, then try to adjust your TS

(e.g. via taking a logarithm to get better variance)

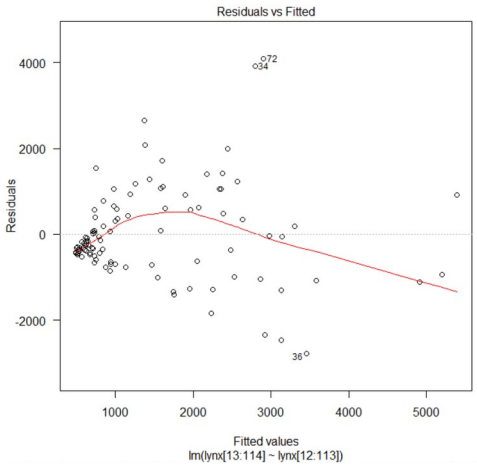
- plot your adjusted time series again
- track visually how observations relate to ones that are *just behind them* and the ones that are *way behind*
- assemble a meaningful set of *time lags* to check, e.g. values of the variable 1 step back, 12 steps back…

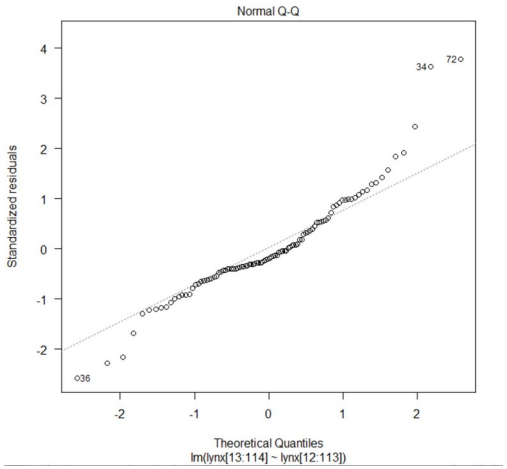
Lynx population in Canada – R dataset



Linear Regression: Intuitions for Modeling

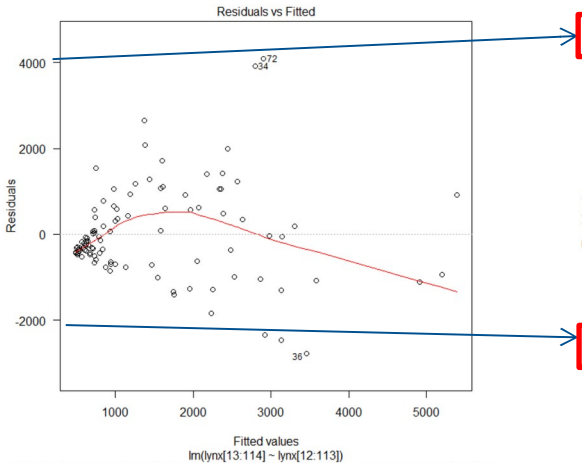
Depends on the previous

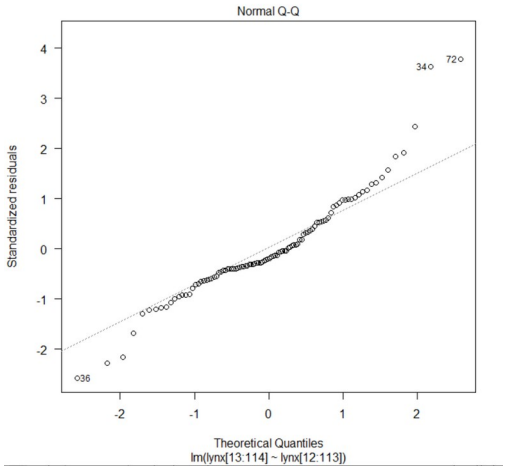




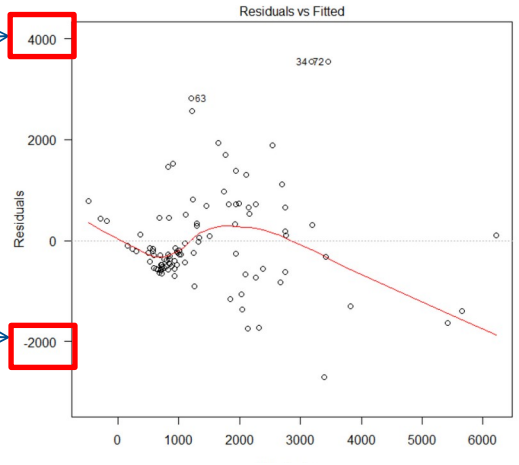
Linear Regression: Intuitions for Modeling

