HarvardX-PH125.9x: Choose Your Own Project Report

Time series analysis using the example of weather data

Regina Castra

2019-06-07

Contents

1 Introductory Summary

The programming language **R** is considered particularly suitable for the analysis of time series, as there are several packages and methods available to perform forecasts. For this project, the data of a weather station in Jena will be used to provide training and test dataset.

The goal is to train a machine learning algorithm using the daily mean temperatures of four years to predict the temperatures of the fifth year. The quality of fitting the signal of the training dataset as well as the forecast's quality of the fifth year is quantified by a selection of accuracy measurements like RMSE, MAE, MASE and MAPE.

Starting with a base model which uses just the average as a constant forecast value, several different models are used. A neural net method as well as models taking advantage of the seasonality of the data perform quite well in forecasting the temperatures.

2 Analysis

2.1 Data Source and Description

The used data file jena_climate_2009_2016.csv.zip is described on the website hosted at kaggle and can be downloaded from the Amazon Simple Storage Service (Amazon S3) without the need of any login credentials. According to the website, there were 14 different quantities (such as air temperature, atmospheric pressure, humidity, wind direction, and so on) recorded every 10 minutes, over several years. This dataset is limited to the years from 2009 to 2016.

The zip file provides only one csv-file:

Name	Length	Date
jena_climate_2009_2016.csv	43164220	2017-03-19 15:42:00

2.2 Data Structure

File *jena_climate_2009_2016.csv* has **15 attributes** stored in the columns and consist of **420551 observations** in the rows. The data seems to be in tidy format:

- Each variable in the dataset is placed in its own column
- Each observation is placed in its own row
- Each value is placed in its own cell

The 15 attributes have the following names, types and short descriptions (taken from the website hosted at kaggle):

Attribute	Type	Description
Date Time	character	date & time
p (mbar)	double	atmospheric pressure
T (degC)	double	temperature
Tpot (K)	double	potential temperature
Tdew (degC)	double	dew point temperature
rh (%)	double	relative humidity
VPmax (mbar)	double	saturation water vapor pressure
VPact (mbar)	double	actual water vapor pressure
VPdef (mbar)	double	water vapor pressure deficit
sh (g/kg)	double	specific humidity
H2OC (mmol/mol)	double	water vapor concentration
rho (g/m**3)	double	air density
wv (m/s)	double	wind velocity
max. wv (m/s)	double	maximum wind velocity
wd (deg)	double	wind direction

To get a first impression of the data, the first 4 observations are shown in the following table. Please note that the table has been transposed for better readability:

Attribute	Observation 1	Observation 2	Observation 3	Observation 4
Date Time	01.01.2009 00:10:00	01.01.2009 00:20:00	01.01.2009 00:30:00	01.01.2009 00:40:00
p (mbar)	996.52	996.57	996.53	996.51
T (degC)	-8.02	-8.41	-8.51	-8.31
Tpot (K)	265.40	265.01	264.91	265.12
Tdew (degC)	-8.90	-9.28	-9.31	-9.07
rh (%)	93.3	93.4	93.9	94.2
VPmax (mbar)	3.33	3.23	3.21	3.26
VPact (mbar)	3.11	3.02	3.01	3.07
VPdef (mbar)	0.22	0.21	0.20	0.19
sh (g/kg)	1.94	1.89	1.88	1.92
H2OC (mmol/mol)	3.12	3.03	3.02	3.08
rho (g/m**3)	1307.75	1309.80	1310.24	1309.19
wv (m/s)	1.03	0.72	0.19	0.34
max. wv (m/s)	1.75	1.50	0.63	0.50
wd (deg)	152.3	136.1	171.6	198.0

2.3 Data Check and Transformation

The dataset does not contain any *missing values (NA)*. The number of *NA*-entries for each attribute in the dataset is:

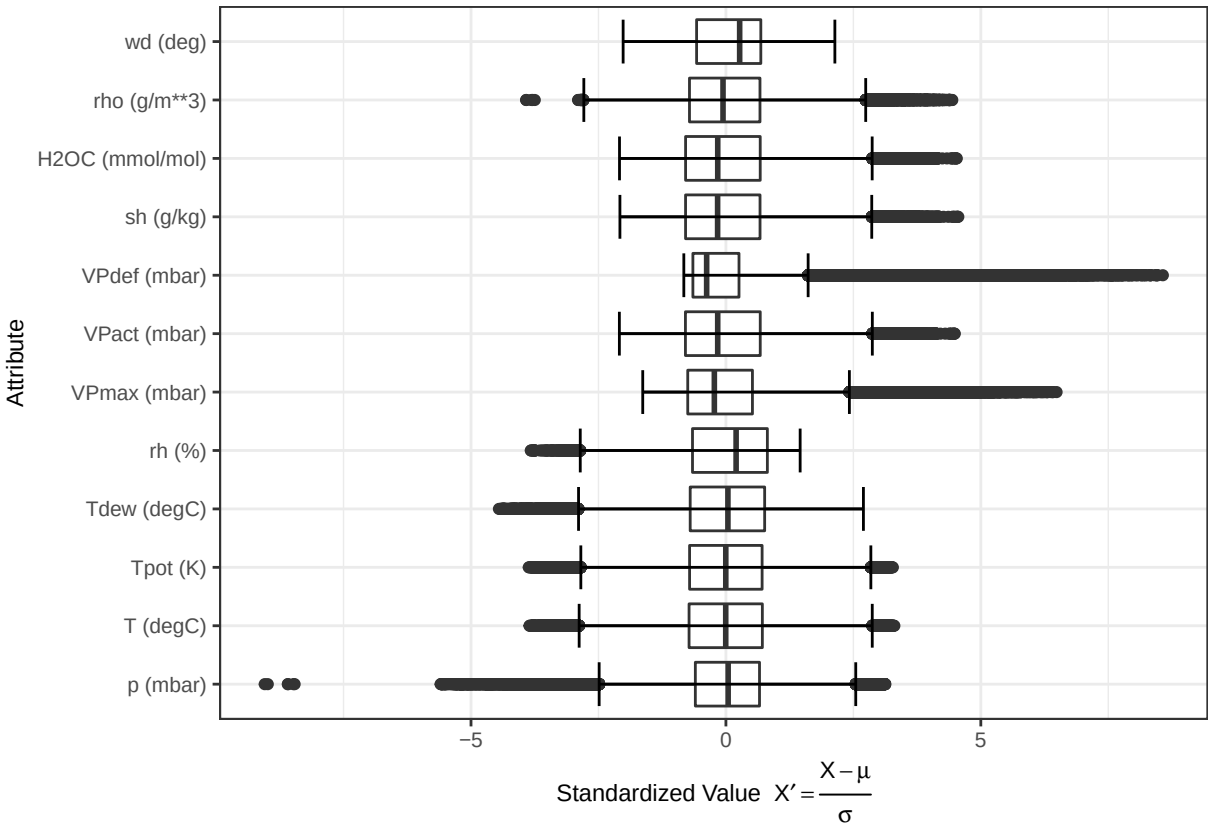
	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
Number of NA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The statistical properties give a first impression of the data quality:

Attribute	Statistical Properties					
Date Time	Length:420551	Class :character	Mode :character	NA	NA	NA
p (mbar)	Min. : 913.6	1st Qu.: 984.2	Median : 989.6	Mean : 989.2	3rd Qu.: 994.7	Max. :1015.4
T (degC)	Min. :-23.01	1st Qu.: 3.36	Median : 9.42	Mean : 9.45	3rd Qu.: 15.47	Max. : 37.28
Tpot (K)	Min. :250.6	1st Qu.:277.4	Median :283.5	Mean :283.5	3rd Qu.:289.5	Max. :311.3
Tdew (degC)	Min. :-25.010	1st Qu.: 0.240	Median : 5.220	Mean : 4.956	3rd Qu.: 10.070	Max. : 23.110
rh (%)	Min. : 12.95	1st Qu.: 65.21	Median : 79.30	Mean : 76.01	3rd Qu.: 89.40	Max. :100.00
VPmax (mbar)	Min. : 0.95	1st Qu.: 7.78	Median :11.82	Mean :13.58	3rd Qu.:17.60	Max. :63.77
VPact (mbar)	Min. : 0.790	1st Qu.: 6.210	Median : 8.860	Mean : 9.534	3rd Qu.:12.350	Max. :28.320
VPdef (mbar)	Min. : 0.000	1st Qu.: 0.870	Median : 2.190	Mean : 4.042	3rd Qu.: 5.300	Max. :46.010
sh (g/kg)	Min. : 0.500	1st Qu.: 3.920	Median : 5.590	Mean : 6.022	3rd Qu.: 7.800	Max. :18.130
H2OC (mmol/mol)	Min. : 0.80	1st Qu.: 6.29	Median : 8.96	Mean : 9.64	3rd Qu.:12.49	Max. :28.82
rho (g/m**3)	Min. :1059	1st Qu.:1187	Median :1214	Mean :1216	3rd Qu.:1243	Max. :1394
wv (m/s)	Min. :-9999.000	1st Qu.: 0.990	Median : 1.760	Mean : 1.702	3rd Qu.: 2.860	Max. : 28.490
max. wv (m/s)	Min. :-9999.000	1st Qu.: 1.760	Median : 2.960	Mean : 3.057	3rd Qu.: 4.740	Max. : 23.500
wd (deg)	Min. : 0.0	1st Qu.:124.9	Median :198.1	Mean :174.7	3rd Qu.:234.1	Max. :360.0

Comparing the "*Min.*" and "*Max.*" values with the quantiles "*1st Qu.*" and "*3rd Qu.*" lead to the conclusion that the values of the attributes "*wv (m/s)*" and "*max. wv (m/s)*" show outliers.

For the other attributes, boxplots of their standardized values will give a graphical view of their distribution:



Attributes *"rho (g/m**3)"* and "*p (mbar)*" show separated *outliers*. The remaining attributes seem to have plausible values.

The first column "*Date Time*" is of type *character* and needs to be converted into a *POSIXct* "date time" column. This is done by using the function *strptime()* with the format "%d.%m.%Y %H:%M:%S".

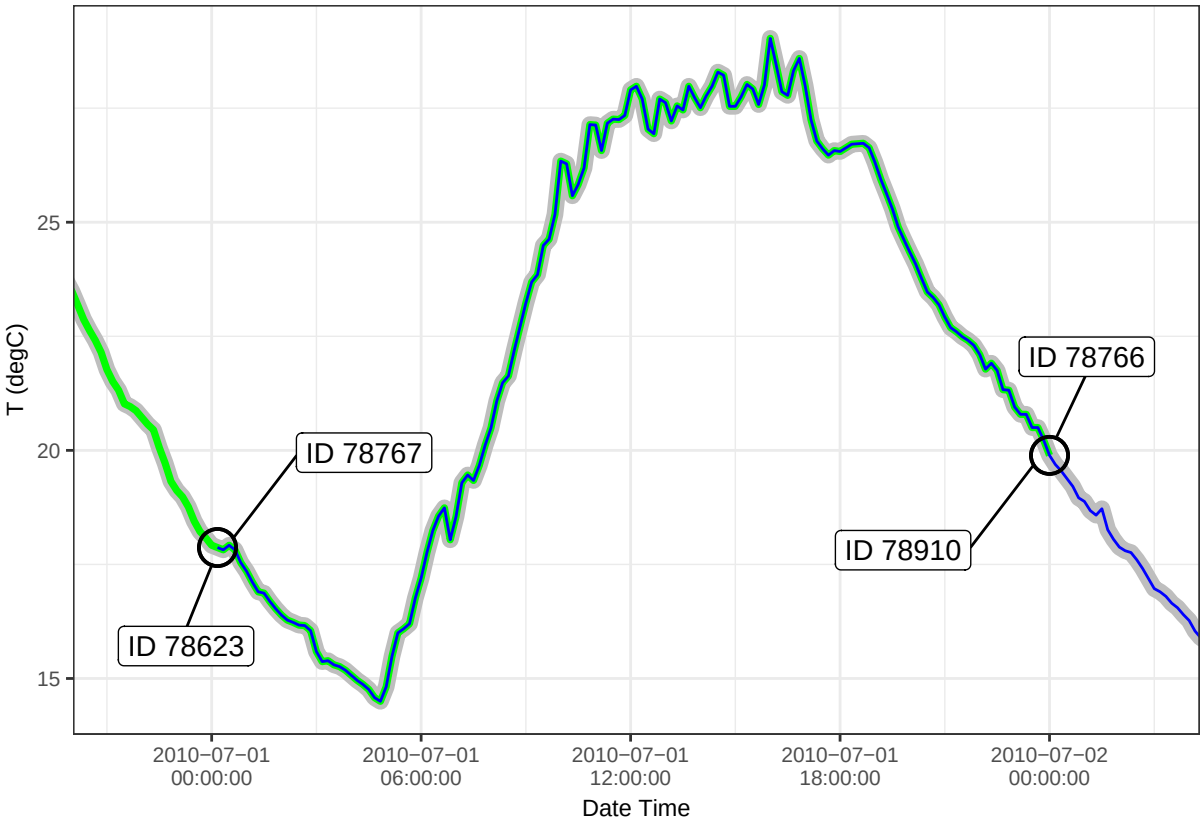
For analysis of time series it is mandatory to use data of constant lags. According to the data description there should be an observation every 10 minutes. A check confirms that the data has a periodicity of 10 minutes:

```
## 10 minute periodicity from 2009-01-01 00:10:00 to 2017-01-01
```

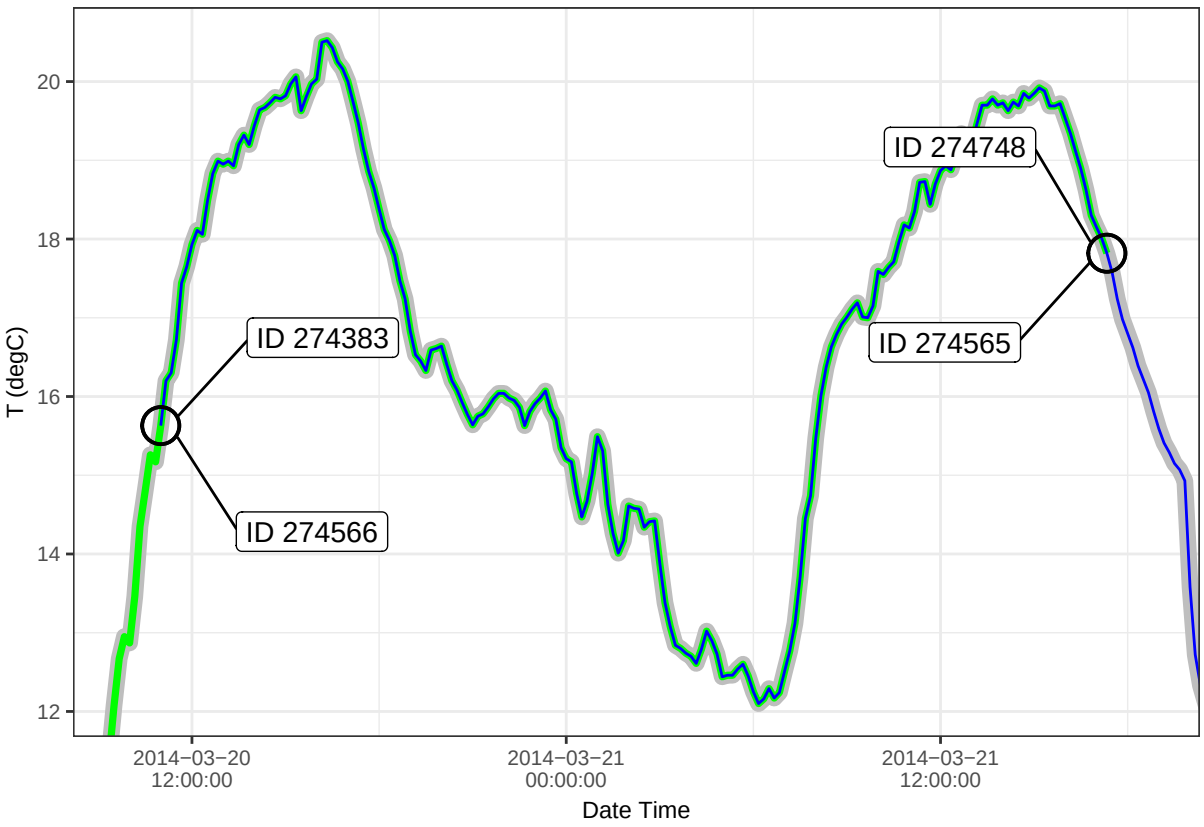
But there are also some observations missing (positive lag values > 10min) and others seem to be redundant (negative lag values). For easier handling an unique *ID* is introduced for each observation. The *ID* is equivalent to the line number in the *raw*-data.

