wastewater

Jakub Owczarek

2023-11-26

GAM Model for Wastewater Time-Series

Description

This case study attempts to investigate the usage of Generalized Additive Model (GAM) on Wastewater dataset from Sandomierz, evaluate it's performance and compare it to another popular time-series forecasting model.

It was done for Data Mining course led by Ph.D Monika Chuchro on final year of Data Engineering and Analysis course on AGH UST.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyr)
library(readr)
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```
## This is mgcv 1.9-0. For overview type 'help("mgcv-package")'.
```

```
library(prophet)
```

```
## Loading required package: Rcpp
```

```
## Loading required package: rlang
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

Load data

First let's load our data and see the first few rows.

```
df <- read_delim(
  "wastewater.txt",
  delim = "\t",
  col_names = c("Date", "Sandomierz"),
  skip = 1
)
```

```
## Rows: 66 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (2): Date, Sandomierz
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(df)
```

```
## # A tibble: 6 x 2
##    Date Sandomierz
##   <dbl>      <dbl>
## 1 2002.       3.22
## 2 2002.       2.80
## 3 2002.       2.89
## 4 2002.      NA
## 5 2002.       2.98
## 6 2002.       3.10
```

Our dataset is a time series with a single column. Right off the bat we can see we'll deal with some missing values. Let's check how many rows we have.

```
nrow(df)
```

```
## [1] 66
```

Now let's check our datatypes.

```
str(df)
```

```
## spc_tbl_ [66 x 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Date      : num [1:66] 2002 2002 2002 2002 2002 ...
##  $ Sandomierz: num [1:66] 3.22 2.8 2.89 NA 2.98 ...
```

```
##  - attr(*, "spec")=
##   .. cols(
##   ..   Date = col_double(),
##   ..   Sandomierz = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

We can see that our date column inferred `num` type. Let's fix that by casting it to `date`.

```
df$Date <- as.character(df$Date)

df <- df %>%
  mutate(
    Year = floor(as.numeric(Date)),
    Month = round((as.numeric(Date) - Year) * 100),
    Date = ymd(paste(Year, Month, "1"))
  ) %>%
  select(-Year, -Month)

head(df)
```

```
## # A tibble: 6 x 2
##   Date       Sandomierz
##   <date>          <dbl>
## 1 2002-01-01       3.22
## 2 2002-02-01       2.80
## 3 2002-03-01       2.89
## 4 2002-04-01      NA
## 5 2002-05-01       2.98
## 6 2002-06-01       3.10
```

Finally let's check for missing values.