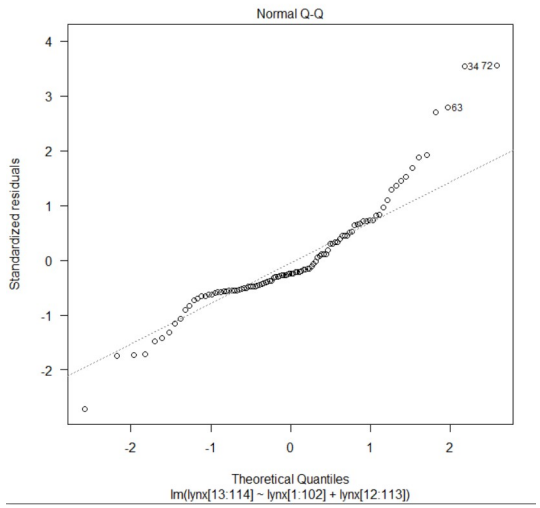
Depends on the previous





Depends on the previous & period



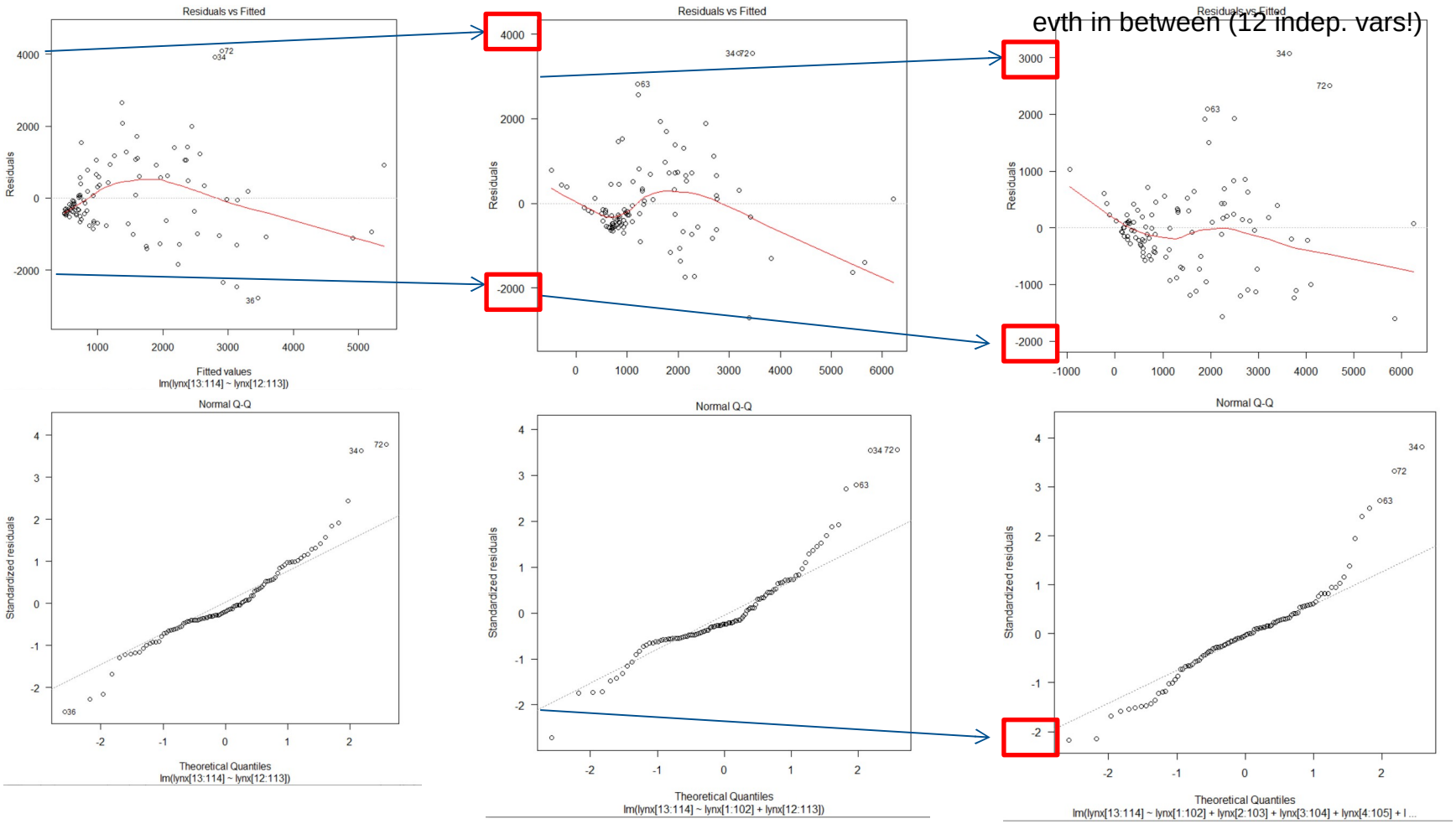


Linear Regression: Intuitions for Modeling

Depends on the previous

Depends on the previous & period

Depends on the previous & period
&
evth in between (12 indep. vars!)



Autoregression

Autoregressive (AR) models represent random processes (e.g. occupancy of the seminar room). In particular, an AR model specifies that the variable of the process depends *linearly* on its past values and on a *stochastic term*. AR is close to LR model.

Autoregression

Autoregressive (AR) models represent random processes (e.g. occupancy of the seminar room). In particular, an AR model specifies that the variable of the process depends *linearly* on its past values and on a *stochastic term*. AR is close to LR model.

AR model is specified by giving a single parameter **p** (lag, order) which corresponds to the number of consequent observations of the variable made in the past:

Autoregression

Autoregressive (AR) models represent random processes (e.g. occupancy of the seminar room). In particular, an AR model specifies that the variable of the process depends *linearly* on its past values and on a *stochastic term*. AR is close to LR model.

AR model is specified by giving a single parameter **p** (lag, order) which corresponds to the number of consequent observations of the variable made in the past:

Order of the model (lag)

Backward shift operator

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} + \epsilon_t = c + x_t \sum_{i=1}^{p} \varphi_i B^i + \epsilon_t$$

Model parameters

White noise (hopefully!)

Backward shift notation

Autoregression

Autoregressive (AR) models represent random processes (e.g. occupancy of the seminar room). In particular, an AR model specifies that the variable of the process depends *linearly* on its past values and on a *stochastic term*. AR is close to LR model.

AR model is specified by giving a single parameter **p** (lag, order) which corresponds to the number of consequent observations of the variable made in the past:

Order of the model (lag)

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} + \varepsilon_t = c + x_t \sum_{i=1}^{p} \varphi_i B^i + \varepsilon_t$$

Backward shift operator

Model parameters

White noise (hopefully!)

Backward shift notation

As the structure of the model is standard, you can simply write AR(p).
For instance, if the value of the variable depends on its own value on two previous steps, then you write AR(2).

Moving Average

Moving average (MA) models serve the same goal – they represent random processes such as occupancy of the room over the time. However, instead of using the past values of the variable in question, MA model specifies the variable as depending linearly on the values of the ***stochastic term***, both current and past.

Moving Average

Moving average (MA) models serve the same goal – they represent random processes such as occupancy of the room over the time. However, instead of using the past values of the variable in question, MA model specifies the variable as depending linearly on the values of the ***stochastic term***, both current and past.

MA model is specified by giving a single parameter **q** which corresponds to the number of *past* consequent observations of the stochastic term for the variable:

Moving Average

Moving average (MA) models serve the same goal – they represent random processes such as occupancy of the room over the time. However, instead of using the past values of the variable in question, MA model specifies the variable as depending linearly on the values of the ***stochastic term***, both current and past.