ID	Date Time	lag (min)
40379	08.10.2009 10:10:00	30
78767	01.07.2010 00:10:00	-1430
230020	16.05.2013 09:10:00	20
274566	20.03.2014 11:00:00	-1820
293557	30.07.2014 08:20:00	20
301674	25.09.2014 09:00:00	960
411268	28.10.2016 12:50:00	4460

The first range of redundant values is shown in the following figure. The resulting "cleaned" data is plotted as grey line. The green line shows the observations up to *ID_78766* at time stamp *2010-07-02 00:00:00*. The next observation *ID_78767* "jumps back" to time stamp *2010-07-01 00:10:00*. This observation and subsequent ones are plotted with a blue line.



The figure shows that the values are truely redundant, i.e. at each time stamp the green and the blue lines show the same value. Same is true for the second range of redundant values:



After dropping the redundant values, the “cleaned” data looks at its joints as follows:

ID	Date Time	lag (min)
78765	01.07.2010 23:50:00	10
78766	02.07.2010 00:00:00	10
78911	02.07.2010 00:10:00	10
78912	02.07.2010 00:20:00	10

ID	Date Time	lag (min)
274564	21.03.2014 17:10:00	10
274565	21.03.2014 17:20:00	10
274749	21.03.2014 17:30:00	10
274750	21.03.2014 17:40:00	10

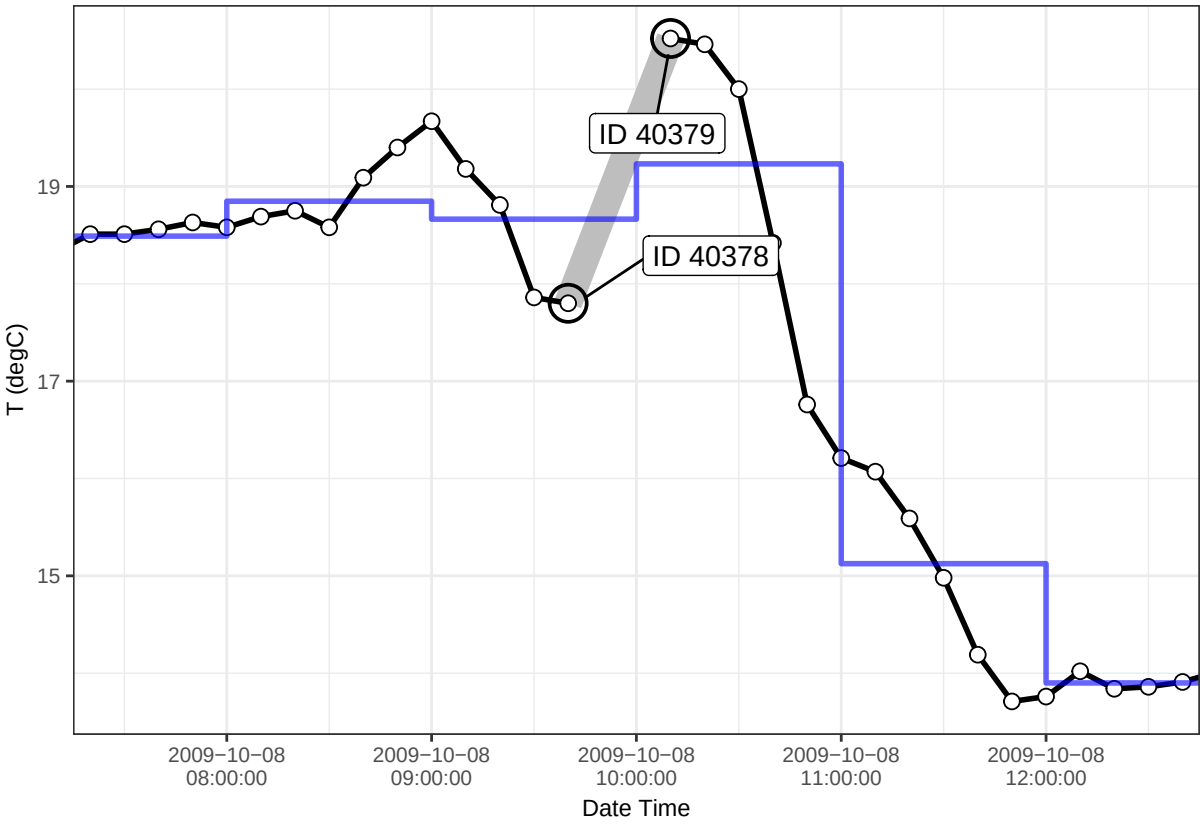
The negative lags have been removed successfully from the data, but positive lags still remain:

ID	Date Time	lag (min)
40379	08.10.2009 10:10:00	30
230020	16.05.2013 09:10:00	20
293557	30.07.2014 08:20:00	20
301674	25.09.2014 09:00:00	960
411268	28.10.2016 12:50:00	4460

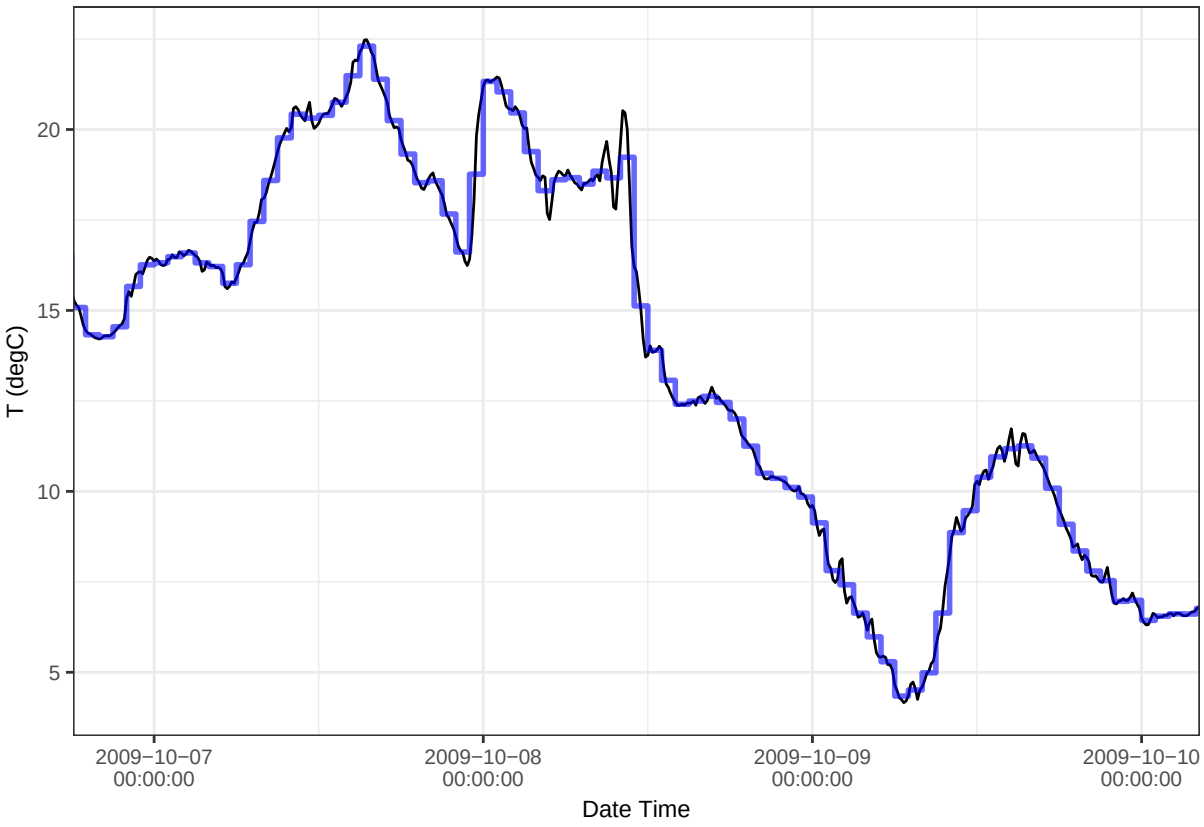
These gaps usually need to be filled, as most modeling techniques require regular time series, i.e. they need a specific constant interval between observations. The data will be aggregated using a time window of one hour, here by calculating a mean value for each hour. Using this approach, the three gaps which are smaller than 60mins can be “closed”. Unfortunately two gaps over several hours still persist:

IDh	Date Time	lag (h)
50227	25.09.2014 09:00:00	16
68493	28.10.2016 12:00:00	74

The small gaps of 20 and 30 minutes disappeared. The following figure gives an impression, how this was accomplished. The raw-data (black line) is used to create a mean value for each hour (blue line). There are two observations missing between *ID_40378* and *ID_40379* (grey band). By averaging over 1h intervalls the gap is automatically “closed”. Remark: Attribute *IDh* is introduced which provides an unique integer for each hourly observation.



Following plot illustrates the aggregation approach and shows how well the 1h-average follows the original data. It can be thought of as a noise-reduction, i.e. higher frequencies are damped.



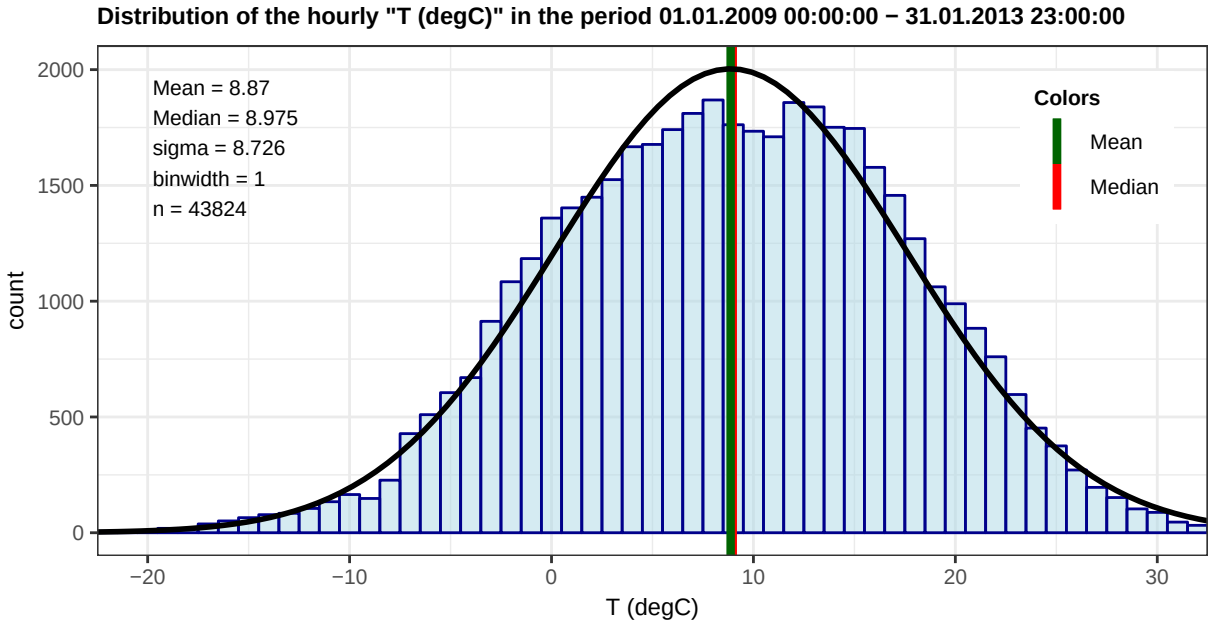
The resulting data used in the following chapters starts at “01.01.2009_00:00:00” and will be cut off at “24.09.2014_00:00:00”.

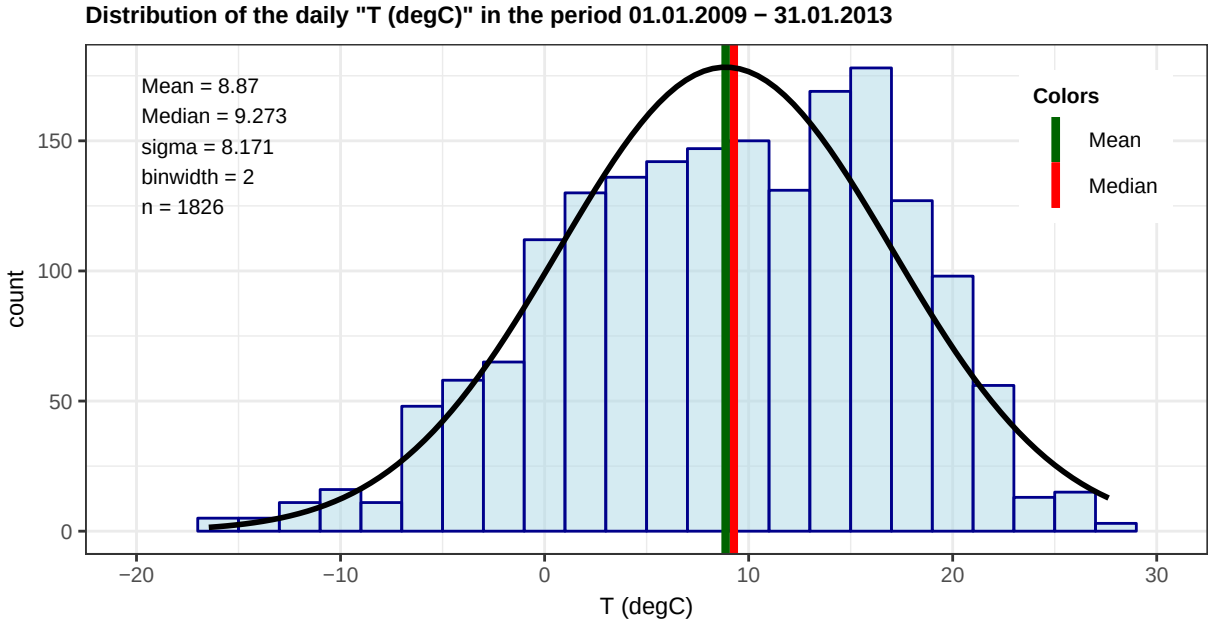
2.4 Data Analysis

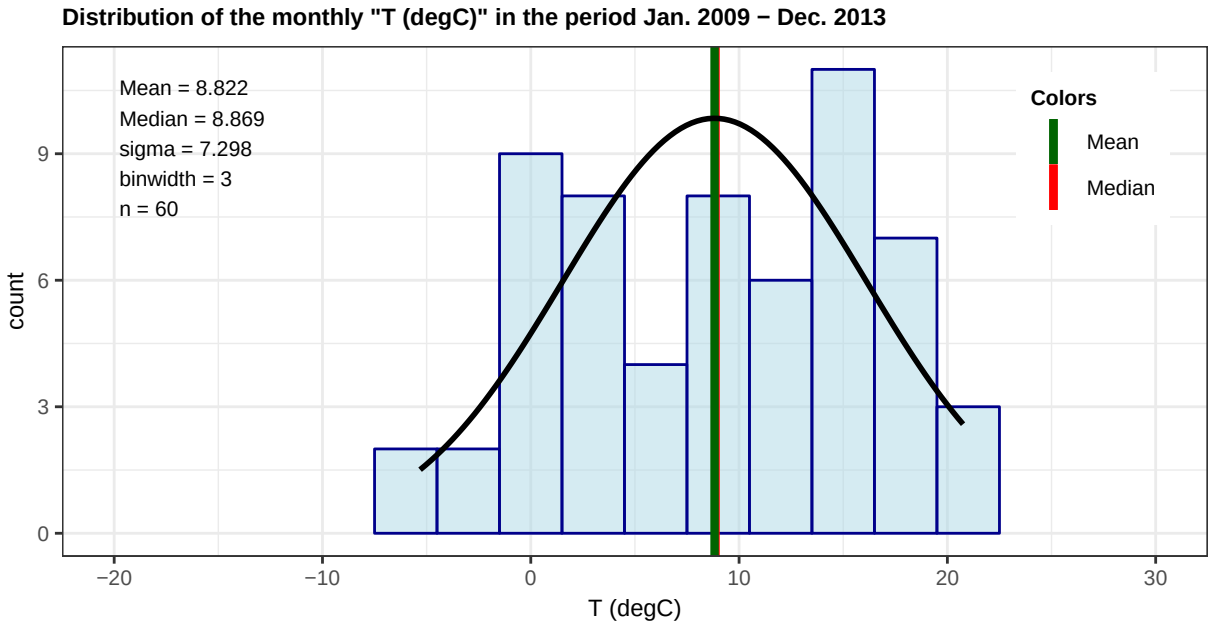
The hourly data is clean for the time period “01.01.2009_00:00:00” to “24.09.2014_00:00:00”. In the following the standard pre-processing and analysis for regular time series will be performed mainly on the attribute “*T (degC)*”, i.e. the focus will be on univariate regular time series modelling.