```
sum(is.na(df$Sandomierz))
```

```
## [1] 1
```

There is only one missing value so we will interpolate it.

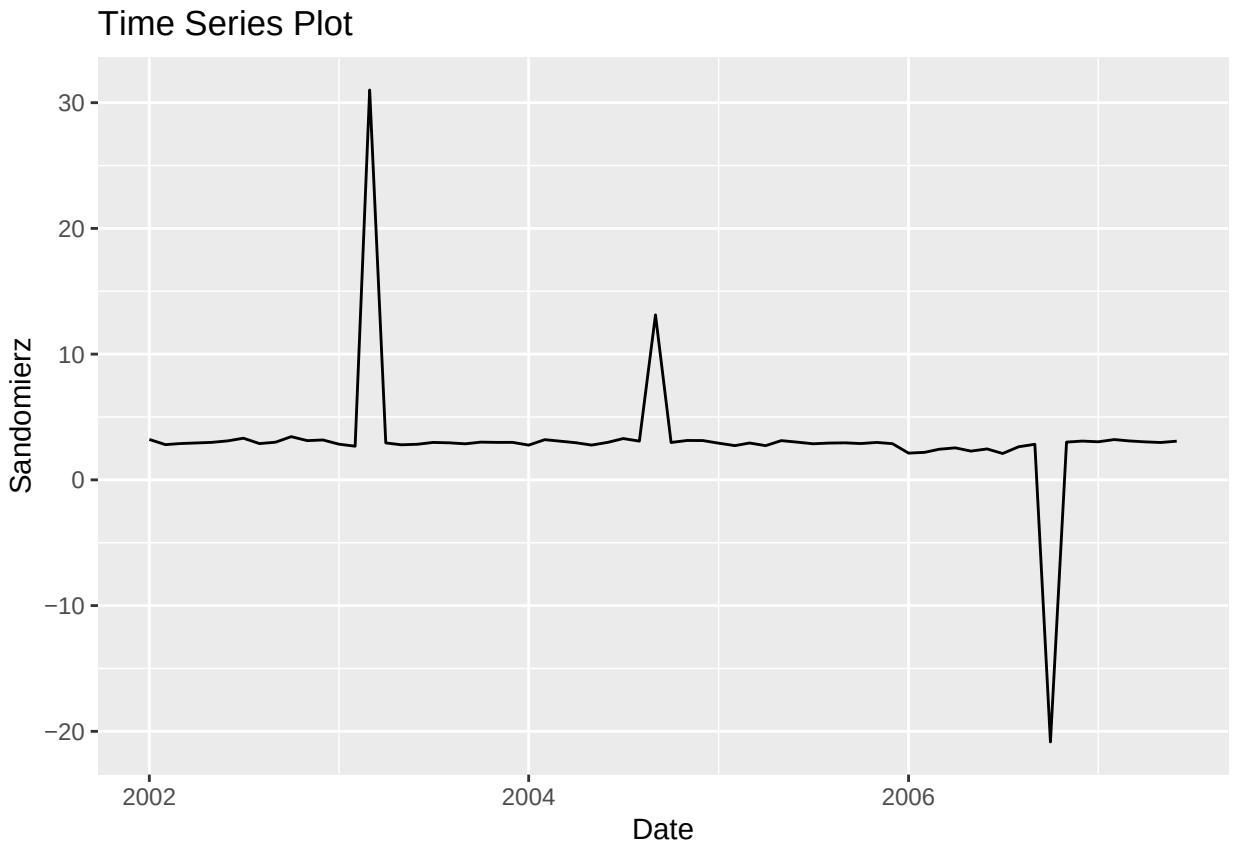
```
df$Sandomierz <- na.approx(df$Sandomierz)
```

Exploratory Data Analysis

Now let's perform Exploratory Data Analysis to see how our data looks.

Let's start by doing a simple plot.

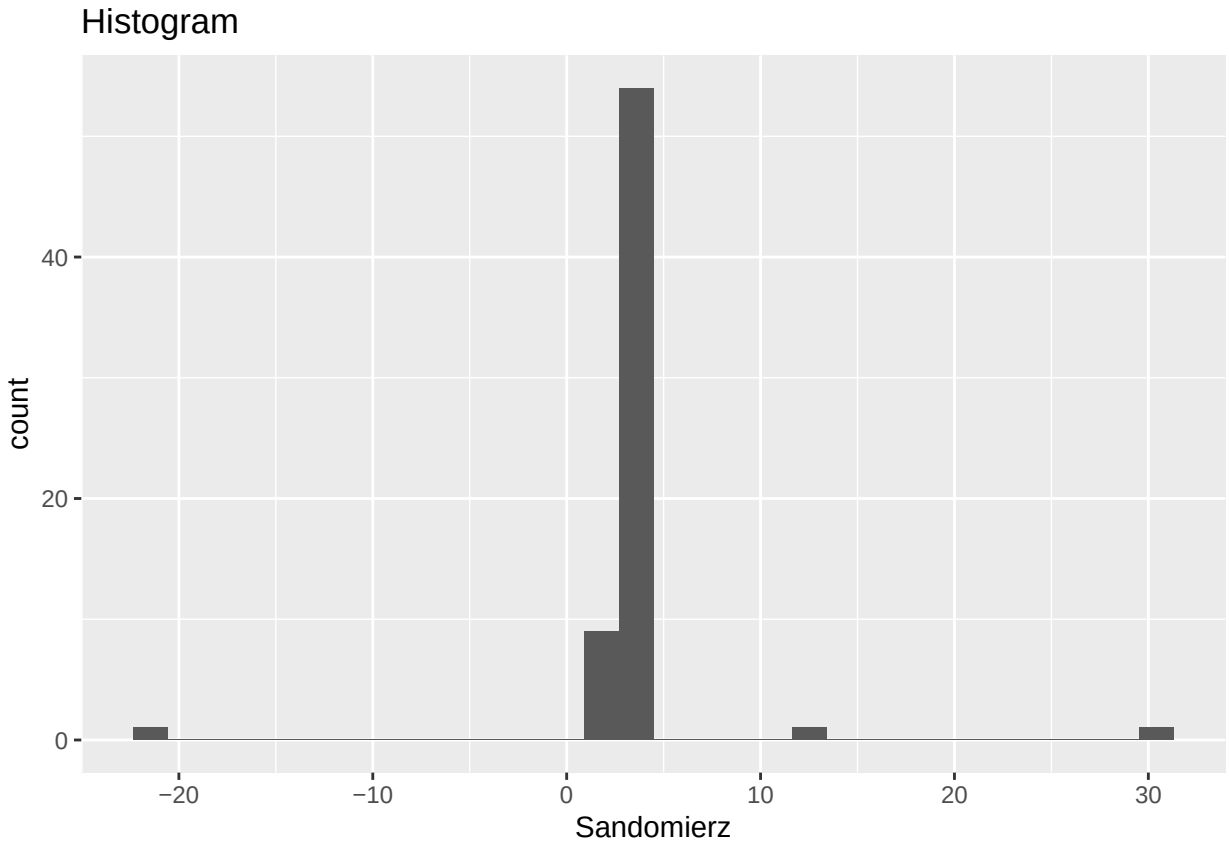
```
ggplot(df, aes(x = Date, y = Sandomierz)) +
  geom_line() +
  labs(title = "Time Series Plot")
```



There seem's to be 3 outliers in our dataset, we will deal with it later.

Let's also see how our values are distributed on a histogram.