MA model is specified by giving a single parameter **q** which corresponds to the number of *past* consequent observations of the stochastic term for the variable:

Order of the model (lag)

$$x_t = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-1} + \varepsilon_t = \mu + \varepsilon_t \left[\sum_{i=1}^{q} \theta_i B^i + 1 \right]$$

Mean of the series

Model parameters

White noise error terms

As the structure of the model is standard, you can simply write MA(q), e.g. MA(6).

Moving Average

Moving average (MA) models serve the same goal – they represent random processes such as occupancy of the room over the time. However, instead of using the past values of the variable in question, MA model specifies the variable as depending linearly on the values of the ***stochastic term***, both current and past.

MA model is specified by giving a single parameter **q** which corresponds to the number of *past* consequent observations of the stochastic term for the variable:

Order of the model (lag)

$$x_t = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-1} + \varepsilon_t = \mu + \varepsilon_t \left[\sum_{i=1}^{q} \theta_i B^i + 1 \right]$$

Mean of the series

Model parameters

White noise error terms

As the structure of the model is standard, you can simply write MA(q), e.g. MA(6).

Note, that due to the smaller compounding effects of error terms, you can find yourself using higher order for MA models than for AR models, this is ok.

Autoregressive Moving Average (ARMA)

In the real world, random processes are quite complex. They both build on their *past states* and drag along the *accumulated past errors* from some unaccounted processes. The combination of AR and MA models, known as ARMA, represents such a case.

Autoregressive Moving Average (ARMA)

In the real world, random processes are quite complex. They both build on their *past states* and drag along the *accumulated past errors* from some unaccounted processes. The combination of AR and MA models, known as ARMA, represents such a case.

As the combination of AR with MA, ARMA's structure is defined by both **p** and **q**:

Autoregressive Moving Average (ARMA)

In the real world, random processes are quite complex. They both build on their *past states* and drag along the *accumulated past errors* from some unaccounted processes. The combination of AR and MA models, known as ARMA, represents such a case.

As the combination of AR with MA, ARMA's structure is defined by both **p** and **q**:

$$x_t - \varphi_1 x_{t-1} - \ldots - \varphi_p x_{t-p} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_1 \varepsilon_{t-q}$$

The difference between the value of x at time t and the regression on its p previous values, or AR(p),

…can be represented…

… as the linear combination of the q previous and one current white noise error terms, or MA(q)

Autoregressive Moving Average (ARMA)

In the real world, random processes are quite complex. They both build on their *past states* and drag along the *accumulated past errors* from some unaccounted processes. The combination of AR and MA models, known as ARMA, represents such a case.

As the combination of AR with MA, ARMA's structure is defined by both **p** and **q**:

$$x_t - \varphi_1 x_{t-1} - \ldots - \varphi_p x_{t-p} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_1 \varepsilon_{t-q}$$

The difference between the value of x at time t and the regression on its p previous values, or AR(p),

…can be represented…

… as the linear combination of the q previous and one current white noise error terms, or MA(q)

$$x_t - AR(p) = MA(q) \rightarrow x_t = AR(p) + MA(q)$$

Autoregressive Moving Average (ARMA)

In the real world, random processes are quite complex. They both build on their *past states* and drag along the *accumulated past errors* from some unaccounted processes. The combination of AR and MA models, known as ARMA, represents such a case.

As the combination of AR with MA, ARMA's structure is defined by both **p** and **q**:

$$x_t - \varphi_1 x_{t-1} - \ldots - \varphi_p x_{t-p} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_1 \varepsilon_{t-q}$$

The difference between the value of x at time t and the regression on its p previous values, or AR(p), …can be represented… … as the linear combination of the q previous and one current white noise error terms, or MA(q)

$$x_t - AR(p) = MA(q) \rightarrow x_t = AR(p) + MA(q)$$

This model is denoted as ARMA(p, q). It is only good for *stationary* time series, i.e. time series with its properties independent of the time at which the time series is observed. Meaning, no matter which **p** and **q** values you pick, the pattern will persist over the whole time series without the distortion from the side of changing mean or variance.

Autoregressive Moving Average (ARMA)

In the real world, random processes are quite complex. They both build on their *past states* and drag along the *accumulated past errors* from some unaccounted processes. The combination of AR and MA models, known as ARMA, represents such a case.

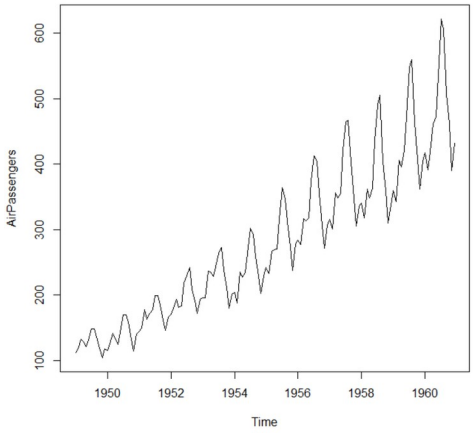
As the combination of AR with MA, ARMA's structure is defined by both **p** and **q**:

$$x_t - \varphi_1 x_{t-1} - \ldots - \varphi_p x_{t-p} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_1 \varepsilon_{t-q}$$

The difference between the value of x at time t and the regression on its p previous values, or AR(p), …can be represented… … as the linear combination of the q previous and one current white noise error terms, or MA(q)

$$x_t - AR(p) = MA(q) \rightarrow x_t = AR(p) + MA(q)$$

This model is denoted as ARMA(p, q). It is only good for *stationary* time series, i.e. time series with its properties independent of the time at which the time series is observed. Meaning, no matter which **p** and **q** values you pick, the pattern will persist over the whole time series without the distortion from the side of changing mean or variance.



Autoregressive *Integrated* Moving Average

What nonstationarity means mathematically?

It means: there is a *unit root* of the process' characteristic equation (its root is 1).

Autoregressive *Integrated* Moving Average

What nonstationarity means mathematically?
It means: there is a *unit root* of the process' characteristic equation (its root is 1).

The process can be *stationarized* by taking differences, i.e. subtracting the previous value from the next one: $\acute{x}_t = x_t - x_{t-1}$

Autoregressive *Integrated* Moving Average

What nonstationarity means mathematically?
It means: there is a *unit root* of the process' characteristic equation (its root is 1).