During the first investigations it became obvious that working on hourly data requires a lot of computing resources. Therefore an aggregation is performed again to obtain daily data, i.e. for the given time period using the hourly data, the mean temperature “*T (degC)*” for each day is calculated.

Furthermore the time period of the investigated data is reduced to five years (01.01.2009_00:00-31.01.2013_23:00) to obtain a dataset with frequency of 365 days (whole year). For the hourly, daily and monthly aggregated data the histograms are presented:



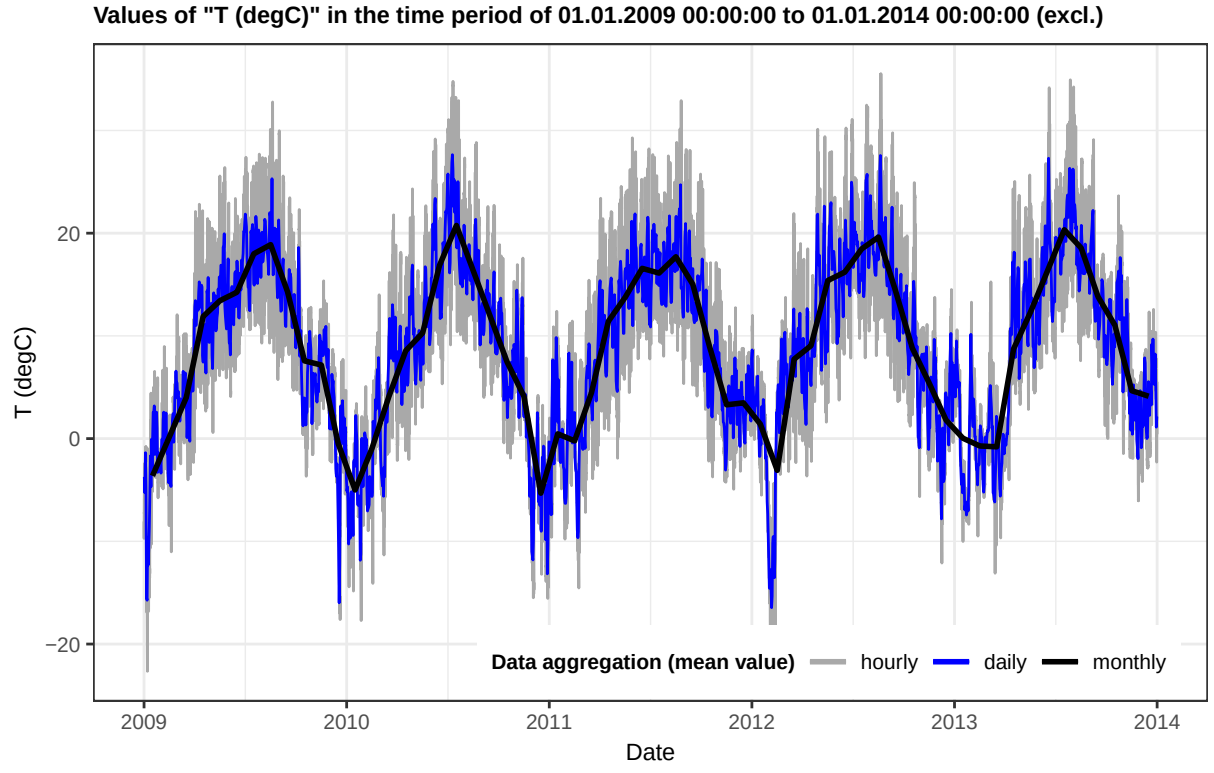




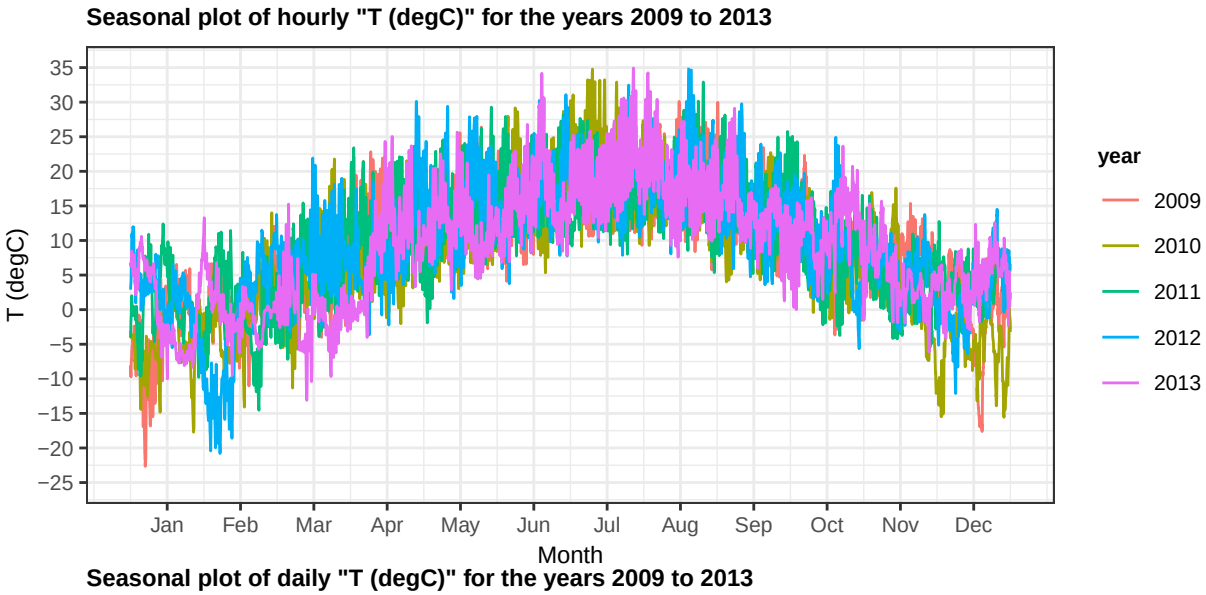
It turns out, that calculating the mean has a huge influence on the distribution: By averaging over a longer period of time the distribution deviates more and more from normal distribution. Normality test methods like the Shapiro-Wilk test and the Anderson-Darling test applied to the hourly, daily and monthly aggregated data also support this theses by returning higher *p-values* for increasing deviation:

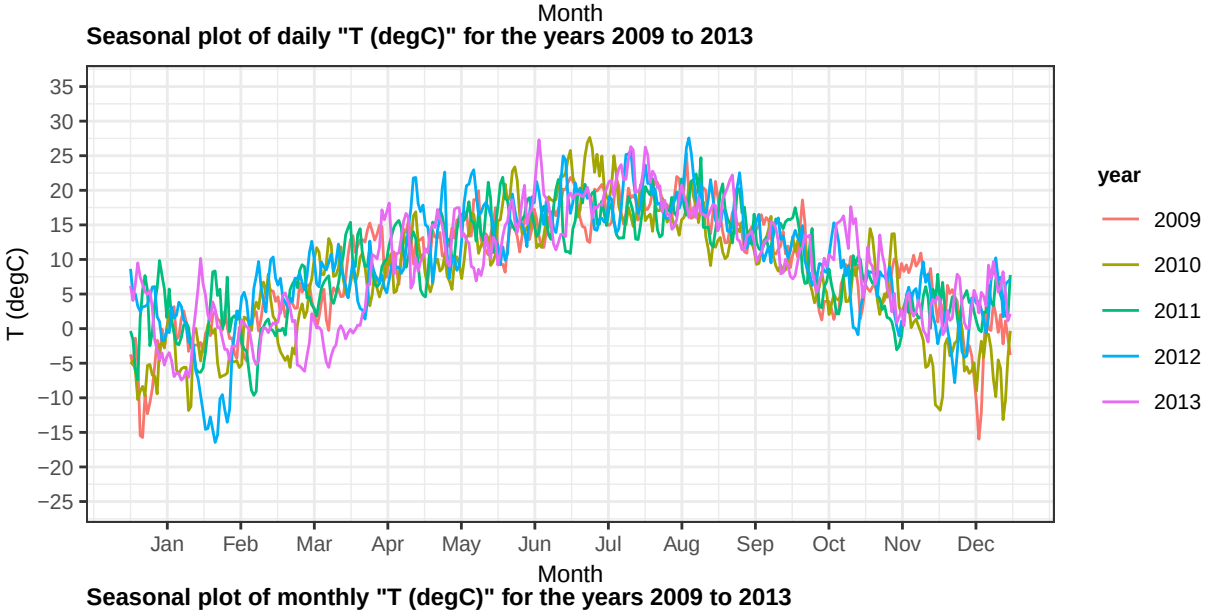
Test on data	hourly data	daily data	monthly data
Shapiro-Wilk normality test, p-value	N/A	2.6e-12	0.0287
Anderson-Darling normality test, p-value	3.7e-24	1.16e-17	0.0378

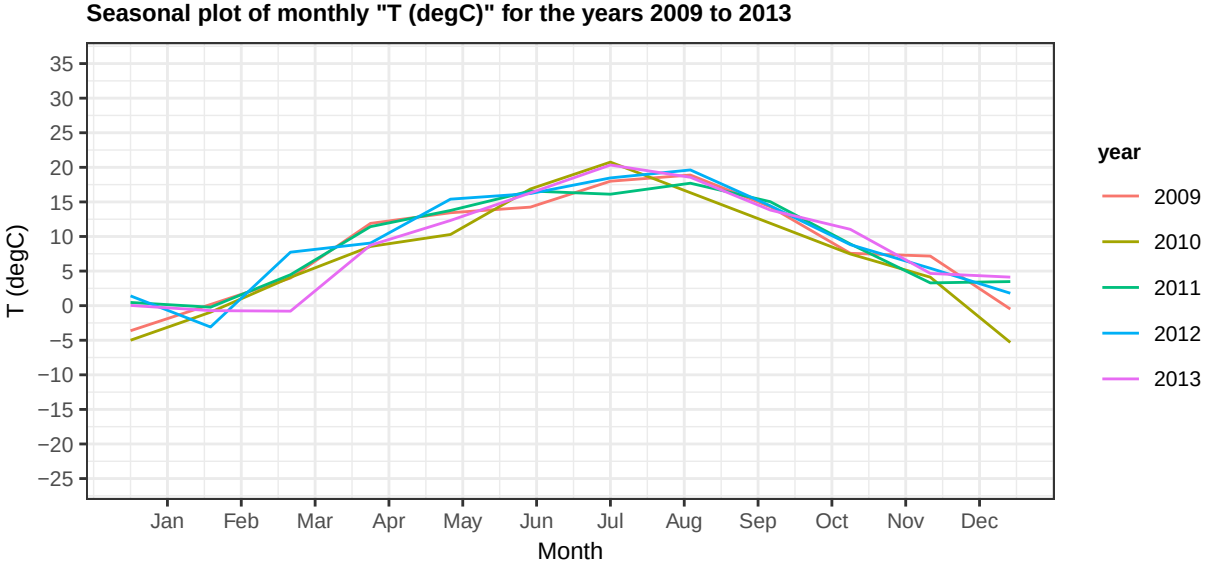
Aggregating reduces the maximum values (spikes) like a filter, which is well shown by line plots of “T (degC)”:



The plot shows data of high seasonality with a yearly frequency, i.e. every 12 months the signal seems to repeat itself. Seasonal plots (see function seasonplot()) are able to visualize this behaviour by plotting the data against the seasons (January to December) for separate years, see also Hyndman (2014, chapter 2):

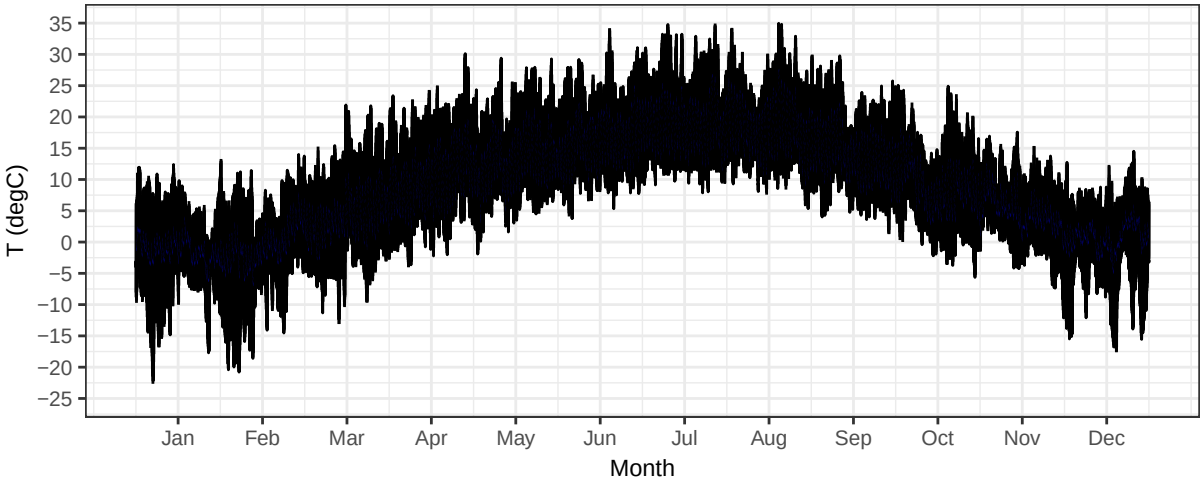




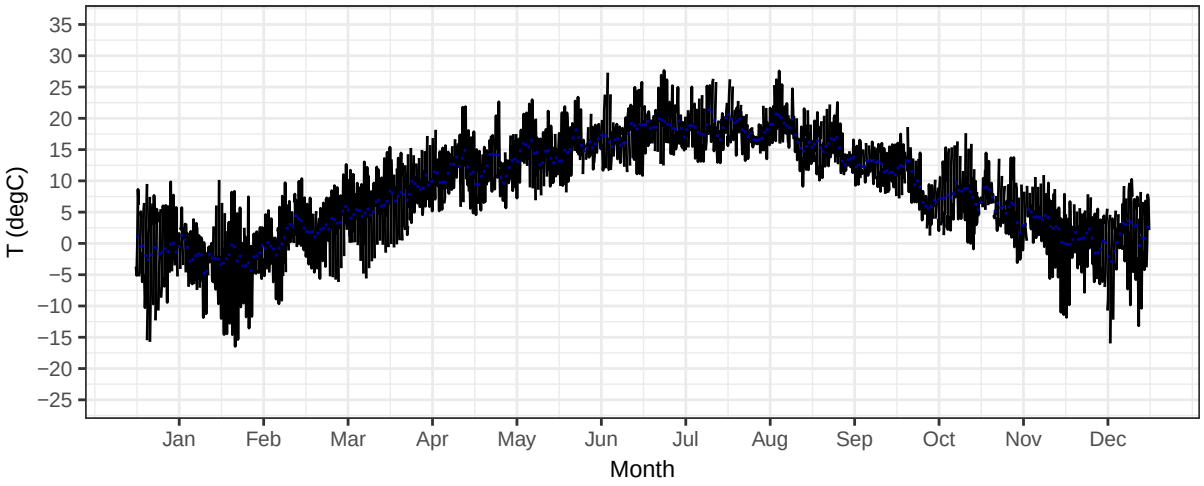


To get a better impression, how much the value of “T (degC)” varies from year to year for each hour/day/month, so called “month plots” (see function monthplot()) are created. For each hour/day/month of the five given years, the value is plotted side by side together with its average as blue line.