```
ggplot(df, aes(x = Sandomierz)) +
  geom_histogram(bins = 30) +
  labs(title = "Histogram")
```

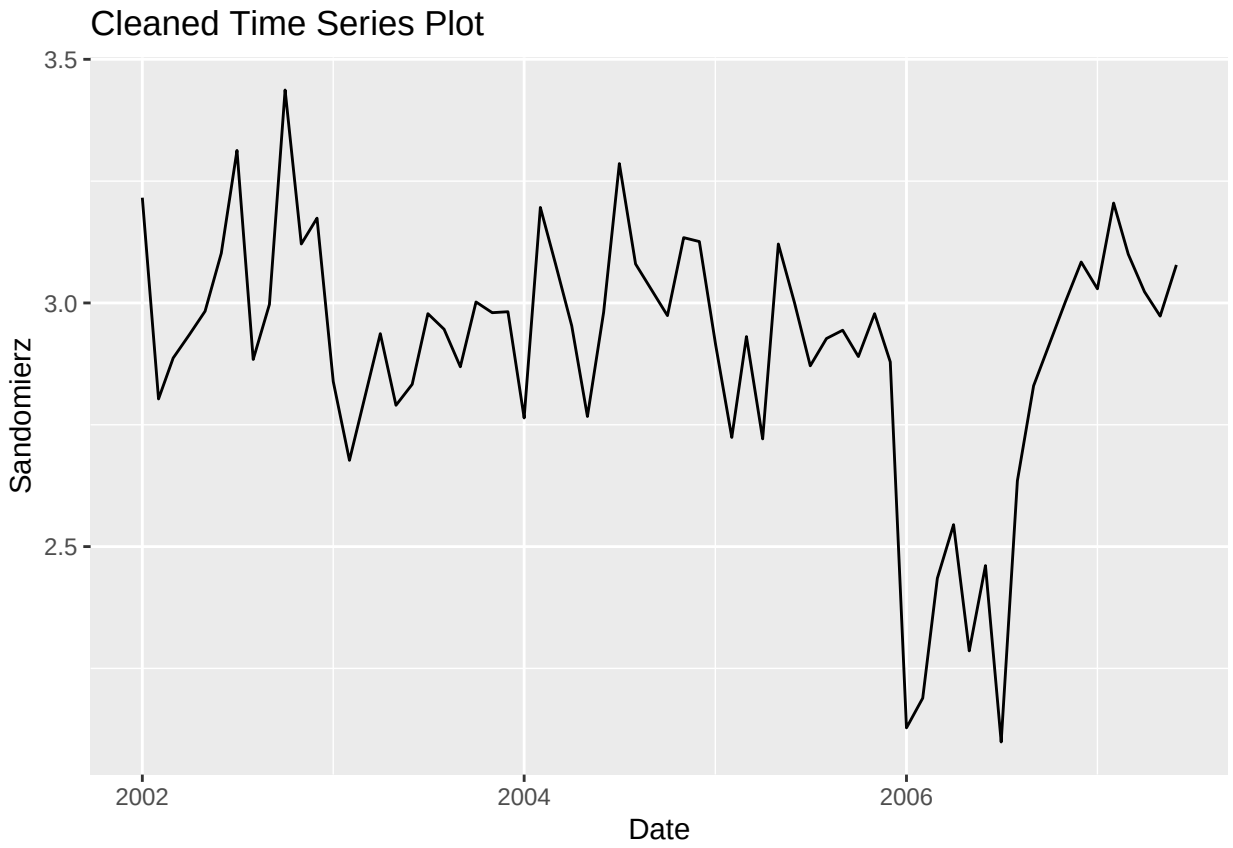


We can clearly see that we have 3 outliers, so let's change them to missing values and interpolate them. One of our outliers is close to the majority of our values. A common practice is to label values with a Z-score greater than 2 or 3 as outliers, but in our case we will assume that an outlier is a value larger than 1.

```
mean_value <- mean(df$Sandomierz, na.rm = TRUE)
std_dev <- sd(df$Sandomierz, na.rm = TRUE)

df$z_score <- (df$Sandomierz - mean_value) / std_dev

df_cleaned <- df %>% filter(abs(z_score) <= 1)

df_cleaned <- df_cleaned %>% select(-z_score)