The process can be *stationarized* by taking differences, i.e. subtracting the previous value from the next one: $\acute{x}_t = x_t - x_{t-1}$

Taking this into account, for the nonstationary process, ARMA equation can be transformed such that the AR component is the product of the AR(p) and of a unit root of multiplicity d:

Integrated…

drift

$$\left(1 - \sum_{i=1}^{p} \varphi_i B^i\right)(1 - B)^d x_t = \delta + \left(\sum_{i=1}^{q} \theta_i B^i + 1\right)\varepsilon_t$$

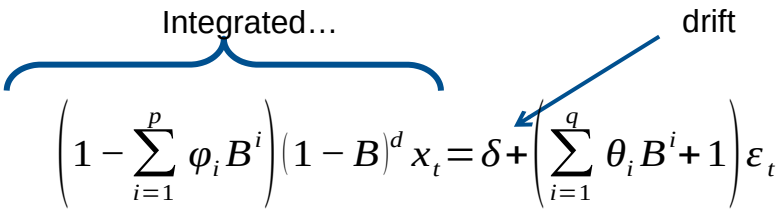
If there is only one such root, and other ones are inside the unit circle, then d = 1. Otherwise, d = number of unit roots. Each unit root is removed through one differencing operation, hence d can also be viewed as the number of differences to be applied.

Autoregressive *Integrated* Moving Average

What nonstationarity means mathematically?
It means: there is a *unit root* of the process' characteristic equation (its root is 1).

The process can be *stationarized* by taking differences, i.e. subtracting the previous value from the next one: .

Taking this into account, for the nonstationary process, ARMA equation can be transformed such that the AR component is the product of the AR(p) and of a unit root of multiplicity d:



If there is only one such root, and other ones are inside the unit circle, then d = 1. Otherwise, d = number of unit roots. Each unit root is removed through one differencing operation, hence d can also be viewed as the number of differences to be applied.

Note: if the model that you use already accepts d as a parameter, you do not need to difference the time series in advance!

Seasonal ARIMA

As you saw, AR, MA, ARMA, and ARIMA with their parameters account only for the *uninterrupted sequence* of the previous values. Hence, seasonality is not considered.

Seasonal ARIMA

As you saw, AR, MA, ARMA, and ARIMA with their parameters account only for the *uninterrupted sequence* of the previous values. Hence, seasonality is not considered.

The adjusted *seasonal ARIMA* model helps to address this by duplicating the same terms but with backshifts *B* of the seasonal period, e.g. if observations are taken each month during several years, then such period is 12; for the quarterly data it is 4. It could also be applied to the smaller periods such as week (7) or a day (24).

Seasonal ARIMA

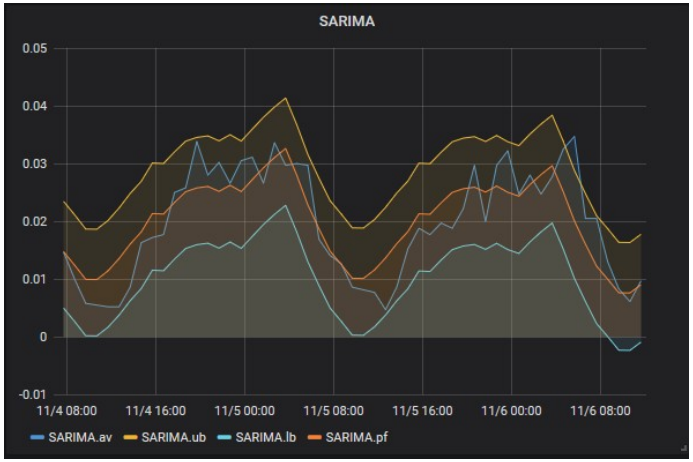
As you saw, AR, MA, ARMA, and ARIMA with their parameters account only for the *uninterrupted sequence* of the previous values. Hence, seasonality is not considered.

The adjusted *seasonal ARIMA* model helps to address this by duplicating the same terms but with backshifts *B* of the seasonal period, e.g. if observations are taken each month during several years, then such period is 12; for the quarterly data it is 4. It could also be applied to the smaller periods such as week (7) or a day (24).

The model is shortly written as:

$$ARIMA(p,d,q) \times (P,D,Q)_S$$

So, we have 7 parameters. How to identify them?



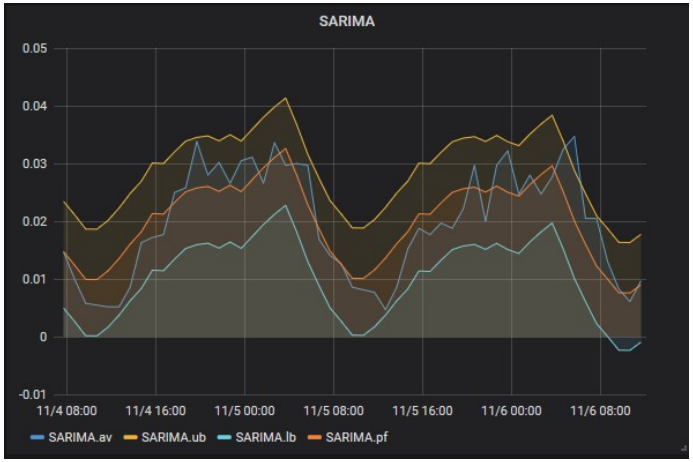
Seasonal ARIMA

As you saw, AR, MA, ARMA, and ARIMA with their parameters account only for the *uninterrupted sequence* of the previous values. Hence, seasonality is not considered.

The adjusted *seasonal ARIMA* model helps to address this by duplicating the same terms but with backshifts *B* of the seasonal period, e.g. if observations are taken each month during several years, then such period is 12; for the quarterly data it is 4. It could also be applied to the smaller periods such as week (7) or a day (24).