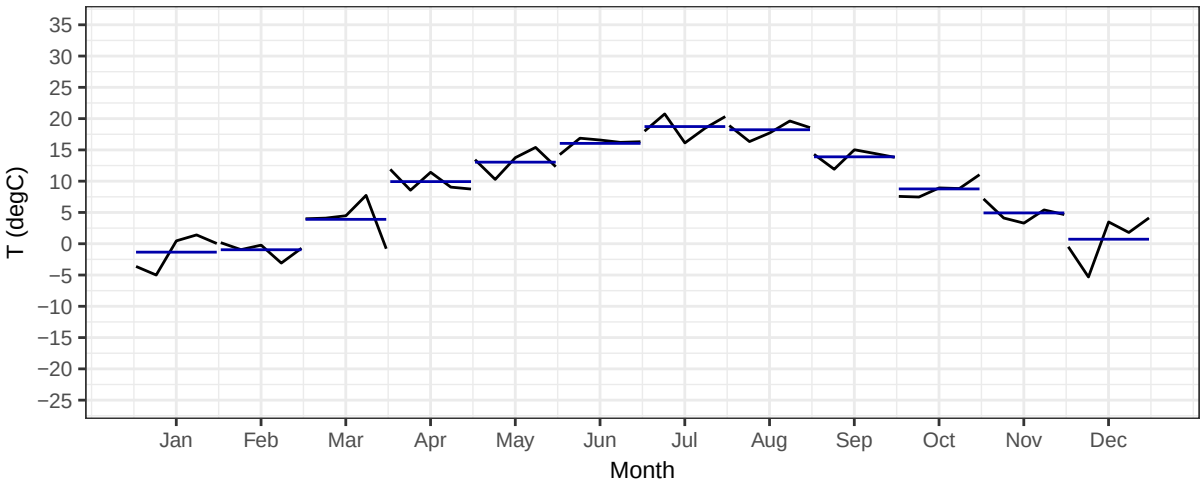
"Month" (=hour) plot of hourly "T (degC)" for the years 2009 to 2013



"Month" (=day) plot of daily "T (degC)" for the years 2009 to 2013

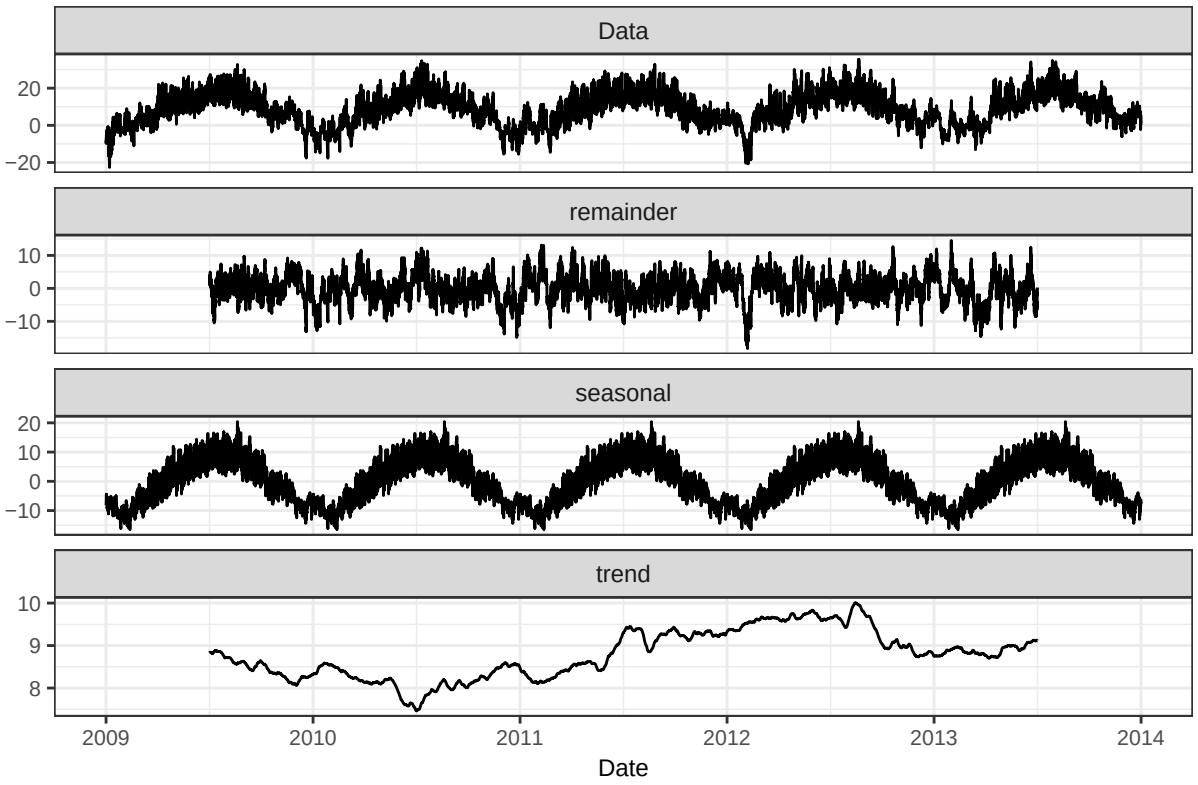


Month plot of monthly "T (degC)" for the years 2009 to 2013

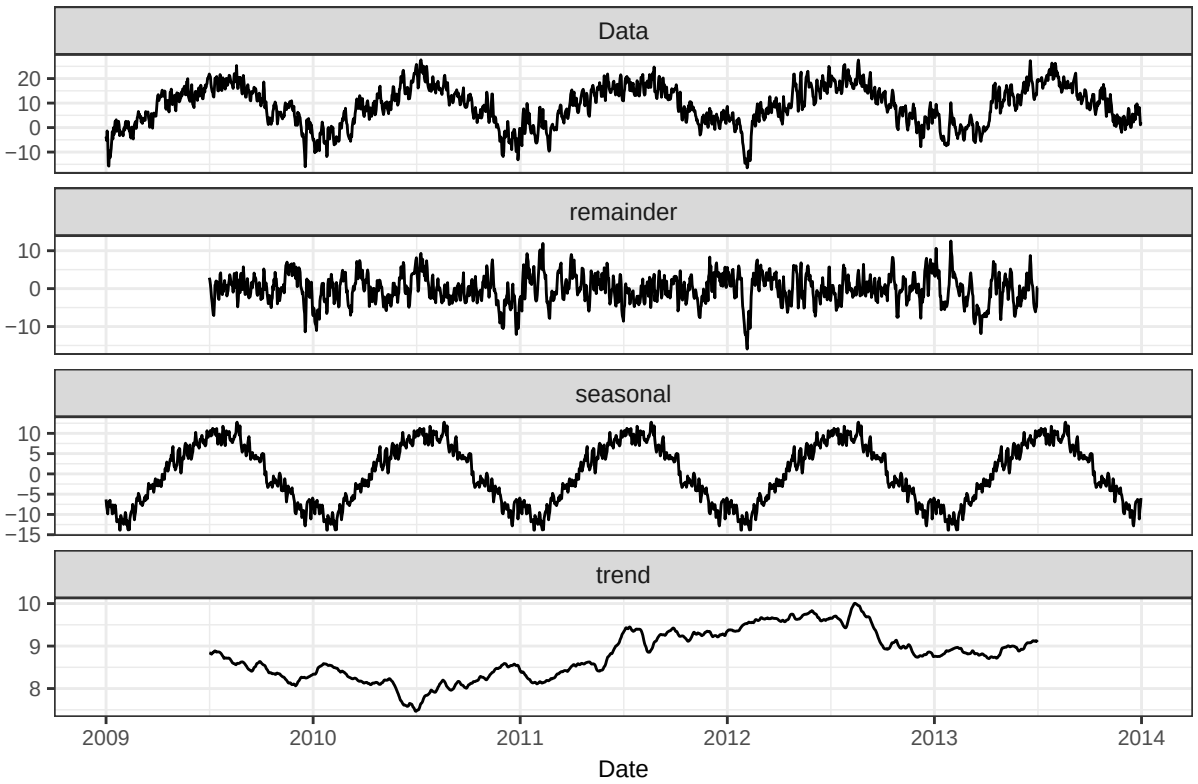


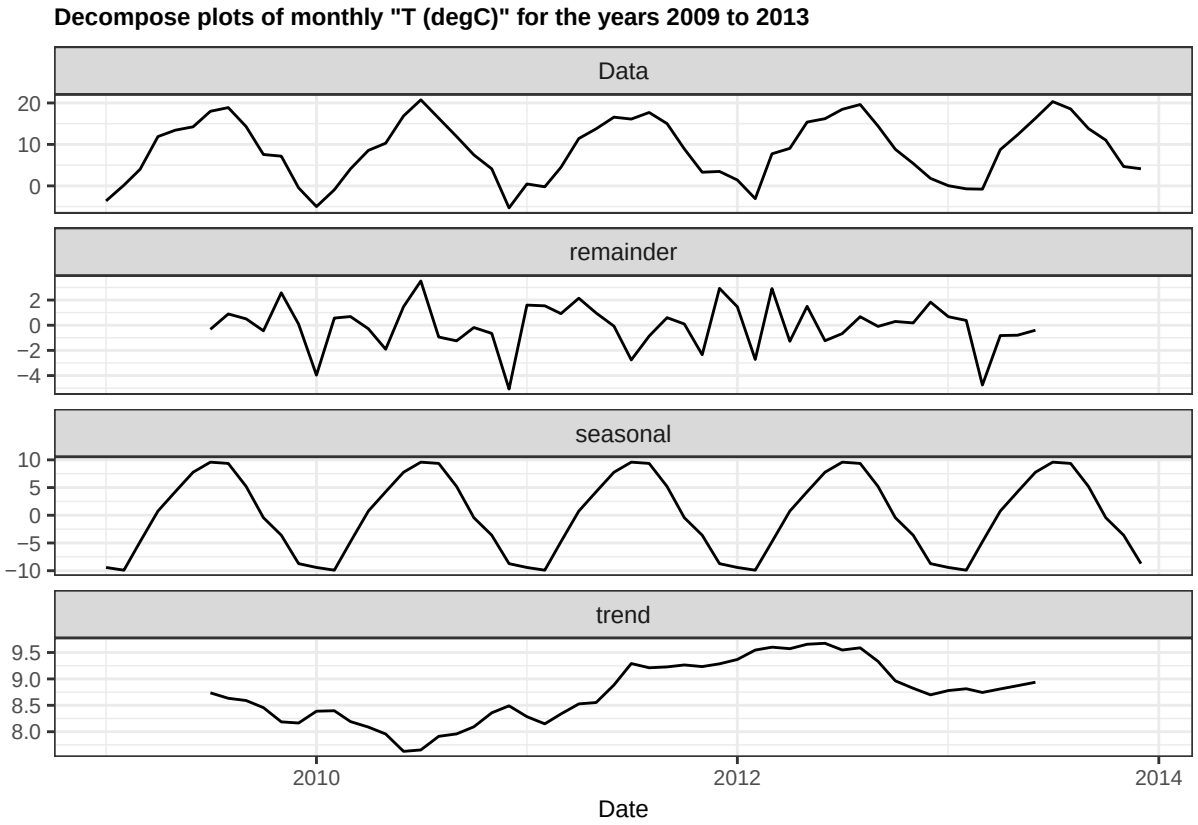
The plots confirm data of high seasonality. This characteristic can be used for forecasting. Most forecasting methods depend on data without a trend present. By decomposing the data regarding season and trend, the remainder does not show any obvious pattern:

Decompose plots of hourly "T (degC)" for the years 2009 to 2013



Decompose plots of daily "T (degC)" for the years 2009 to 2013





The Augmented Dickey–Fuller test delivers *p-values* smaller than 0.015, which is also an indication for stationary data, i.e. data without a trend. According to Tavish Srivastava's Time Series Tutorial stationary data must fulfill the following:

1. The mean of the series should be constant over time.
2. The variance of the series should be constant over time.
3. The covariance of the series should be constant over time.

Summary of data analysis

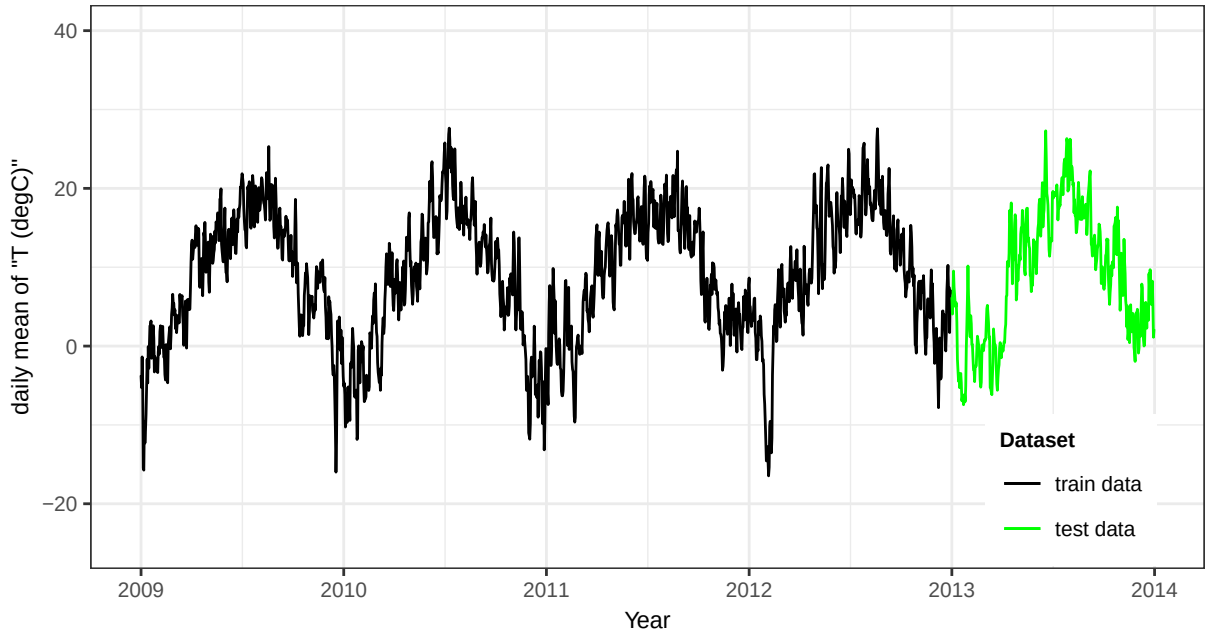
By looking at the line plots of the data and its decomposition it can be summarized:
The data is of high seasonality, is stationary and shows no distinct trend. The remainder has no obvious pattern.

3 Modeling

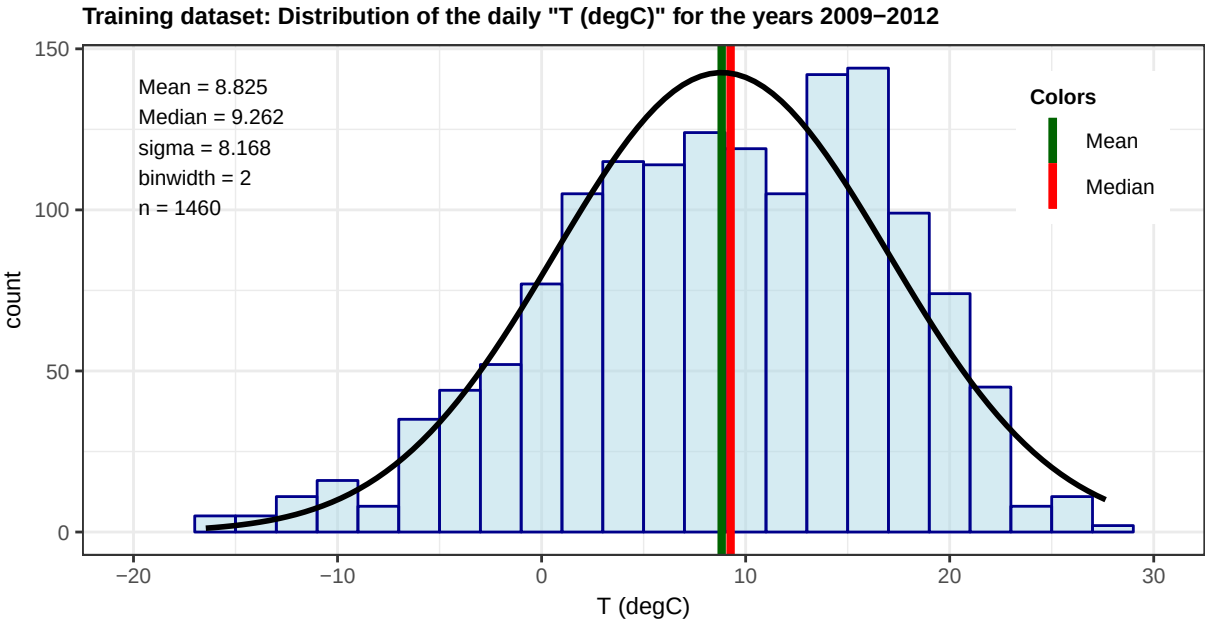
In this chapter training and test dataset will be defined. After introducing the methods to quantify the accuracy, several models for forecasting will be presented.