df_cleaned$Sandomierz <- na.approx(df_cleaned$Sandomierz)
```

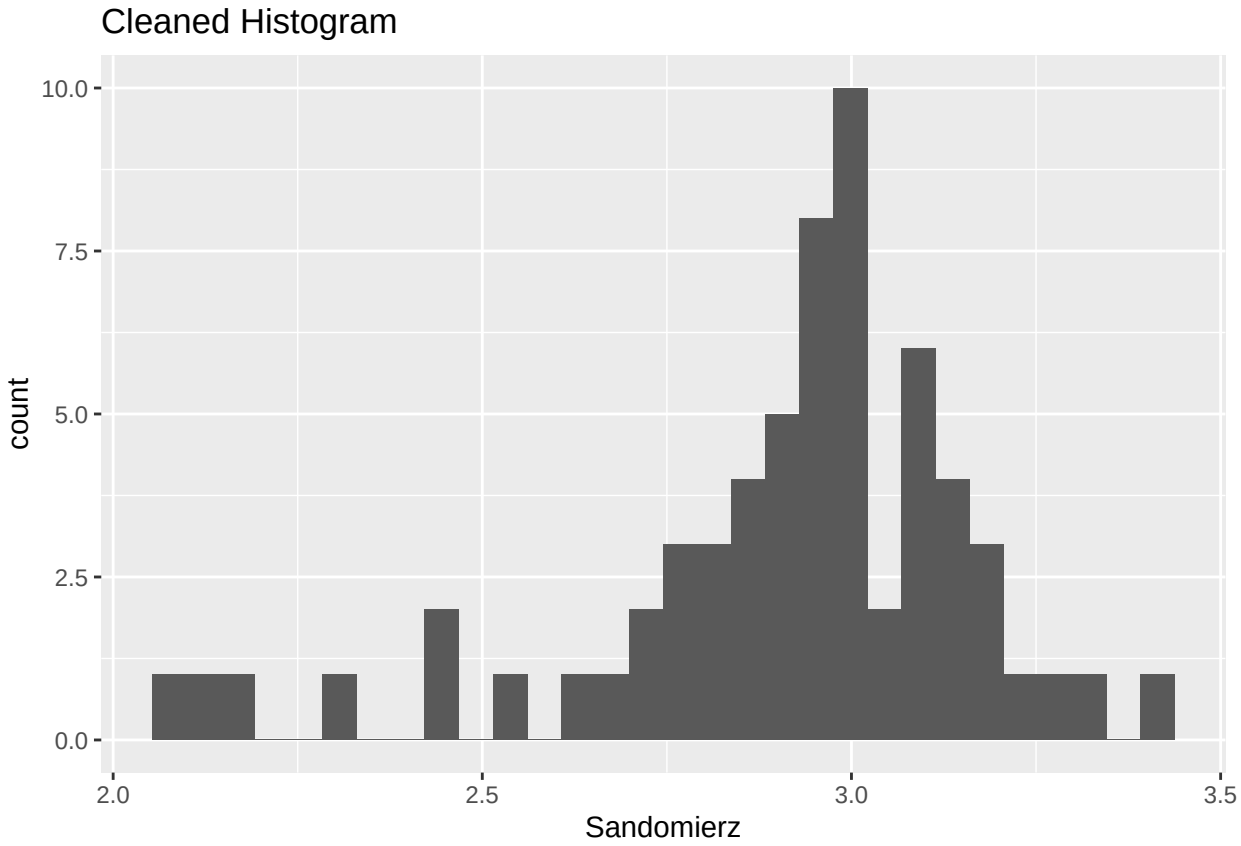
Now let's plot our cleaned data.

```
ggplot(df_cleaned, aes(x = Date, y = Sandomierz)) +
  geom_line() +
  labs(title = "Cleaned Time Series Plot")
```



This time-series looks much better and is something we can work with. We can see that around the year 2006 there is a significant drop. Let's also check our histogram.