The model is shortly written as:

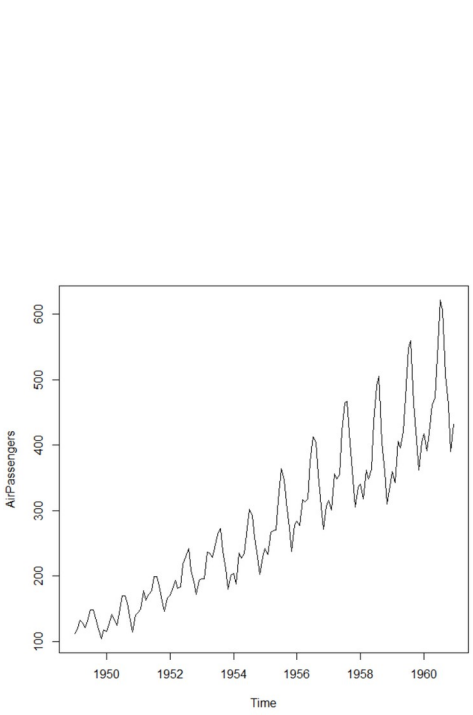
$$ARIMA(p,d,q) \times (P,D,Q)_S$$

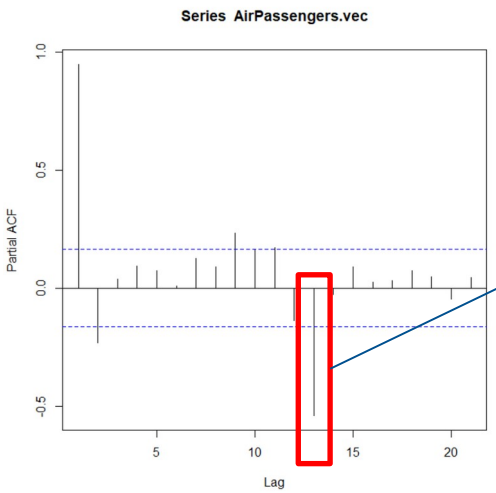


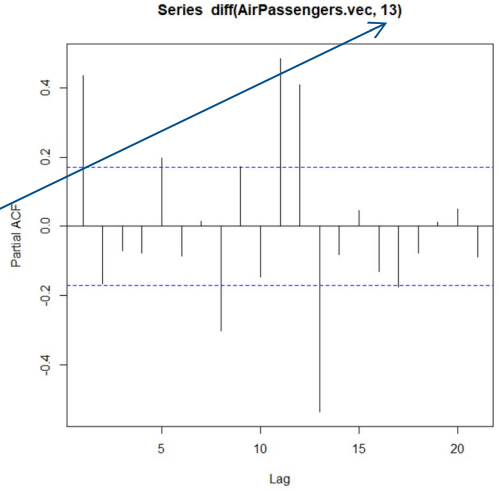
You can do a grid search (most of those are not higher than 2), but also you can analyze.
Read more on seasonal ARIMA.

ARIMA: Intuitions for Modeling

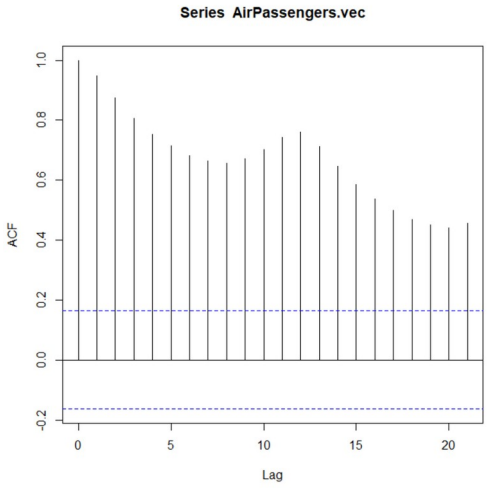
Partial autocorrelation function (PACF) residuals plot points at the value of p for AR(p)

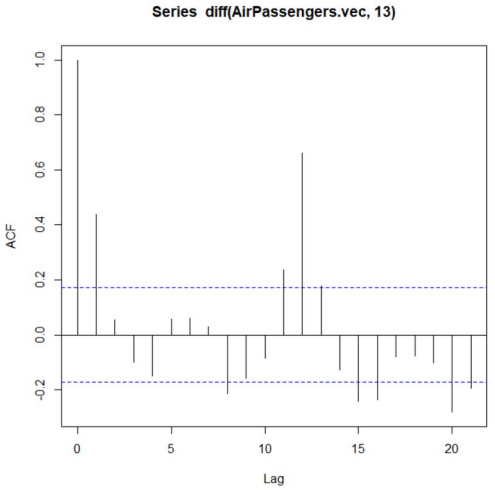






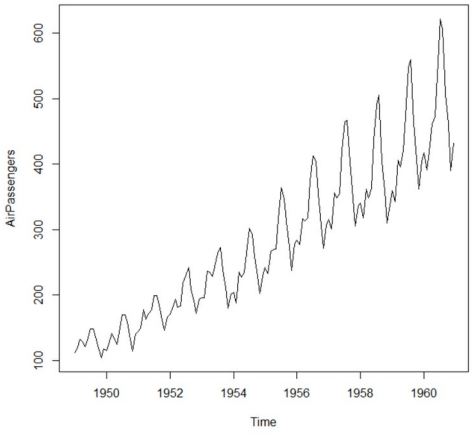
Autocorrelation function (ACF) residuals plot points at the value of q for MA(q)

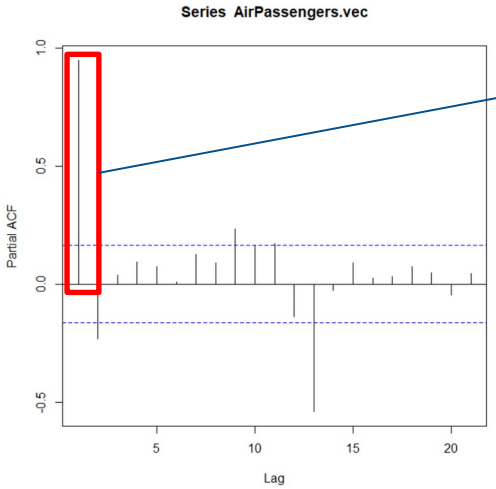


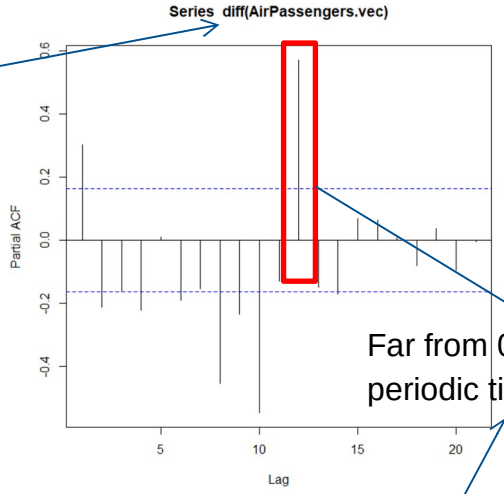


ARIMA: Intuitions for Modeling

Partial autocorrelation function (PACF) residuals plot points at the value of p for AR(p)