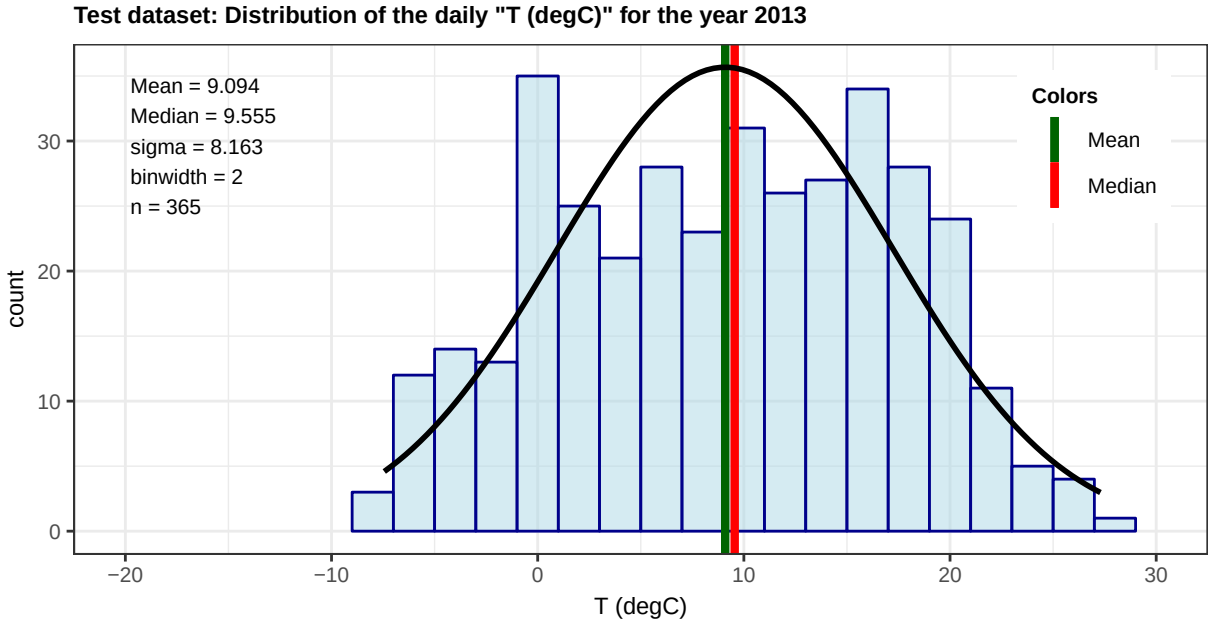
3.1 Train and Test Dataset

In order to evaluate the quality of the forecast, the daily data of the 5-year period is split into two datasets: The time frame 01.01.2009-31.12.2012 will be used as training dataset. The values of the fifth year (2013) are sepcified as test dataset (holdout) and will be compared to the forecasted values. The following line plot shows the two datasets:



A first comparison of the two datasets shows very different distributions to each other. Both datasets deviate from normality distribution:





3.2 Quantification of Accuracy

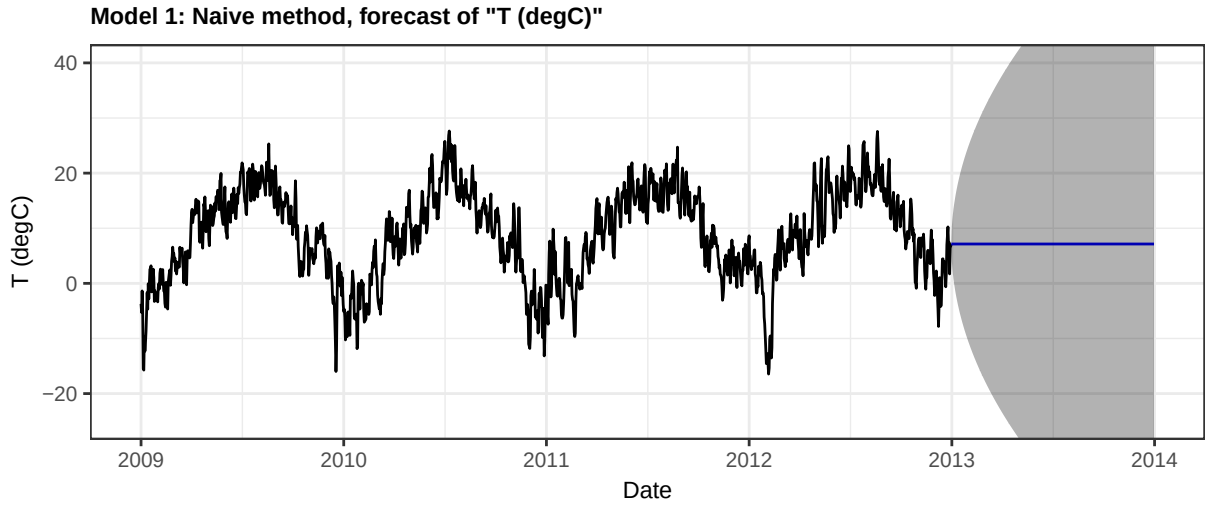
The R-function *accuracy()* is used to quantify the quality of the different models regarding fitting the *training* dataset and forecasting the *test* datatset. Following selection of methods is used for comparison:

- **MAE: Mean Absolute Error** - mean of all differences between actual and forecasted absolute values, see Wikipedia-MAE
- **RMSE: Root Mean Squared Error** - the sample standard deviation of differences between actual and forecasted values, see Wikipedia-RMSE
- **MASE: Mean Absolute Scaled Error** - measures the mean absolute error of the forecast values, divided by the mean absolute error of the *naive* forecast. Wikipedia-MASE
- **MAPE: Mean Absolute Percentage Error** - measures the difference of forecast errors and divides it by the actual value, see Wikipedia-MAPE

3.3 Model 1: Naive Method

The naive model projects the last observation into the future. So all forecasts are constant and equal to the last observed value. Here in this case the predicted constant value for the year 2013 is 7.1420833.

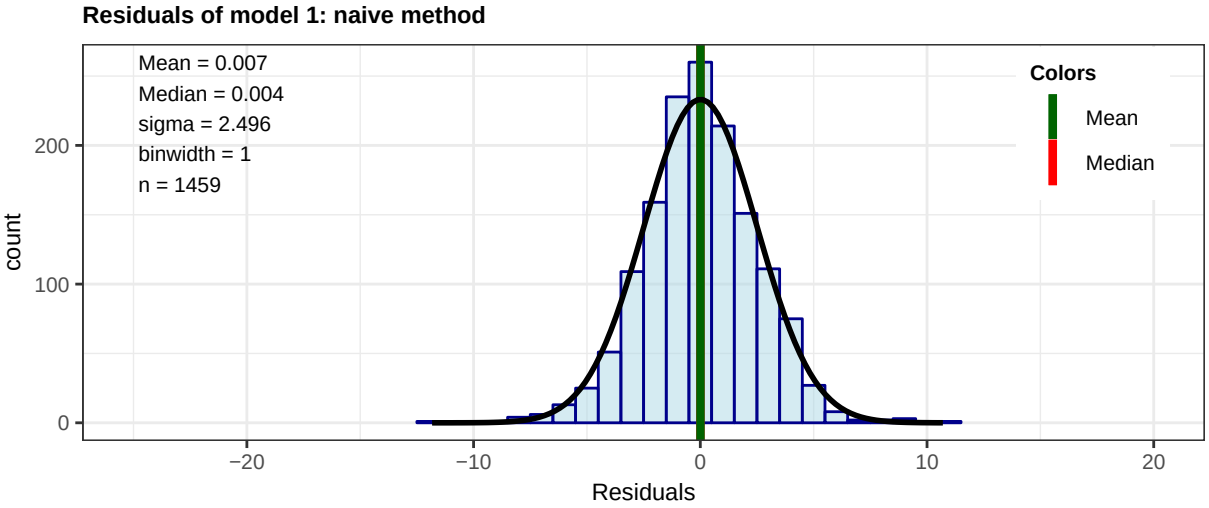
The prediction provided by this model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0074505	2.49558	1.925976	-13.29462	101.8107	0.424108	0.0990825	NA
Test set	1.9520487	8.38258	7.158433	-363.25914	696.2888	1.576317	0.9599685	24.76179

The residual distribution of this model is shown by a histogram:

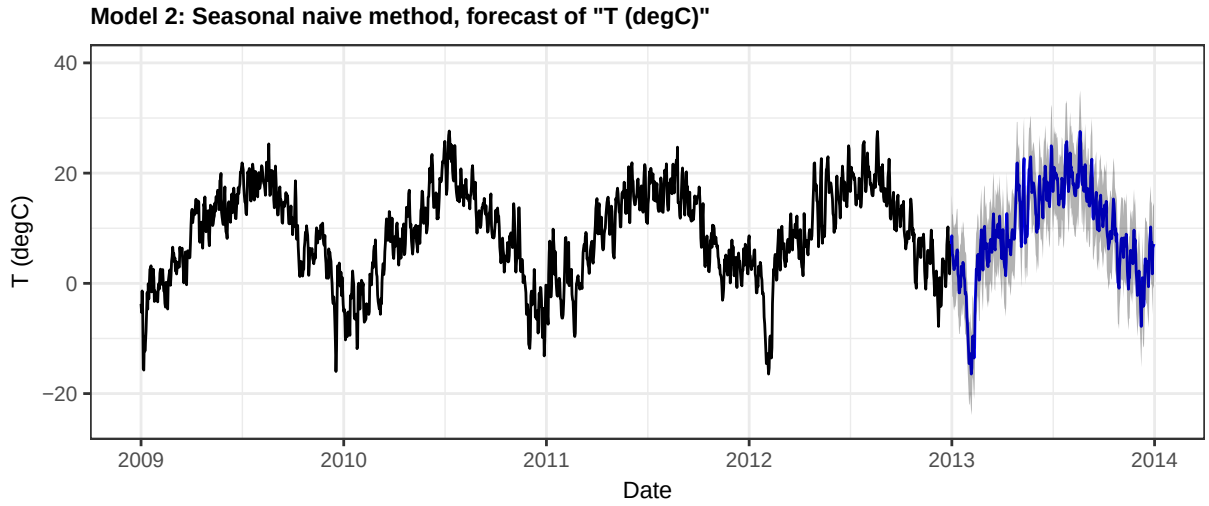


The residuals tell a lot about the quality of the model. They are what is left after the fitted values (here it is only one constant value) got subtracted from the original dataset. Only randomness should stay in the residuals. Everything else should be included in the model. Therefore the residuals should have a mean of zero, constant variance and ideally they are normally distributed.

3.4 Model 2: Seasonal Naive Method

Each forecast is equal to the last value from the same season. In this case here the forecast values for 2013 are just the same values as from the year 2012. Same day, same month, just the year is replaced.

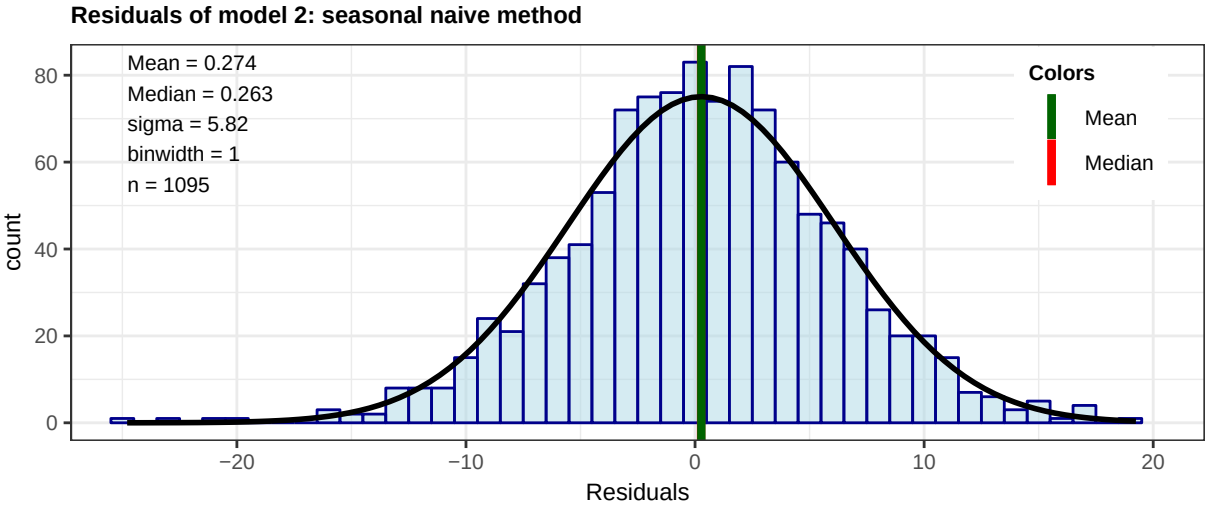
The prediction provided by this model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.2740457	5.823403	4.541240	-297.9243	546.9821	1.000000	0.8080532	NA
Test set	-0.5665716	6.470656	4.913277	269.9433	478.3431	1.081924	0.8620621	4.874977

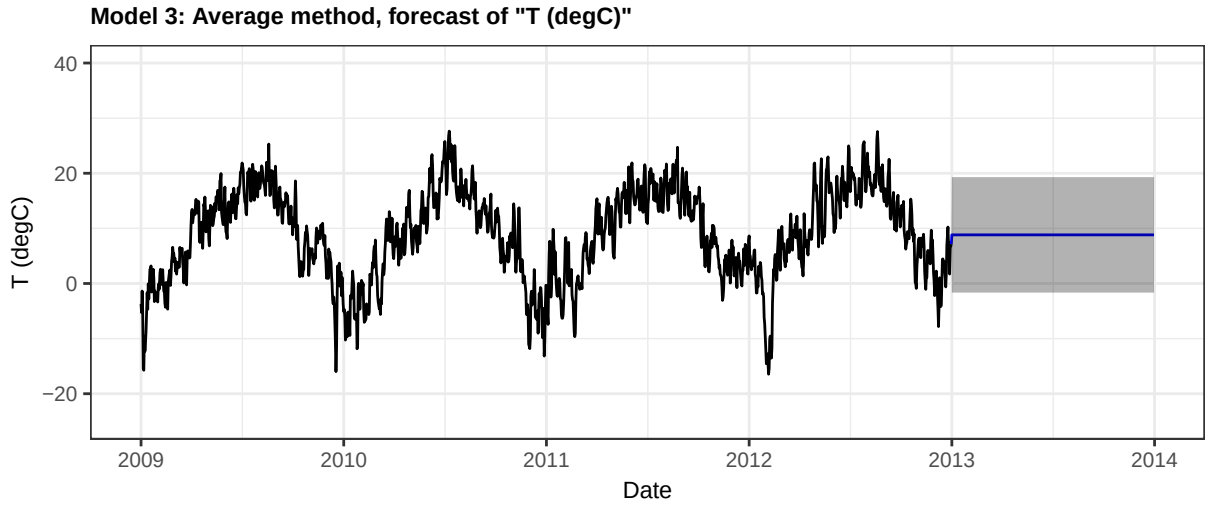
The residual distribution of this model is shown by a histogram:



3.5 Model 3: Average Method

The average model calculates the mean of the given historical data. All forecasts are constant and equal to this mean value. Here in this case it turns out, that the calculated average value is 8.8246051, which is very similar to the last observation used for the naive model (7.1420833).

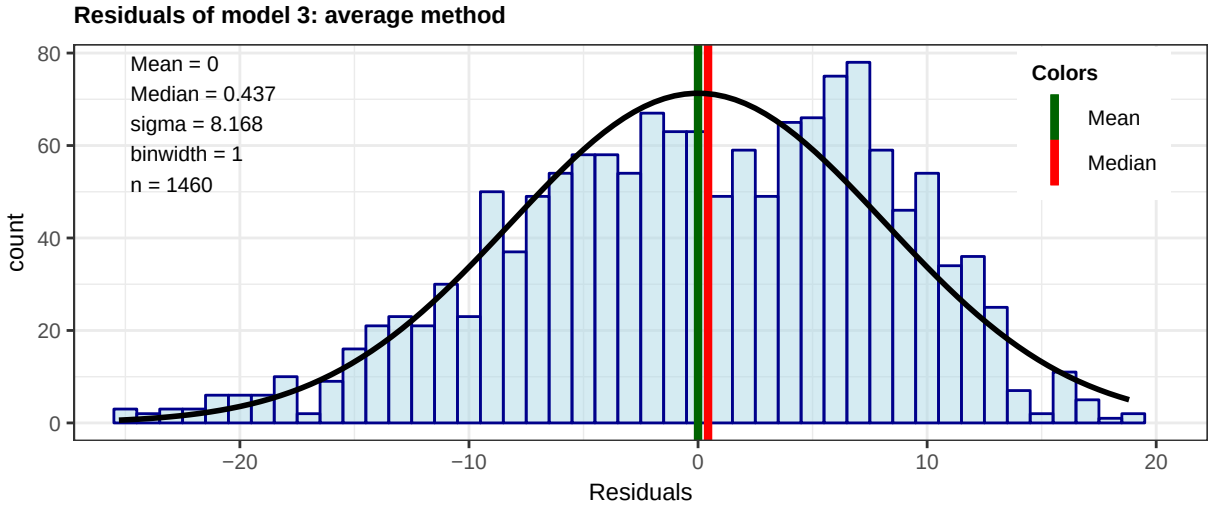
The prediction provided by this model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0000000	8.16504	6.730552	-186.9798	654.2696	1.482096	0.9524995	NA
Test set	0.2695269	8.15658	6.972895	-472.3931	850.3962	1.535461	0.9599685	30.68584

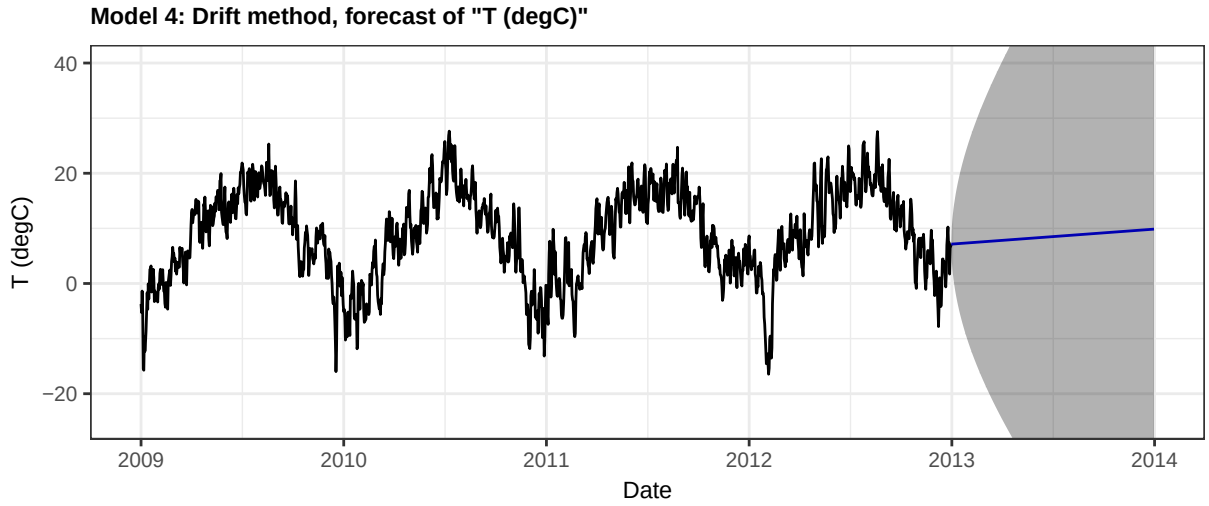
The residual distribution of this model is shown by a histogram:



3.6 Model 4: Drift Method

The drift method is best described as a forecast equal to last value plus average change. It calculates the difference between first and last observation and extrapolates that gradient into the future. Here the slope is small as there is no obvious trend present.