```
ggplot(df_cleaned, aes(x = Sandomierz)) + geom_histogram(bins = 30) + labs(title = "Cleaned Histogram")
```

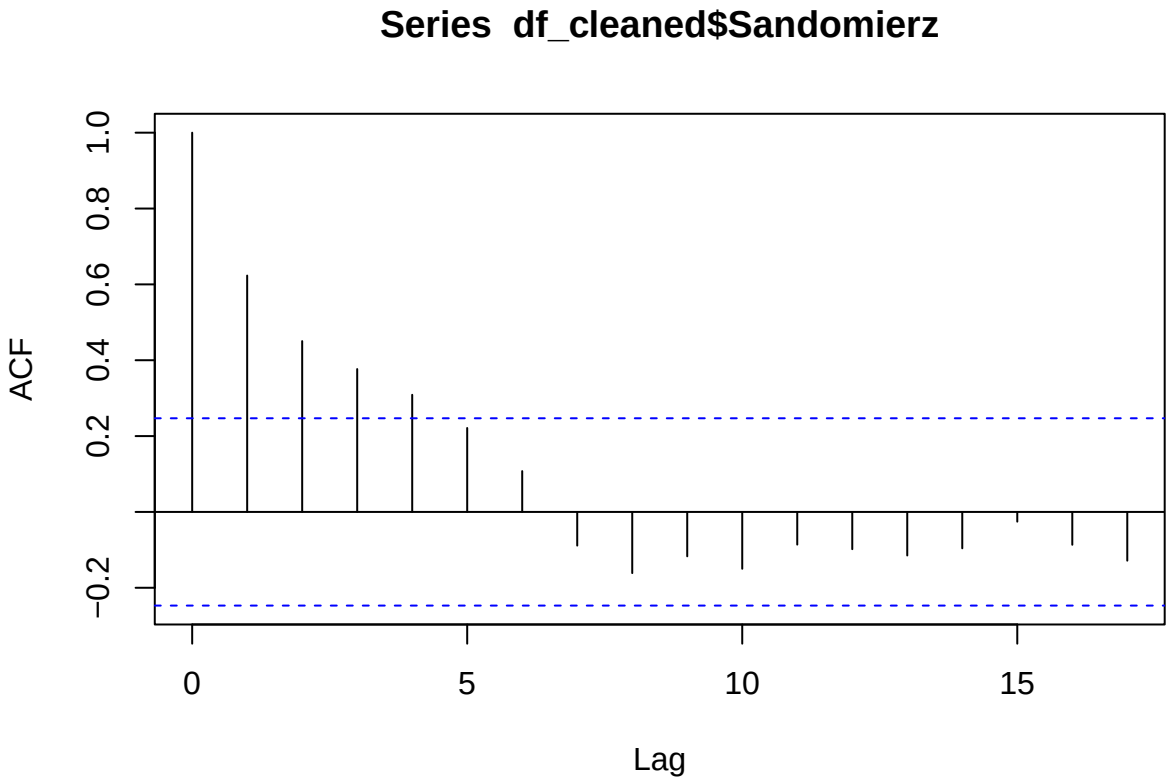


We can clearly see the lower values on the left side which correspond to our "dent" on line plot and that the majority of values are oscillating between the values 3.

Time-Series Analysis

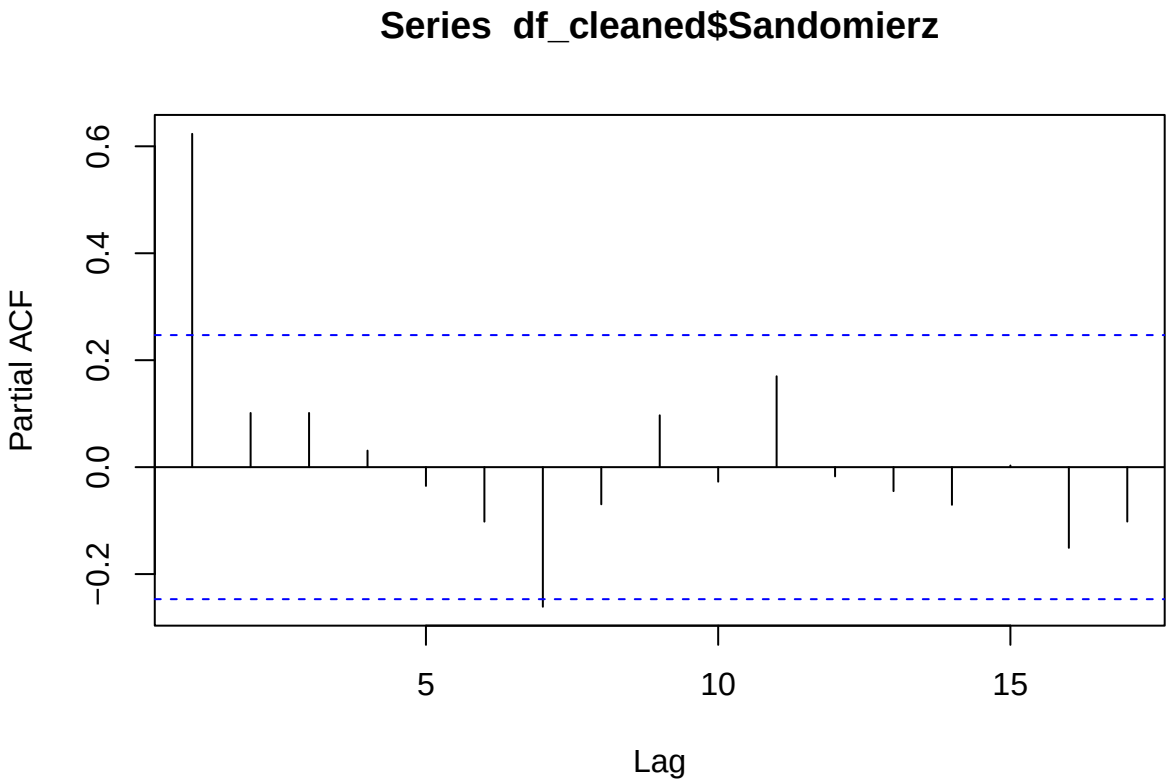
To analyze our data better we will plot the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PCF)