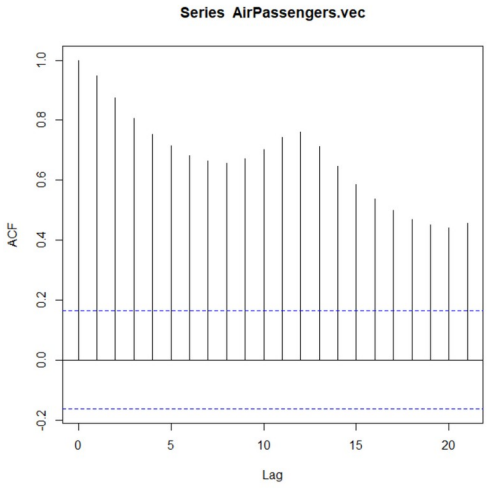


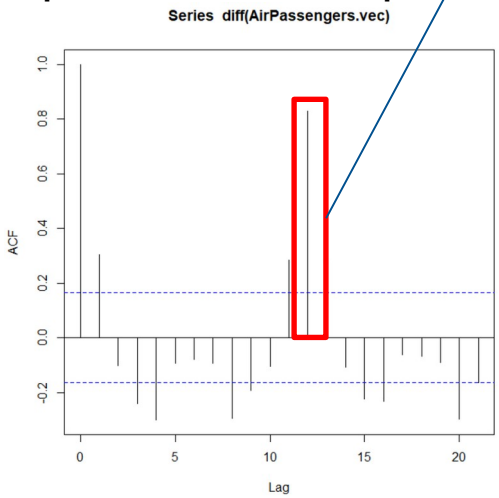




Far from 0, might be periodic time series…

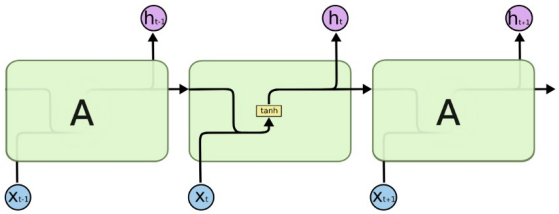
Autocorrelation function (ACF) residuals plot points at the value of q for MA(q)





Long short-term memory Recurrent NeuralNets

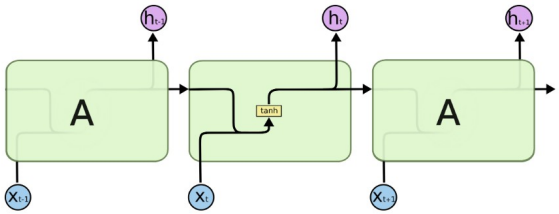
The time series is generally just *a series*, i.e. an ordered *sequence of values*. Recurrent neural networks (RNN) were designed to deal with the ordered data by having a repeated module in which the output of one module is fed to the next one.



[https://colah.github.io/posts/2015-08-Understanding-LSTMs/]

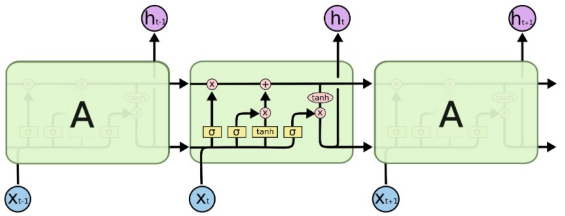
Long short-term memory Recurrent NeuralNets

The time series is generally just *a series*, i.e. an ordered *sequence of values*. Recurrent neural networks (RNN) were designed to deal with the ordered data by having a repeated module in which the output of one module is fed to the next one.



[https://colah.github.io/posts/2015-08-Understanding-LSTMs/]

Their modification, long short-term memory RNN (LSTM RNN), addresses the weak point of RNNs – their inability to deal with really *long-term dependencies* (vanishing gradient problem). It is achieved by the persisting the *modifiable cell state* which runs through all the modules. The sigmoid *forget gate* decides, what part of state to keep, and what part to remove (by multiplying by vals from 0 to 1). The *input gate* decides, what part of state to update. Finally, the updated cell state and the value from the previous module is used to output the value to the next module.



[https://colah.github.io/posts/2015-08-Understanding-LSTMs/]

Other models

Exponential smoothing is a rule of thumb technique for smoothing time series data using the exponential window function [Wikipedia], i.e.:

$$s_0 = x_0$$
$$s_t = \alpha x_t + (1-\alpha)s_{t-1},\ t > 0 \qquad \alpha = 1 - e^{\frac{-\Delta T}{\tau}}$$

The Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) model describes the variance of the error term in the source time series model, therefore this model is useful for time series with changing variance, i.e.:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i a_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 \qquad GARCH(p,q)$$

Support vector regression (SVR) is an extension proposed by H. Drucker et al. to the support vector machines (SVMs) from the domain of the supervised learning, i.e.:

$$f(x,\omega) = \sum_{j=1}^{m} \omega_j g_j(x) + b$$

Singular Spectrum Analysis (SSA) is a non-parametric method of time series decomposition that obtains spectral information on time series. SSA includes: 1) embedding of original time series into the vector space of specific dimension; 2) singular value decomposition resulting in a set of elementary matrices of rank 1; 3) eigentriple grouping to group elementary matrices; 4) diagonal averaging to receive the original time series representation as a sum of reconstructed subseries.

Convolutional Neural Networks (CNN) – time series should be represented as a 1D image.