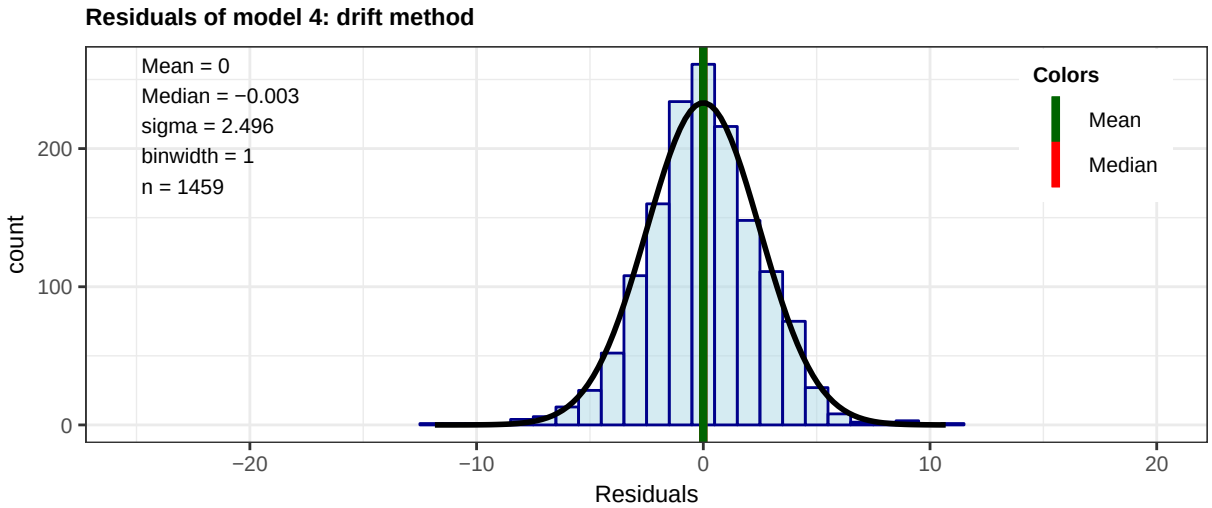
The prediction provided by this model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0000000	2.495569	1.925976	-13.53721	102.0374	0.4241079	0.0990825	NA
Test set	0.5886074	7.957176	6.849685	-501.57763	853.2988	1.5083291	0.9574310	33.64937

The residual distribution of this model is shown by a histogram:

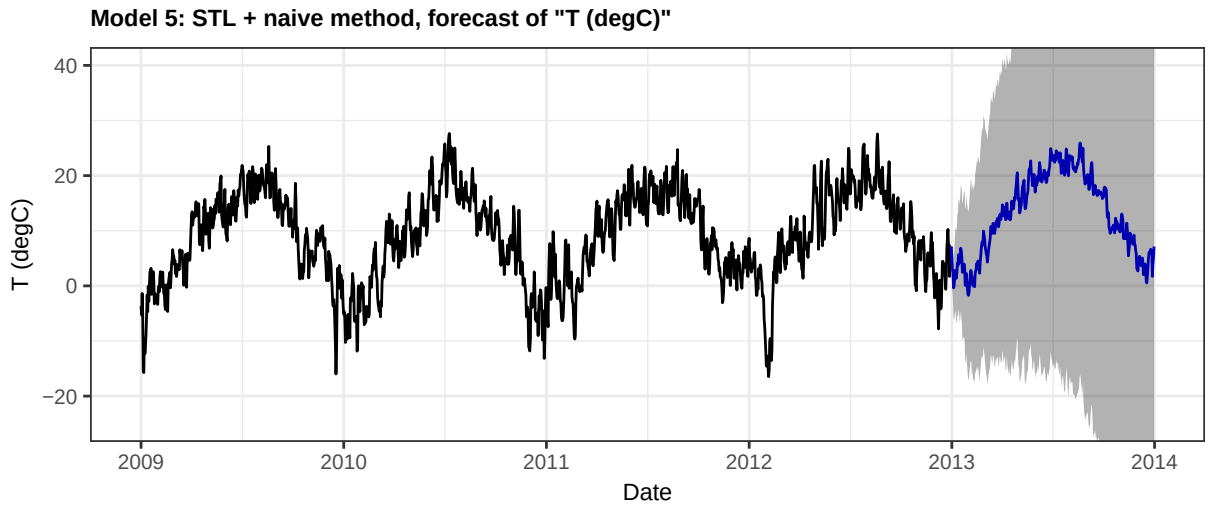


3.7 Model 5: Seasonal Decomposition + Naive Method

Seasonal decomposition means separating a seasonal time series into its constituent components, which are a seasonal component, a trend component and a random component. The acronym *STL* is related to this method, i.e. decompose a time series into seasonal, trend and irregular components using loess.

Forecasts of STL objects are obtained by applying a non-seasonal forecasting method to the seasonally adjusted data and re-seasonalizing using the last year of the seasonal component. Here the naive method is applied.

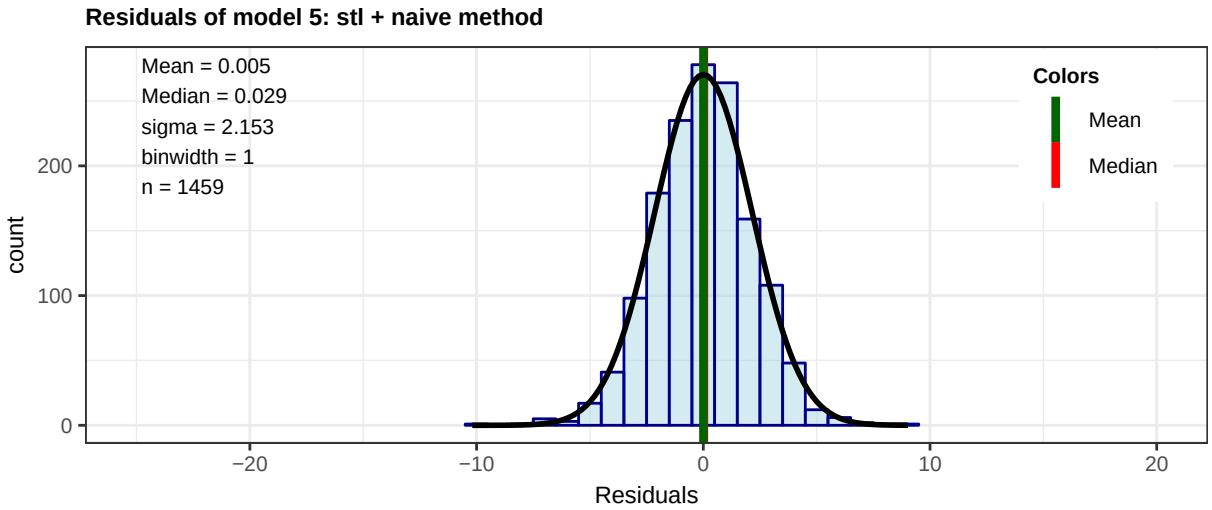
The prediction provided by this model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0051251	2.152354	1.690407	20.11617	108.3317	0.3722347	0.0879339	NA
Test set	-3.9688554	6.558164	5.267545	-110.44978	495.6259	1.1599353	0.8745245	11.93835

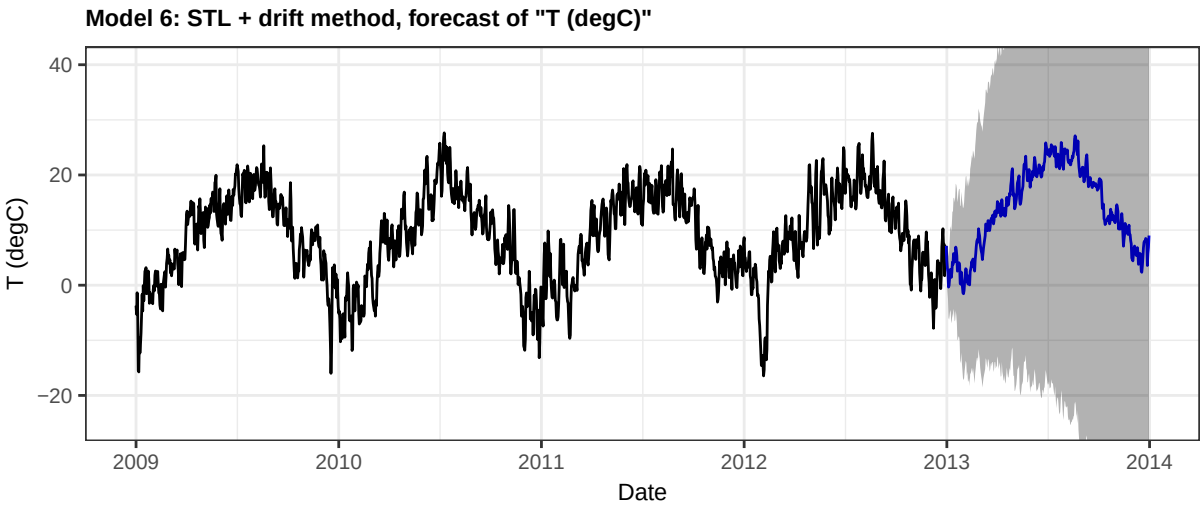
The residual distribution of this model is shown by a histogram:



3.8 Model 6: Seasonal Decomposition + Drift Method

This model uses the STL method like described before and applies the drift method to obtain a forecast.

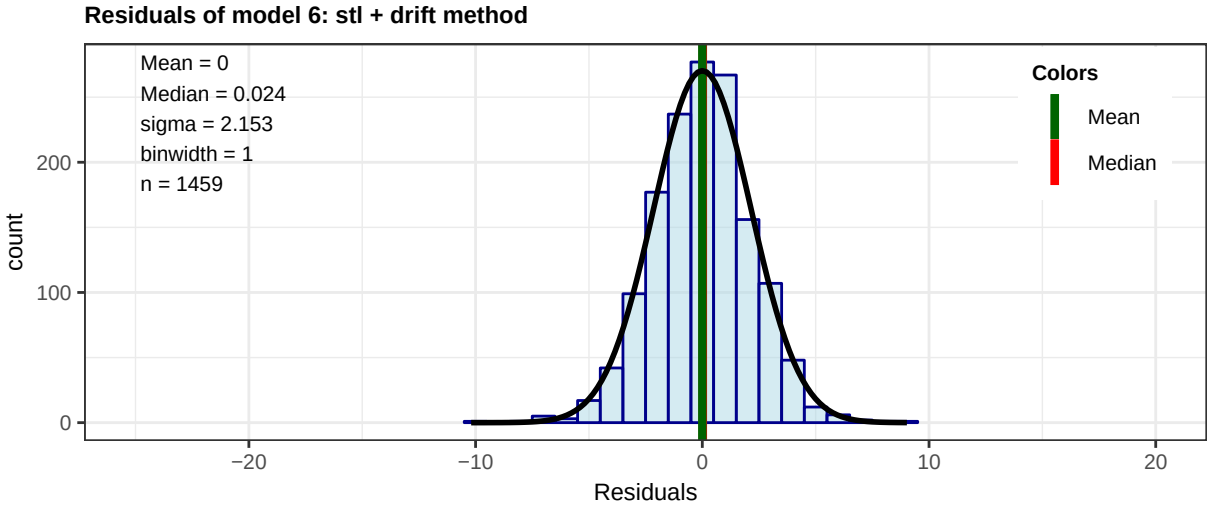
The prediction provided by this model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.000000	2.152347	1.690334	19.94928	108.1788	0.3722185	0.0879339	NA
Test set	-4.906758	7.086769	5.837008	-205.59814	607.6533	1.2853334	0.8684414	17.49467

The residual distribution of this model is shown by a histogram:

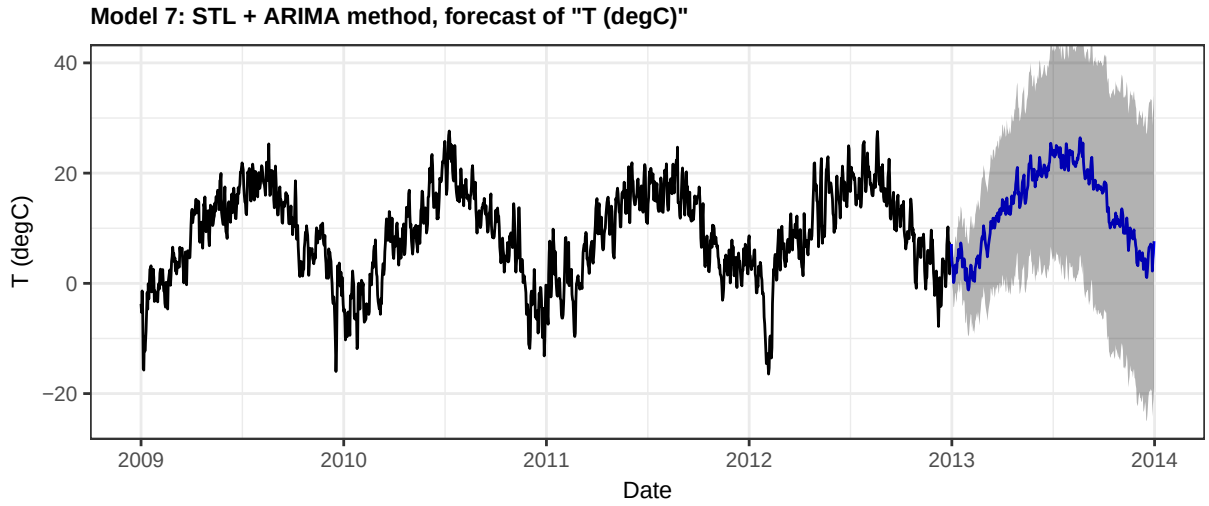


As there is no distinct drift present, the resulting values of this model look quite similar to the values obtained by the *seasonal decomposition + naive model*.

3.9 Model 7: Seasonal Decomposition + ARIMA Method

This model uses the STL method and applies the ARIMA method to obtain a forecast. The ARIMA model is described later as a separate method.