```
acf(df_cleaned$Sandomierz)
```



The ACF plot shows a gradual decline indication that our data is non-stationary.

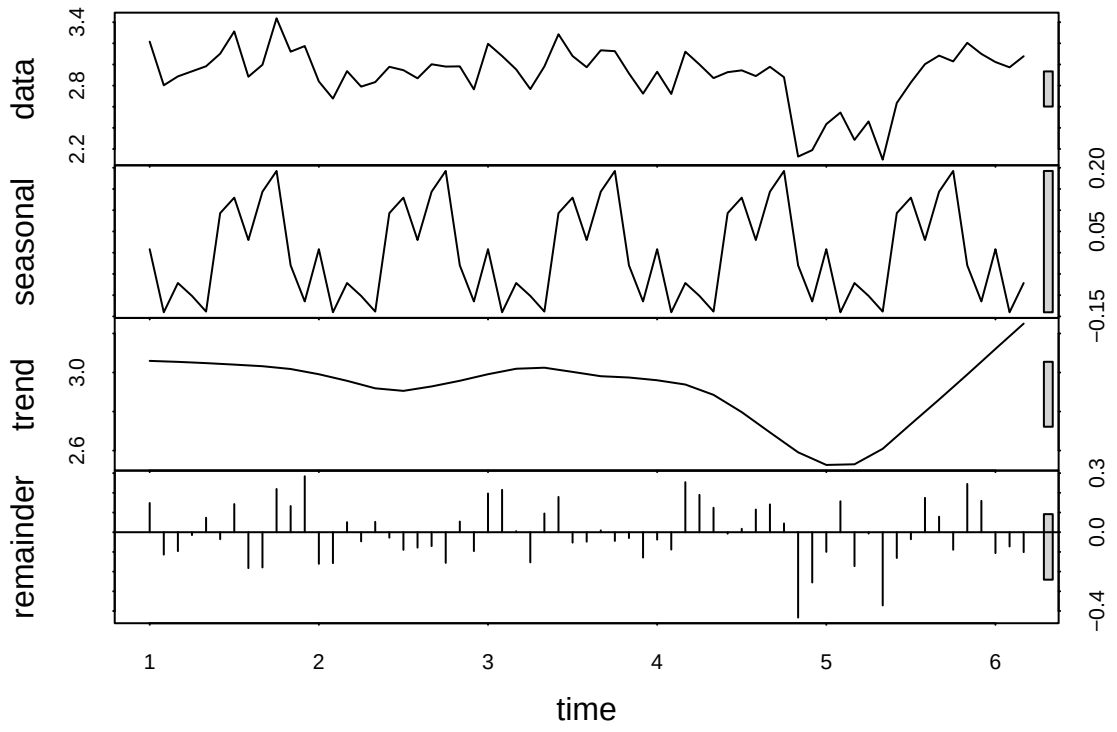
```
pacf(df_cleaned$Sandomierz)
```



Our PACF plot has a significant spike at the beginnig, which suggests an autoregressive term of order 1.

Additionally let's perform a decomposition to see the trend, seasonality and residuals.

```
decomp <- stl(ts(df_cleaned$Sandomierz, frequency = 12), s.window = "periodic")
plot(decomp)
```



From these plots we can clearly see that our data has non-linear trend and distincitve yearly seasonality.

GAM Model

Since the data we are working with is a time-series with a single curve and non-linear pattern first we will try to fit a Generalized Additive Model (GAM).

Because there is seasonality in our data we will definitely like to catch it with our model. To do so we will need to prepare our data a bit.

First we will add a numerical index for time to capture the non-linear trend in our data and month column.

```
df_gam <- data.frame(
  time = 1:nrow(df_cleaned),
  month = month(df_cleaned$Date),
  Sandomierz = df_cleaned$Sandomierz
)
```

To capture our seasonality better we will add a cyclic term.

```
df_gam$month_sin <- sin(2 * pi * df_gam$month / 12)
df_gam$month_cos <- cos(2 * pi * df_gam$month / 12)
```

Now to evaluate GAM model performance we will split our data to train and test sets. We will leave **80%** of data for training.

```
train_size <- round(0.8 * nrow(df_gam))

train <- df_gam[1:train_size,]
test <- df_gam[(train_size+1):nrow(df_gam),]
```

Now let's specify our GAM model. We will add a spline for each of our new seasonality features.

```
gam_fit <- gam(
  Sandomierz ~ s(time, k = 3) + s(month_sin, k = 5) + s(month_cos, k = 5),
  data = train
)
```

Let's calculate Mean Squared Error and Mean Absolute Error, we will interpret those value at the end.

```
predictions <- predict(gam_fit, newdata = test)

gam_mse <- mean((predictions - test$Sandomierz)^2)
gam_mae <- mean(abs(predictions - test$Sandomierz))