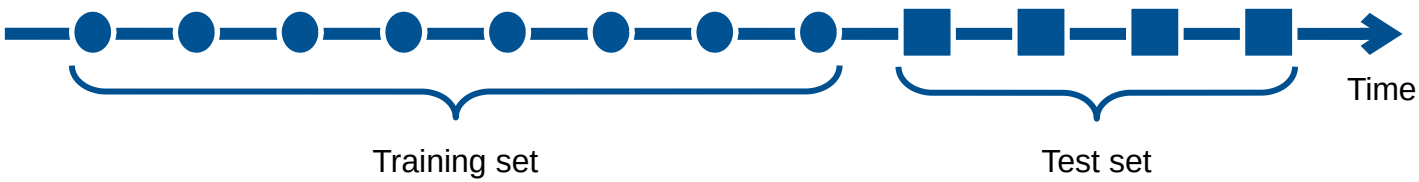
Evaluation of the Forecasting Models

Metrics classes:

- Forecast accuracy metrics
- Resource efficiency metrics → represented via model fitting time / prediction time

Forecast accuracy metrics:

(!) Before we start: time series differ from other datasets – *order of observations matters*!
→ training data should come *before* the test data; they cannot be interleaved or mixed:



Now, metrics.

Forecasting Accuracy Metrics

Forecast error – difference between the actual value at the given time and the forecasted value at the same time: $e_{T+h} = y - \hat{y}$

Metrics:

- Scale-dependent:
  - Mean absolute error (MAE): $MAE = \sum_t |e_t| / n$
  - Root mean squared error (RMSE): $RMSE = \sqrt{\sum_t (e_t)^2 / n}$
- Percentage:
  - Mean absolute percentage error (MAPE): $MAPE = \sum_t |100 \cdot (\frac{e_t}{y_t})| / n$
- Scaled:
  - Mean absolute scaled error (MASE): $MASE = \sum_t \frac{|e_t|}{|ENF_t|} / n$, where ENF is the forecast error for the naïve forecast (the baseline).

All these metrics are computed by the function **accuracy** in the R package forecast.

You can read more in the online book *Forecasting: Principles and Practice* | 5.8

Learning Time Series Forecasting in Practice: ARIMAX case

Data Generator

Data Generator is a simple Python code that implements a model to generate the time series of a count of students in the room with 1 minute interval. Properties of this model are subject to change:

- model covers 4 weeks (**`timestamps_onemin`**)
- nobody is in the room during weekend
- lessons are held in the room from 8:00 (**`starting_hour`**) till 20:00 (**`end_hour`**) without pauses in 2 hour slots (**`lesson_duration_hour`**s)
- students arrive during first 8 minutes of the lesson and leave during last 8 minutes thereof (**`arrival_and_exit_delay_min`**)
- the number of students in the room during the lesson is constant and equals room capacity of 25 people (**`room_capacity`**)
- a small random variation is added to the students' count during arrival and exit

Purpose of Data Generator is to have your own data set to test various forecasting models/techniques.

You can change the model if you see that it does not reflect reality well.

Data Generator: Settable Params

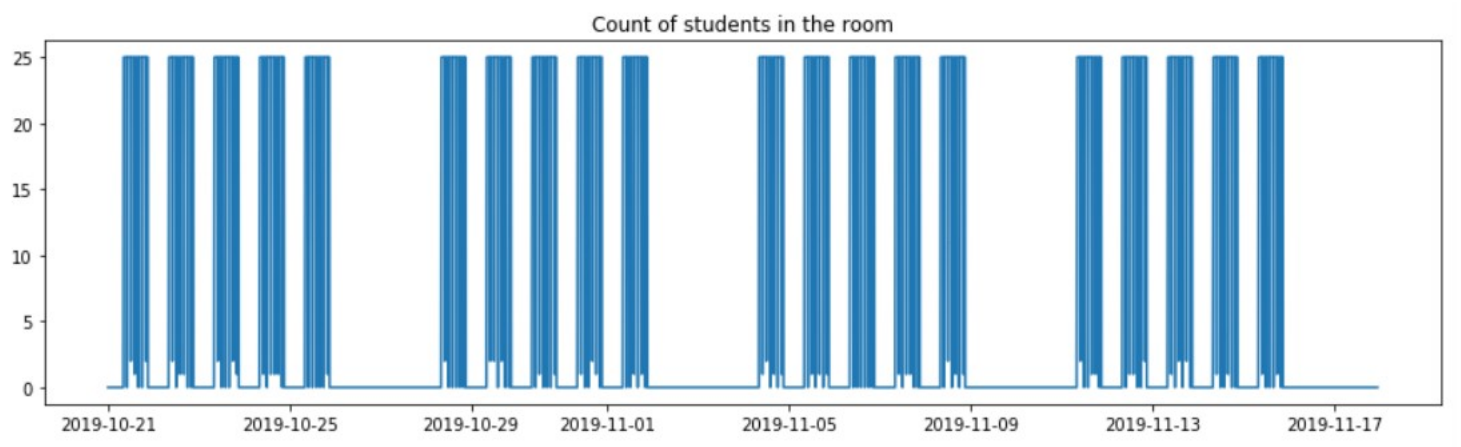
```
# Settable parameters
timestamps_onemin = 4 * 7 * 24 * 60
timestamp_start_s = 1571608800
starting_hour = 8
end_hour = 20
room_capacity = 25
lesson_duration_hours = 2
lesson_duration_min = lesson_duration_hours * 60
arrival_and_exit_delay_min = 8
falloff_border = lesson_duration_min - arrival_and_exit_delay_min
arrival_and_exit_coef_min = room_capacity / arrival_and_exit_delay_min
```

Data Generator: Data Gen Loop

```
# Generating simulated students' count data -
for i in range(timestamps_onemin):
    cur_timestamp_s = timestamp_start_s + i * 60
    cur_date = datetime.fromtimestamp(cur_timestamp_s)
    cur_wd = cur_date.weekday()
    cur_hour = cur_date.hour
    cur_min = cur_date.minute

    cnt_in_room = 0
    if (cur_wd < 5) & (cur_hour >= starting_hour) & (cur_hour <= end_hour):
        lessons_cur_min = (cur_hour * 60 + cur_min) % lesson_duration_min
        cnt_in_room = count_in_room(lessons_cur_min, arrival_and_exit_coef_min, arrival_and_exit_delay_min, room_capacity, falloff_border)

    df_row = pd.DataFrame([[cur_date, cnt_in_room]], columns=['t', 'count'])
    generated_ts = generated_ts.append(df_row)

    # TODO: uncomment to check what is being generated
    # print("%s : %d" % (cur_date , cnt_in_room))
```