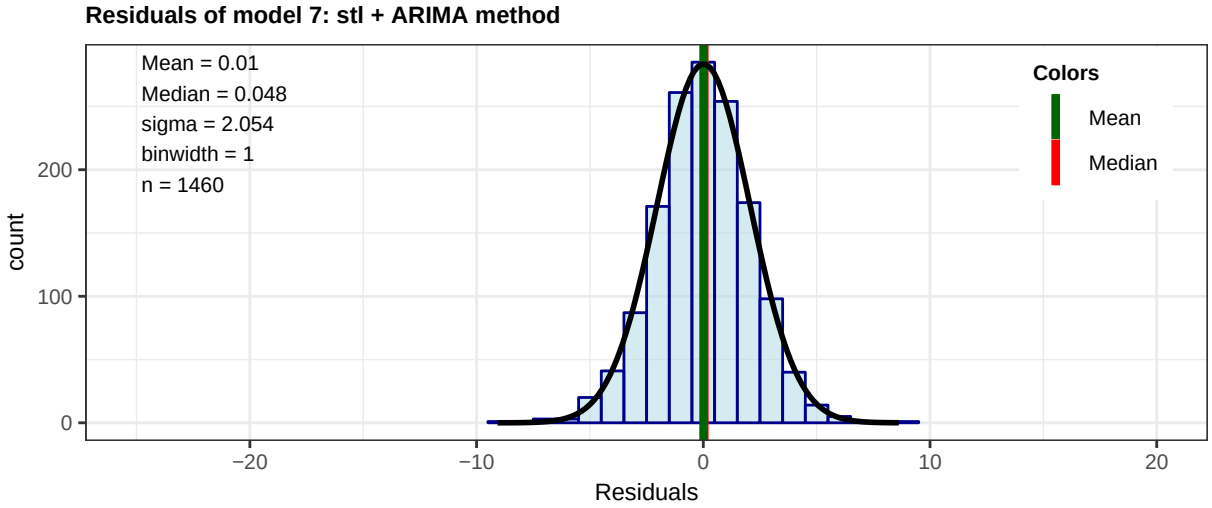
The prediction provided by this model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.010102	2.053543	1.614299	-21.29192	104.5006	0.3554753	0.0245485	NA
Test set	-4.470851	6.876190	5.575407	-143.15375	543.3810	1.2277278	0.8742186	13.57744

The residual distribution of this model is shown by a histogram:



The residuals are very close to a normal distribution.

3.10 Model 8: Seasonal ARIMA Method

The Autoregressive Integrated Moving Average (ARIMA) model for univariate time series combines three parts with the integers (p,d,q) denoting the grade of order of the three parts:

- **AR** - autoregressive part (p)
- **I** - integration, i.e. degree of differencing (d)
- **MA** - moving average part (q)

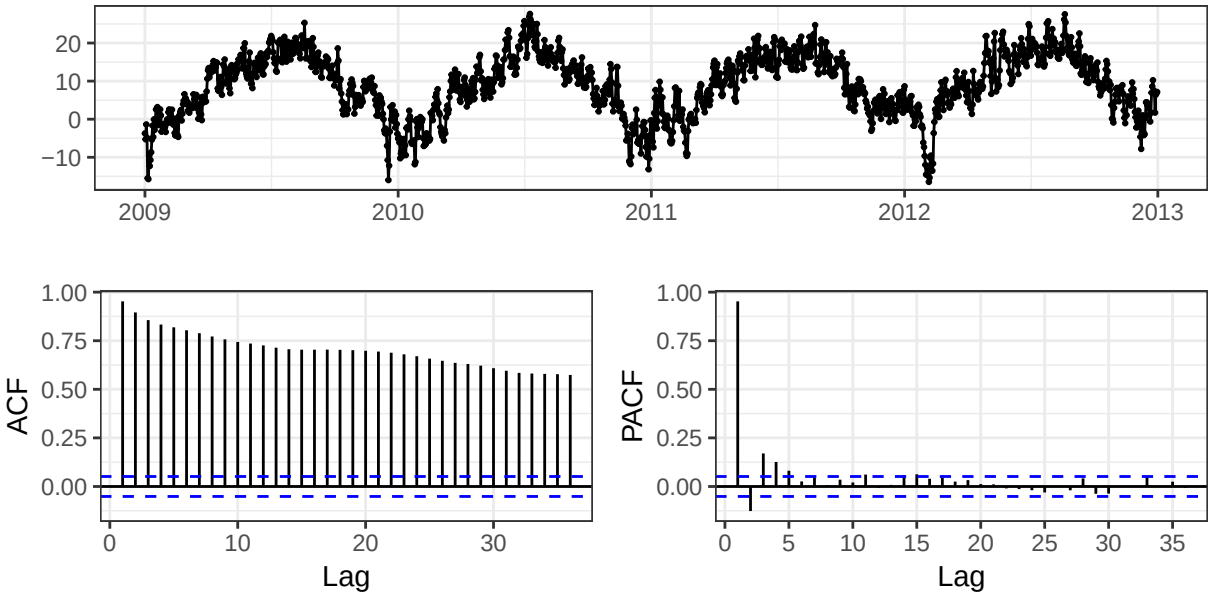
For non-stationary time series differencing is applied d-times to obtain a stationary signal. This stationary signal is then represented by the *ARMA*-model. To obtain the values for the original non-stationary signal again, integration is performed on the results of the *ARMA*-model. The *ARMA*-model is defined as:

$$y_t = c + \epsilon_t + \sum_{i=1}^{p} a_i y_{t-i} + \sum_{j=1}^{q} b_j \epsilon_{t-j}$$

The signal y_t is build by a weighted moving average of random values $\epsilon_{t-j}, j = 1, ..., q$ of the q previous periods. The so-called **MA**-coefficients $b_j, j = 1, ..., q$ determine with which weight the random values contribute to the signal. Furthermore the signal is composed of a constant c, a random value ϵ_t and a weighted moving average of the previous p signal values y_{t-i}, where the **AR**-coefficients $a_i, i = 1, ..., p$ are the weights. Regarding the random values ϵ_t, it is assumed that they are time independent of each other and are normally distributed.

The goal is to determine the three describing parameters (p,d,q) of the *ARIMA*-model such that the *AIC*-value is as small as possible. The *Akaike Information Criterion (AIC)* is a widely used measure to quantify the fitting quality together with the simplicity of the model. The R-function *auto.arima()* uses the *AIC*-value to automatically determine the "best" parameters (p,d,q) for a given dataset.

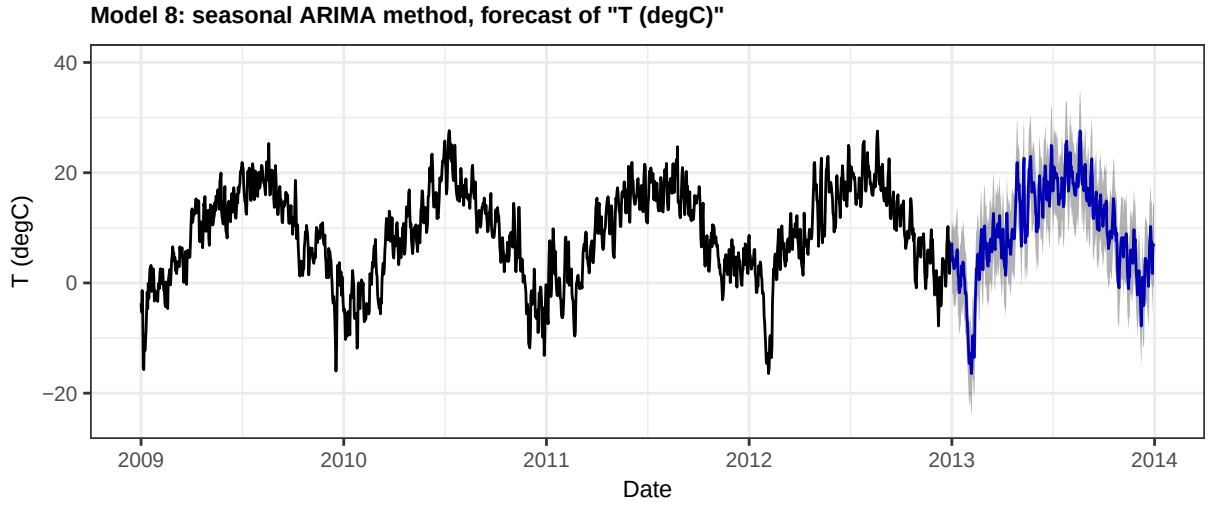
For the training dataset the autocorrelation function (*ACF*) and the partial autocorrelation function (*PACF*) show several bars ranging out of the 95% confidence intervals (blue lines).



There is a good chance that an *ARIMA*-model is able to reconstruct and forecast the signal with high accuracy, as...

- *ACF* is used to identify the moving average (MA) part of the ARIMA model (omit first bar)
- *PACF* is used to identify the autoregressive part (AR)

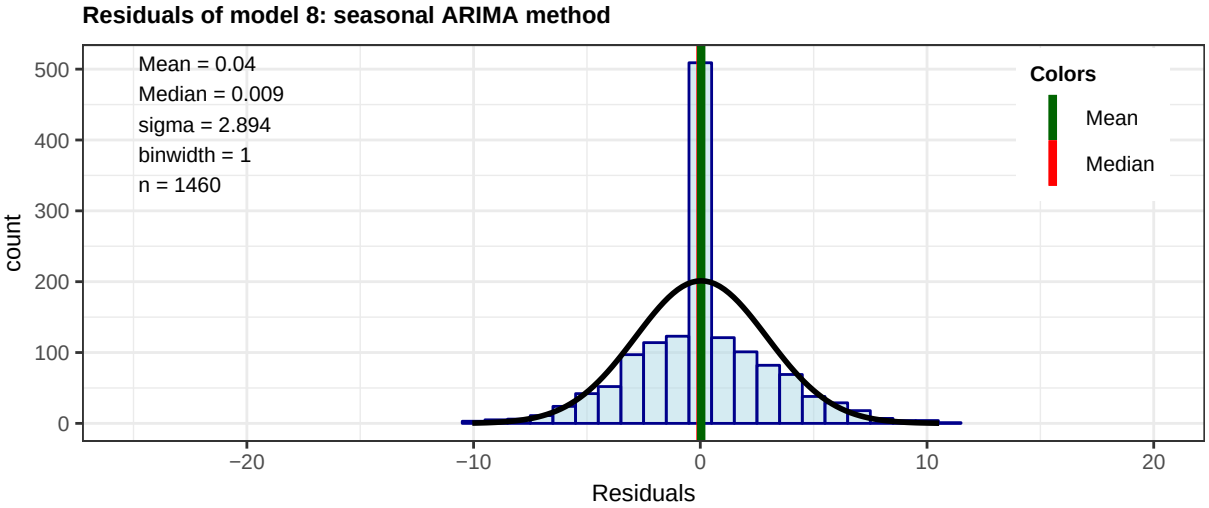
The prediction provided by the ARIMA model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0397264	2.893695	1.970405	-5.224373	82.08042	0.4338914	0.0022242	NA
Test set	-0.5564001	6.468982	4.907594	270.101389	478.22513	1.0806726	0.8624198	4.875011

The residual distribution of this model is shown by a histogram:

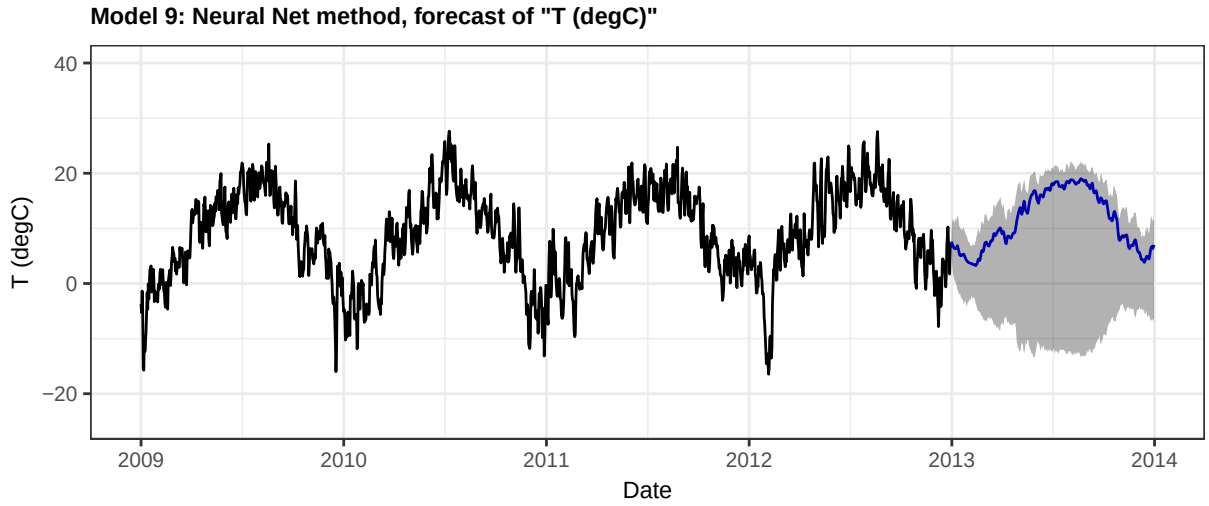


The residual distribution shows a huge peak for the zero bin. This might be an indication for overfitting the given data. At least the goal of obtaining a normal distribution is not achieved here.

3.11 Model 9: Neural Net Method

A neural network simplifies input variables over one or several layers into one output value. In this case a *Neural Network Autoregression Model* is used with p lagged values as input and k nodes in the hidden layer. The R function nnetar determines automatically the parameters p and k.

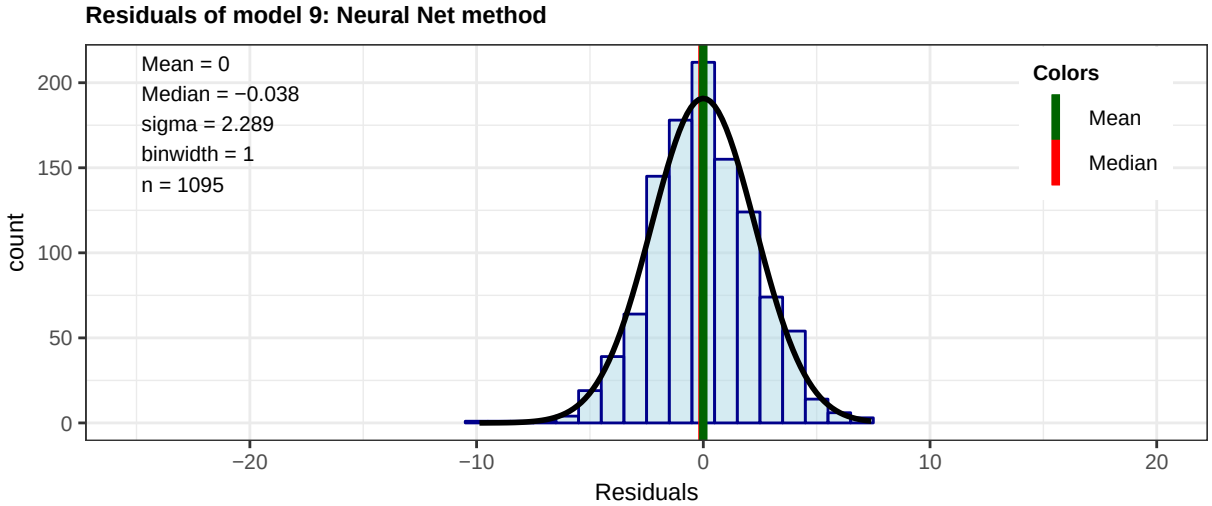
The prediction provided by this model for the year 2013 is shown as blue line:



With this model the following accuracies on training and test dataset can be achieved:

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0002198	2.288433	1.794453	-45.2161	107.2658	0.3951461	-0.0006603	NA
Test set	-2.2980066	5.404115	4.296053	-197.7045	509.4290	0.9460088	0.8885235	13.70635

The residual distribution of this model is shown by a histogram:

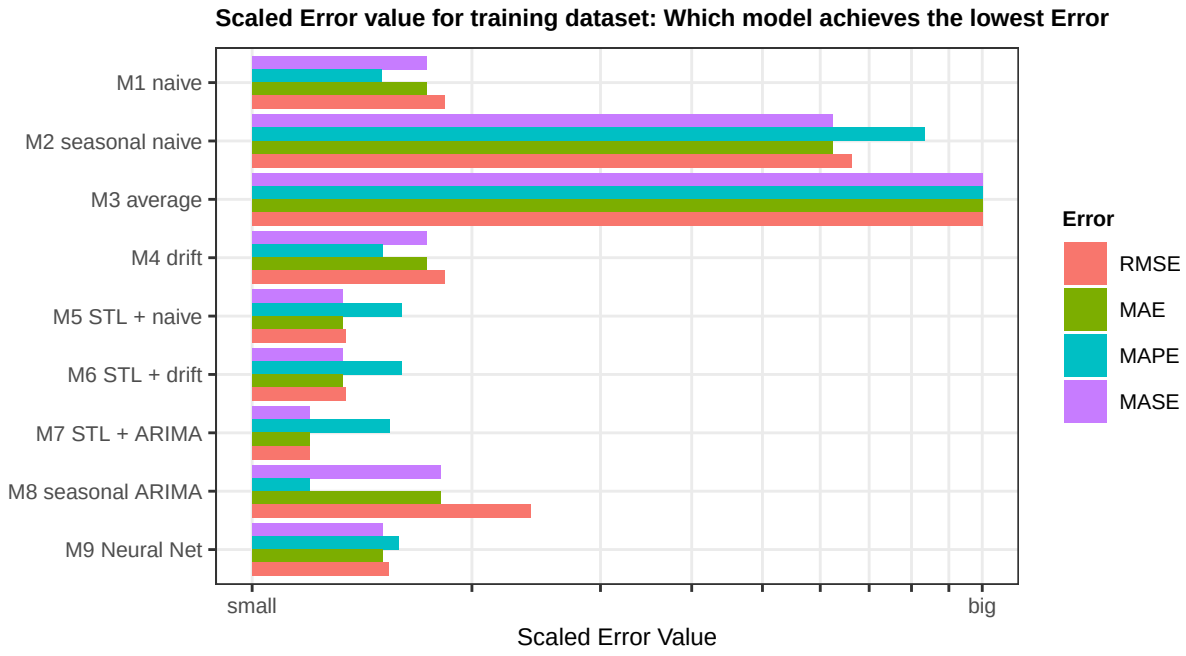


4 Results

Following table summarises the achieved accuracy values for fitting the training data:

Model	RMSE	MAE	MAPE	MASE
M1 naive	2.495580	1.925976	101.81066	0.4241080
M2 seasonal naive	5.823403	4.541240	546.98206	1.0000000
M3 average	8.165040	6.730552	654.26961	1.4820955
M4 drift	2.495569	1.925976	102.03736	0.4241079
M5 STL + naive	2.152354	1.690407	108.33172	0.3722347
M6 STL + drift	2.152347	1.690334	108.17878	0.3722185
M7 STL + ARIMA	2.053543	1.614299	104.50059	0.3554753
M8 seasonal ARIMA	2.893695	1.970405	82.08042	0.4338914
M9 Neural Net	2.288433	1.794453	107.26577	0.3951461

Visualization of accuracy as *Scaled Error* value for training dataset:

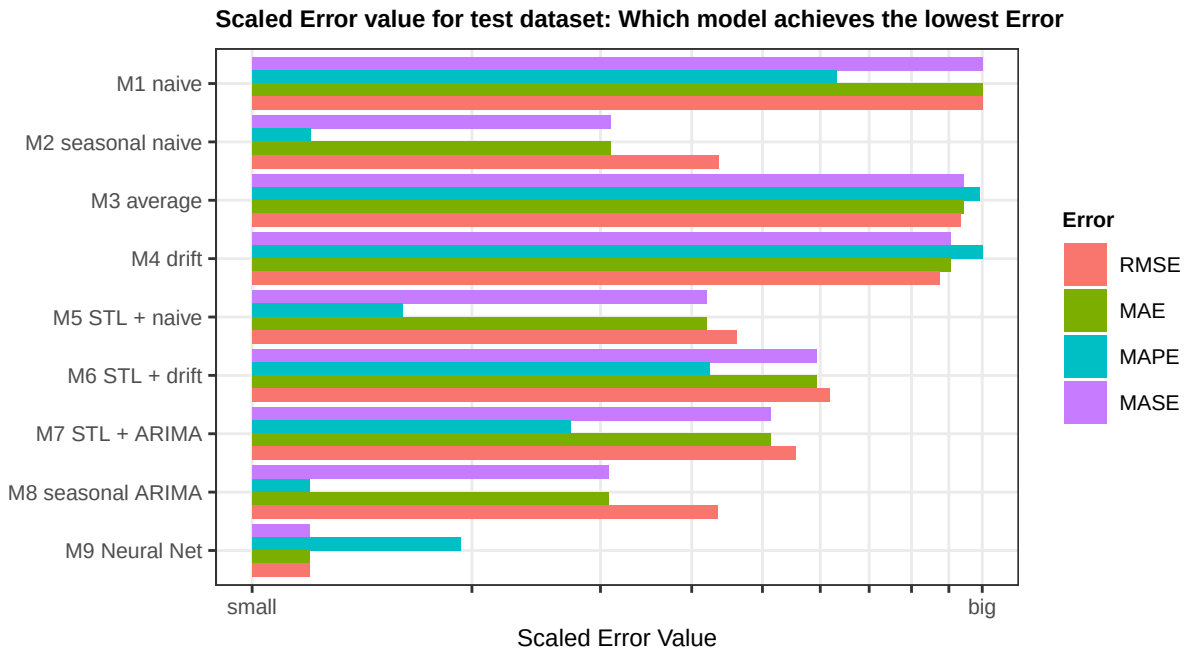


Model *M7*, the *Seasonal Decomposition + ARIMA Method* shows the best accuracy for fitting the training data.

Following table summarises the achieved accuracy values for forecasting the test data:

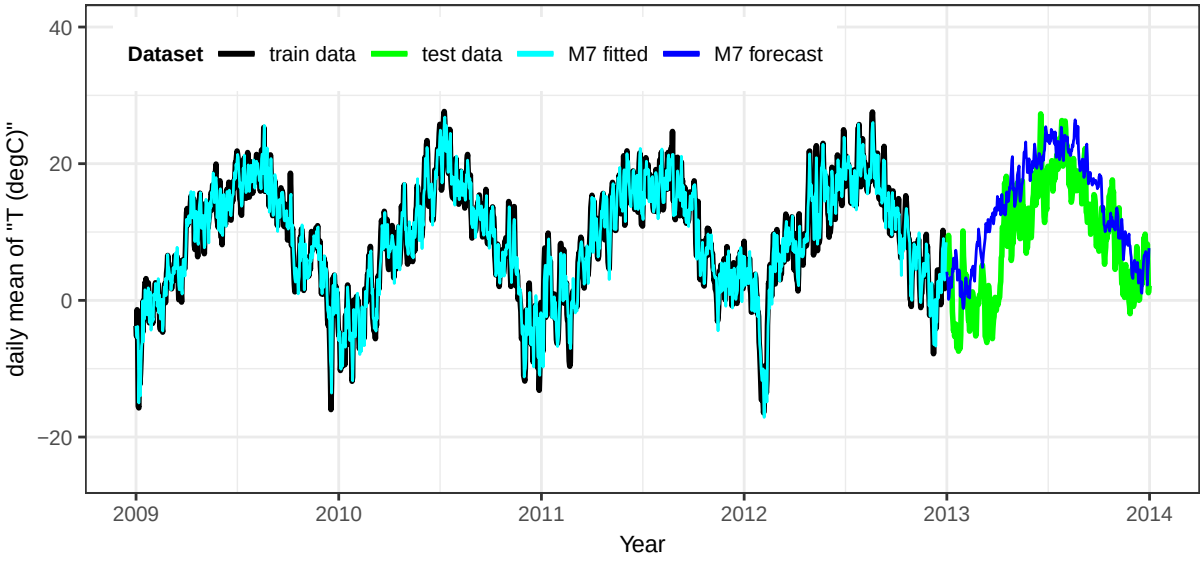
Model	RMSE	MAE	MAPE	MASE
M1 naive	8.382580	7.158433	696.2888	1.5763168
M2 seasonal naive	6.470656	4.913277	478.3431	1.0819241
M3 average	8.156580	6.972895	850.3962	1.5354606
M4 drift	7.957176	6.849685	853.2988	1.5083291
M5 STL + naive	6.558164	5.267545	495.6259	1.1599353
M6 STL + drift	7.086769	5.837008	607.6533	1.2853334
M7 STL + ARIMA	6.876190	5.575407	543.3810	1.2277278
M8 seasonal ARIMA	6.468982	4.907594	478.2251	1.0806726
M9 Neural Net	5.404115	4.296053	509.4290	0.9460088

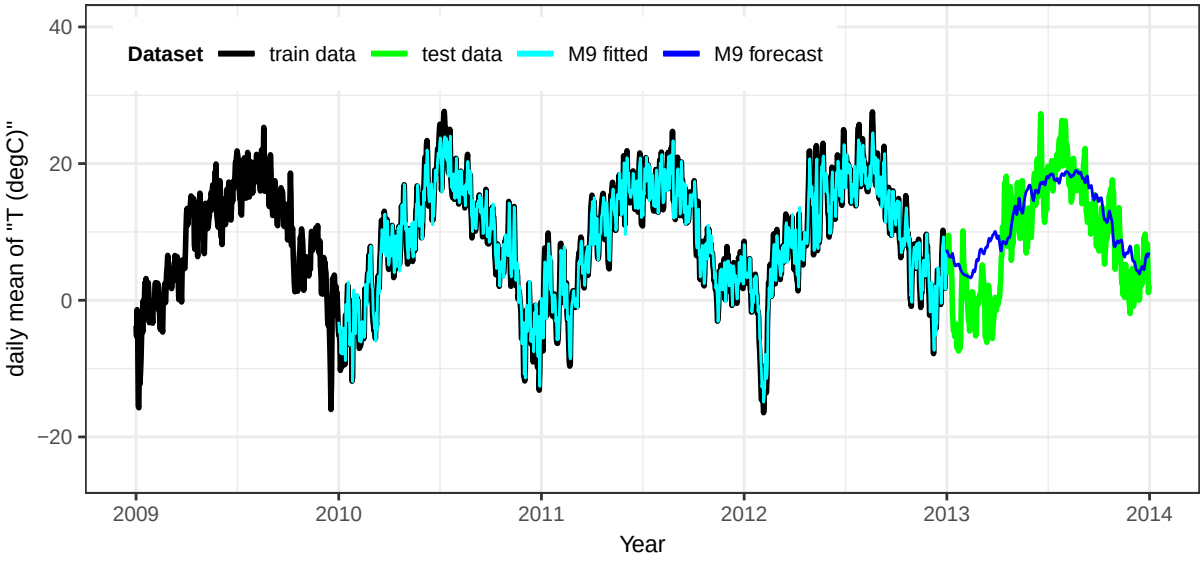
Visualization of accuracy as *Scaled Error* value for test dataset:

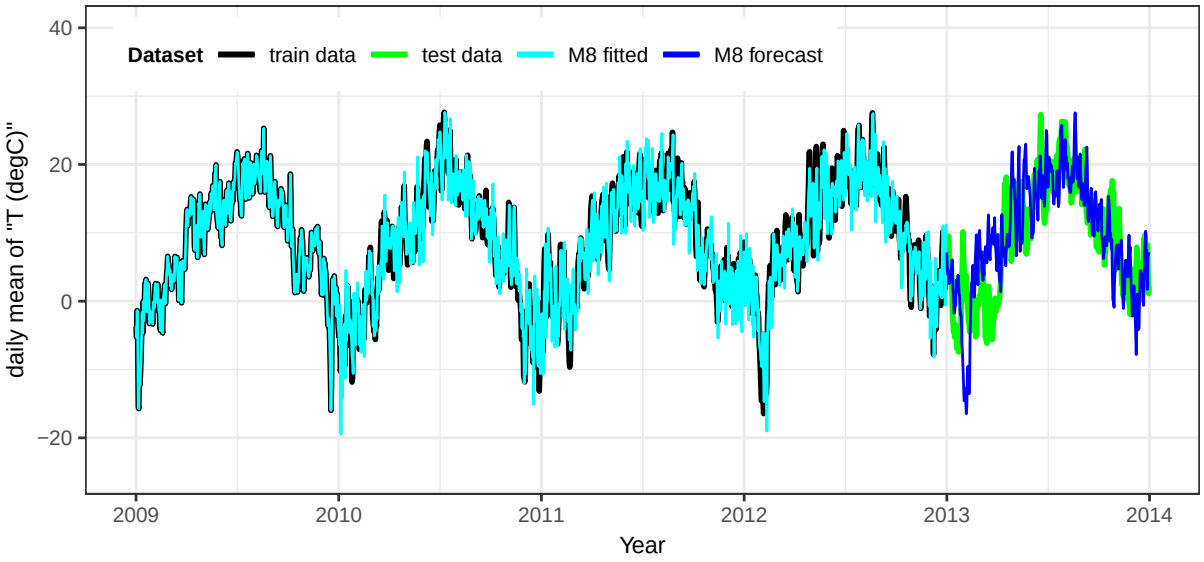


Model *M9*, the *Neural Net Method* provides the best accuracy regarding forecasting the test data. Also the seasonal methods *M2 - Seasonal Naive Method* and *M8 - Seasonal ARIMA Method* do a quite good job. The three models were not the favorites when looking on the accuracy for fitting the training dataset.

The following plots visualize the fitted and forecasted values of the models M7, M9 and M8:


Dataset
train data
test data
M7 fitted
M7 forecast
daily mean of "T (degC)"
40
20
0
−20
2009
2010
2011
2012
2013
2014
Year


Dataset
train data
test data
M9 fitted
M9 forecast
daily mean of "T (degC)"
40
20
0
−20
2009
2010
2011
2012
2013
2014
Year


Dataset
train data
test data
M8 fitted
M8 forecast
daily mean of "T (degC)"
40
20
0
−20
2009
2010
2011
2012
2013
2014
Year

5 Conclusion

In this project machine learning algorithms were used to forecast the daily mean temperatures of one year by fitting the models to the available data of the previous four years. The quality of this task was quantified by a selection of accuracy measurements like RMSE, MAE, MASE and MAPE.

A variety of models were used: Some of the simple ones like naive-, average- or drift-methods already achieve good accuracy. This might be because a large part of the signal (=temperatures) is noise, which usually cannot be forecasted well by sophisticated models. It is also obvious that the signal shows high seasonality. Therefore models based on seasonal decomposition (STL) show quite good performance. Another standard tool is the ARIMA-model. In the simple form, like it was used here, it tended to *overfitt* the training data. So the forecast accurcacy was dissapointing. The neural net model on the other hand fitted the training data with fair quality and showed best forecast accuracy.

It is likely that much more sophisticated methods would achieve better accuracy. So far this project was just a first survey of a selection of methods without any detailed investigation or optimization of each used model. Furthermore the validation by dividing the data into one training and one test dataset could be improved. For example an approach like cross-validation could help to avoid overfitting. For simplicity reasons it was not applied so far. The implementation is left to future work.

6 Appendix - Sessioninfo

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.6 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## locale:
##  [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=de_DE.UTF-8        LC_COLLATE=de_DE.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=de_DE.UTF-8
##  [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] nortest_1.0-4    ggfortify_0.4.6  tseries_0.10-46  quadprog_1.5-7
##  [5] lubridate_1.7.4  forecast_8.7     ggrepel_0.8.1    corrplot_0.84
##  [9] xts_0.11-2       zoo_1.8-5        latex2exp_0.4.0  reshape2_1.4.3
## [13] scales_1.0.0     munsell_0.5.0    kableExtra_1.1.0 knitr_1.22
## [17] caret_6.0-84     lattice_0.20-38  forcats_0.4.0    stringr_1.4.0
## [21] dplyr_0.8.0.1    purrr_0.3.2      readr_1.3.1      tidyr_0.8.3
## [25] tibble_2.1.1     ggplot2_3.1.1    tidyverse_1.2.1  colorspace_1.4-1
##
## loaded via a namespace (and not attached):
```

```
##  [1] httr_1.4.0         jsonlite_1.6       viridisLite_0.3.0
##  [4] splines_3.6.1      foreach_1.4.4      prodlim_2018.04.18
##  [7] modelr_0.1.4       assertthat_0.2.1   TTR_0.23-4
## [10] stats4_3.6.1       cellranger_1.1.0   yaml_2.2.0
## [13] ipred_0.9-9        pillar_1.4.0       backports_1.1.4
## [16] glue_1.3.1         digest_0.6.18      rvest_0.3.3
## [19] recipes_0.1.5      htmltools_0.3.6    Matrix_1.2-18
## [22] plyr_1.8.4         timeDate_3043.102  pkgconfig_2.0.2
## [25] broom_0.5.2        haven_2.1.0        webshot_0.5.1
## [28] gower_0.2.0        lava_1.6.5         generics_0.0.2
## [31] withr_2.1.2        urca_1.3-0         nnet_7.3-12
## [34] lazyeval_0.2.2     quantmod_0.4-14    cli_1.1.0
## [37] survival_3.1-8     magrittr_1.5       crayon_1.3.4
## [40] readxl_1.3.1       evaluate_0.13      nlme_3.1-142
## [43] MASS_7.3-51.4      xml2_1.2.0         class_7.3-15
## [46] tools_3.6.1        data.table_1.12.2  hms_0.4.2
## [49] compiler_3.6.1     rlang_0.3.4        iterators_1.0.10
## [52] rstudioapi_0.10    labeling_0.3       rmarkdown_1.12
## [55] fracdiff_1.4-2     gtable_0.3.0       ModelMetrics_1.2.2
## [58] codetools_0.2-16   curl_3.3           R6_2.4.0
## [61] gridExtra_2.3      stringi_1.4.3      parallel_3.6.1
## [64] Rcpp_1.0.1         rpart_4.1-15       lmtest_0.9-37
## [67] tidyselect_0.2.5   xfun_0.6
```