print(paste("MSE: ", gam_mse))
```

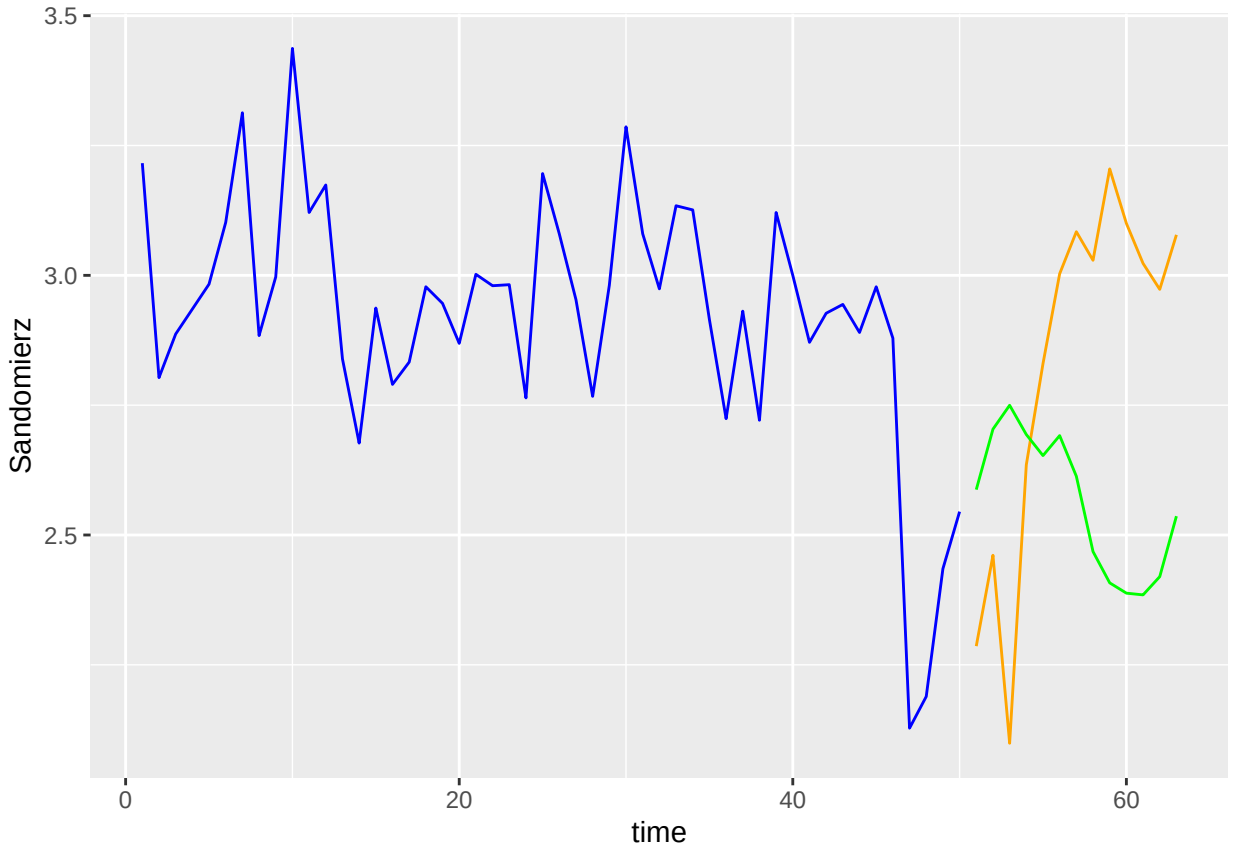
```
## [1] "MSE:  0.260843738626278"
```

```
print(paste("MAE: ", gam_mae))
```

```
## [1] "MAE:  0.462804692770592"
```

Let's see how our model performed on a plot

```
ggplot() +
  geom_line(data = train, aes(x = time, y = Sandomierz), color = "blue") +
  geom_line(data = test, aes(x = time, y = Sandomierz), color = "orange") +
  geom_line(data = test, aes(x = time, y = predictions), color = "green")
```



As we can see our GAM model fails to capture the rising trend at the end. We could further improve our

model by tuning it's parameters or performing grid search.

Comparison

At the end we will compare our GAM model with other popular time-series model Prophet.

The Prophet model, developed by Facebook, is a robust forecasting tool that excels in making predictions with time series data that exhibit strong seasonal patterns and trends.

First we need to slighly change the format of our dataframe for Prophet model.

```
df_prophet <- df_cleaned %>% rename(ds = Date, y = Sandomierz) %>% select(ds, y)
```

Now we can split it into train and test sets.

```
train_set <- df_prophet[1:train_size,]
test_set <- df_prophet[(train_size+1):nrow(df_prophet),]
```

Here we import and create the Prophet mode. By default it models daily, weekly and yearly seasonalities. Let's turn them all off.

```
prophet_model <- prophet(
  daily.seasonality = FALSE,
  weekly.seasonality = FALSE,
  yearly.seasonality = FALSE
)
```

Now we add our own yearly seasonality.

```
prophet_model <- add_seasonality(prophet_model, name = 'yearly', period = 365.25, fourier.order = 5)
```

Let's fit our model to the train set.

```
prophet_model <- fit.prophet(prophet_model, train_set)
```

Here we create a future dataframe and make predicitons.

```
future <- make_future_dataframe(prophet_model, periods = nrow(test_set), freq = "month")
```

```
forecast <- predict(prophet_model, future)
```

Let's evaluate our Prophet model performance by calulating MAE and MSE just like with GAM model.