Data Generator: Count Modeling

```python
# Function that models count of students in the room based on time of the day
def count_in_room(cur_val, coef, delay_val, capacity, falloff_val):
    cnt = capacity
    lb = -math.floor(0.1 * capacity)
    ub = math.ceil(0.1 * capacity)

    if cur_val <= delay_val:
        cnt = min(max(math.floor(coef * cur_val) + random.randrange(lb, ub, 1), 0), capacity)
    elif cur_val >= falloff_val:
        cnt = max(min(capacity, math.floor(capacity - coef * (cur_val - falloff_val)) + random.randrange(lb, ub, 1)), 0)
    # TODO: you can also model students going in and out of the room during the lesson

    return cnt
```

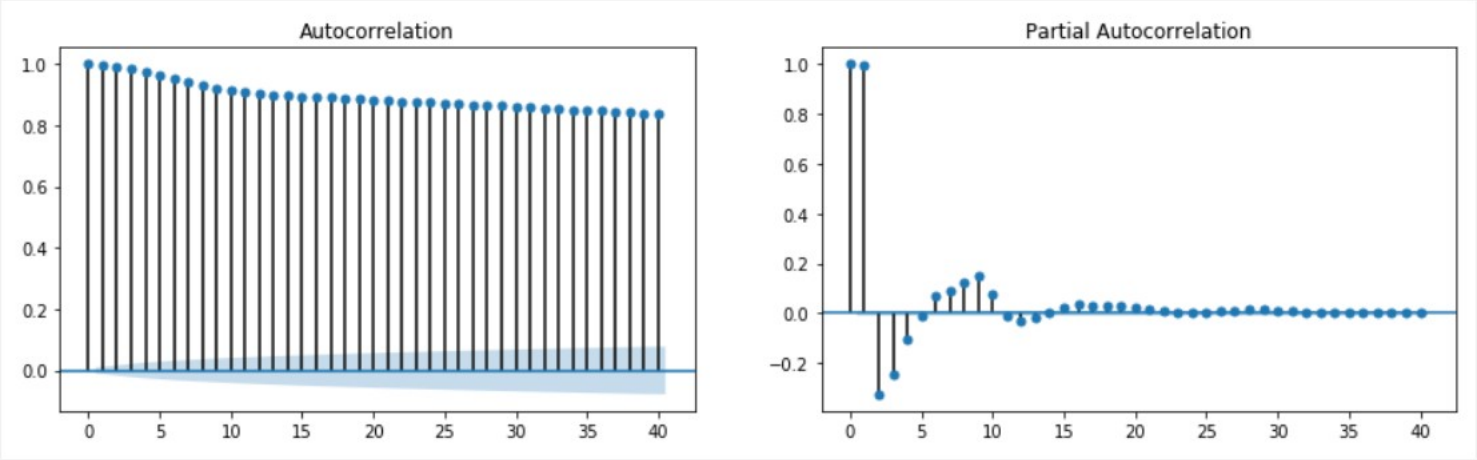


Conventional Time Series Analysis Steps

1. Visualize the data as time series
2. Identify trend, seasonality of the time series, anomalies
3. Classify time series as stationary/non-stationary
4. Preprocess the data by e.g. detrending or differentiating
5. Select the model by considering its properties or try multiple models
6. Analyze the time series data to identify the parameters of the model
7. Fit the model to a training part of the data
8. Predict the result for the testing part of the data
9. Check accuracy and adjust parameters of model if needed

Time Series Analysis in Python: SARIMAX

```
# Visualization and analysis
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 1, figsize=(15,4))

# Visualizing the original data
axes.plot(generated_ts.index._mpl_repr(), generated_ts['count'], '-')
axes.set(title = 'Count of students in the room')

# Visualizing ACF and PACF of time series
fig, axes = plt.subplots(1, 2, figsize=(15,4))
fig = sm.graphics.tsa.plot_acf(generated_ts.iloc[1:]['count'].astype('float'), lags=40, ax=axes[0])
fig = sm.graphics.tsa.plot_pacf(generated_ts.iloc[1:]['count'].astype('float'), lags=40, ax=axes[1])
```



Deriving Model in Python: SARIMAX

```
# TODO: figure out the correct frequency for your data; e.g. 'b' for business days, None by default.
# generated_ts.index.freq = 'b'
# TODO: figure out correct coefficients for your data.
mod = sm.tsa.statespace.SARIMAX(generated_ts['count'].astype(float), trend='c', order=(1,1,(1,0,0,1)))
res = mod.fit(disp=False)
print(res.summary())
```

Statespace Model Results

Dep. Variable:	count	No. Observations:	40320
Model:	SARIMAX(1, 1, (1, 4))	Log Likelihood	-53104.465
Date:	Fri, 22 Nov 2019	AIC	106218.930
Time:	17:01:34	BIC	106261.953
Sample:	0	HQIC	106232.543
	- 40320		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-3.522e-06	0.000	-0.014	0.989	-0.000	0.000
ar.L1	0.9545	0.003	302.026	0.000	0.948	0.961
ma.L1	-0.7662	0.004	-202.786	0.000	-0.774	-0.759
ma.L4	-0.2008	0.002	-127.557	0.000	-0.204	-0.198
sigma2	0.8158	0.001	1169.859	0.000	0.814	0.817

Ljung-Box (Q):	3346.03	Jarque-Bera (JB):	154274240.56
Prob(Q):	0.00	Prob(JB):	0.00
Heteroskedasticity (H):	0.83	Skew:	-10.24
Prob(H) (two-sided):	0.00	Kurtosis:	305.35

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Assignment 5

Assignment

1. **On the generated data:**
   - Adjust data generator to your needs
   - Try at least three different forecasting techniques (e.g. LR, ARIMA, LSTM)
   - Write a report explaining your results and how you found the best parameters (max 4 pages).