```
actuals <- test_set$y

predicted <- forecast$yhat[(nrow(train_set) + 1):nrow(forecast)]
prophet_mse <- mean((actuals - predicted)^2)
prophet_mae <- mean(abs(actuals - predicted))

print(paste("MSE: ", prophet_mse))
```

```
## [1] "MSE:  0.270042679150592"
```

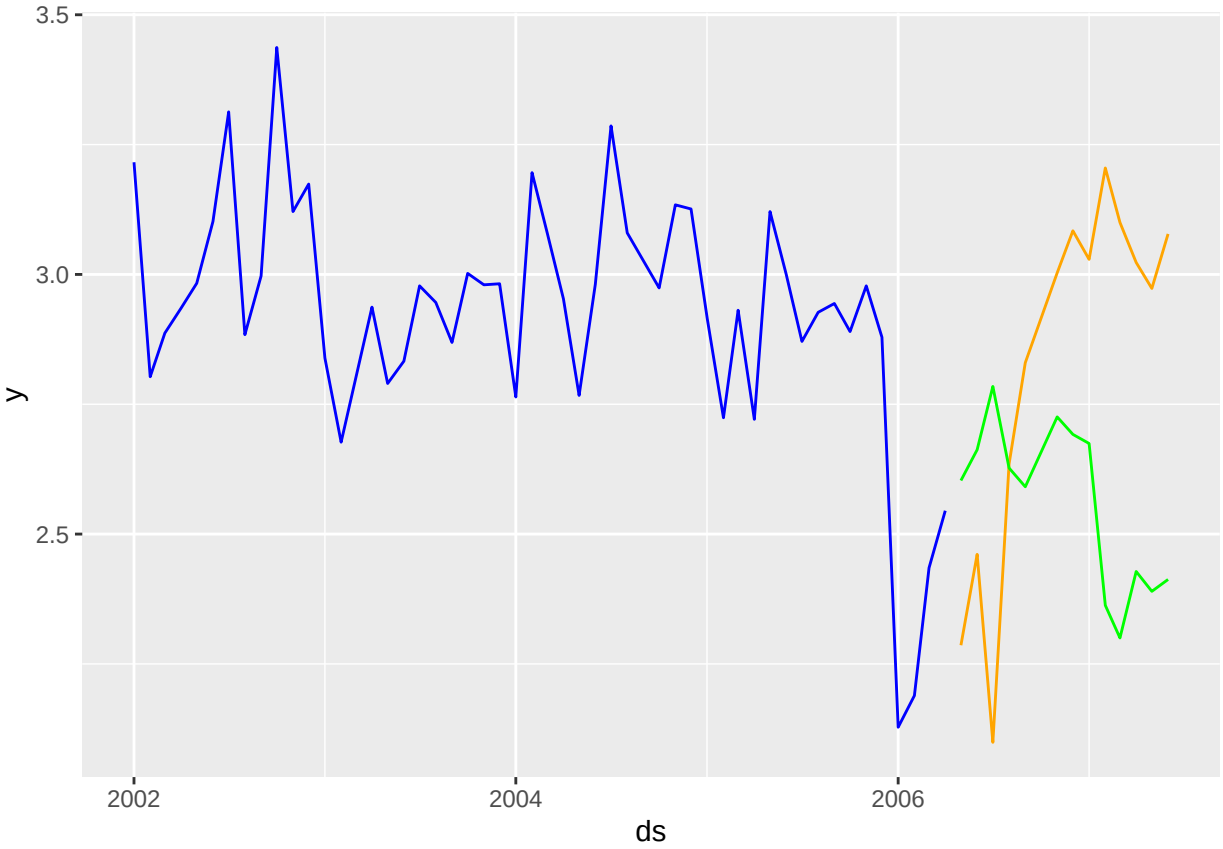
```
print(paste("MAE: ", prophet_mae))
```

```
## [1] "MAE:  0.458486031435898"
```

Finally let's see how our predictions look on a plot.

```
train_set$ds <- as.Date(train_set$ds)
test_set$ds <- as.Date(test_set$ds)
predictions_df <- data.frame(ds = test_set$ds, yhat = predicted)
```

```r
prophet_plot <- ggplot() +
  geom_line(data = train_set, aes(x = ds, y = y), color = "blue") +
  geom_line(data = test_set, aes(x = ds, y = y), color = "orange") +
  geom_line(data = predictions_df, aes(x = ds, y = yhat), color = "green")
print(prophet_plot)
```



We see that Prophet also fails to capture the rising trend at the end. However we didn't perform any parameter tuning or grid search which could improve the models performance.

Conclusions

On our dataset with wastewater from Sandomierz the **GAM** model provided comparable results to the **Prophet** model. Both of the models return promising results for the first few observations, but fail to capture the rising trend towards the end.

However, it's important to acknowledge certain limitations of this study. We did not conduct hyperparameter tuning using grid search or evaluate the models with cross-validation. Additionally, we did not explore alternative models such as SARIMA, Exponential Smoothing or machine learning methods like Gradient Boosting, which may offer potential improvements in predictive accuracy.

```r
results <- data.frame(GAM = c(gam_mse, gam_mae), Prophet = c(prophet_mse, prophet_mae), row.names = c("
print(results)
```

```
##           GAM   Prophet
## MSE 0.2608437 0.2700427
## MAE 0.4628047 0.4584860
```