Munich Cycling Data Analysis Project

Data source dl-de/by-2-0: Landeshauptstadt München – opendata.muenchen.de

Jan Posse

2023-03-19

Overview

In this project I will analyze the open data the city of Munich is providing on bicycle traffic. The city is running 6 stations counting bike traffic on bicycle lanes/ routes. Data is available on a daily level including additional information like weather and on a 15 minute interval.

My focus will be the daily data for the years 2008 to 2022. I will split the data into two time periods 2008 to 2019 (pre Covid) and 2020 to 2022 (Covid years). To ensure consistency I will run several checks and correct identified issues. I will enhance the data by splitting the date provided into smaller intervals (year, month, year-month, month-day) and adding information on school holidays and public holidays.

Focus for the analysis and data modelling are the pre Covid years 2008 to 2019. Using summary statistics and visualizations I will explore the data to identify trends and relationships within the data. The goal is to create a data model to predict traffic for a given day. For the model training I will split the data set (again). Using only part of the data set for the training and the other part to test the model.

In a final step I will use the best model to predict the bicycle traffic on an aggregated level (monthly, annually) for the years 2020 to 2022 and compare it to the actual data available. The hypothesis is that the Covid pandemic had a significant impact on bike usage. Most clearly during lockdowns, but also due to the rise of home office work in general.

The sections of this report are therefore:

1. Data load and consistency check
2. Data exploration and visualization
3. Data model set up and testing
4. Bicycle traffic 2020 to 2022
5. Summary and Outlook

The project has been implemented using R, an open-source free software for statistical computing. For better understanding I will provide snippets of the code in this report. The full code is also available via GitHub or via email request.

Data load and consistency check

Load the data

The data is available as csv file from the opendata page of the city of Munich [1]. There are 2 files per year, one containing daily information and weather data and one containing data on 15 minute intervals. As I will only look at daily data, I will only download the yearly csv files, read them into R as table and join the yearly tables into two summary tables Daily (basis for my analysis and data modelling) and Daily_2020_2022 (for later comparison).

As the load for each year follows the same process, I abbreviated the code below showing the load of the first year only and then the union for all years.

```
# The url for each year is
url_2008_Daily <- "https://opendata.muenchen.de/dataset/022a11ff-4dcb-4f03-b7dd-
a6c94a094587/resource/53ef8c4b-d411-477f-9cf3-b044a4c1aaaa/download/
rad_2008_tage_export_28_02_23_r.csv"

# To ensure proper data load I load each csv in a separate table
Daily2008 <- readr::read_csv(url_2008_Daily)

# I will union the tables using rbind and perform two checks.
# 1. Is the table structure the same across the tables?
# 2. Is the numbers of rows across the tables the same?

DataCheck <- c()
RowCount <- 0
for(i in 2008:2019){
  tblName <- paste0("Daily", i)
  DataCheck[i-2007] <- identical(str(Daily2008), str(get({{tblName}})))
  RowCount <- RowCount + nrow(get({{tblName}}))
}

Daily <- data.frame()
for(i in 2008:2019){
  tblName <- paste0("Daily", i)
  Daily <- rbind(Daily, (get({{tblName}})))
}

sum(DataCheck) == length(DataCheck)
RowCount == nrow(Daily)

# Both checks come back as TRUE
# I then combine the daily data for the years 2020 to 2022 with the same checks

DataCheck <- c()
RowCount <- 0
for(i in 2020:2022){
  tblName <- paste0("Daily", i)
  DataCheck[i-2019] <- identical(str(Daily2008), str(get({{tblName}})))
  RowCount <- RowCount + nrow(get({{tblName}}))
}

Daily_2020_2022 <- data.frame()
for(i in 2020:2022){
  tblName <- paste0("Daily", i)
  Daily_2020_2022 <- rbind(Daily_2020_2022, (get({{tblName}})))
}

sum(DataCheck) == length(DataCheck)
RowCount == nrow(Daily_2020_2022)

# Again both checks are successful
```

Check consistency

Data structure

Looking at the data we can see the following structure.

Table 1: First rows of the Daily data set

datum	uhrzeit_start	uhrzeit_ende	zaehlstelle	richtung_1	richtung_2	gesamt	min.temp	max.temp	niederschlag	bewoelkung	sonnenstunden
2008-06-01	00:00:00	23:59:00	Arnulf	645	22	667	12.5	26.7	0	30	13.9
2008-06-02	00:00:00	23:59:00	Arnulf	1070	47	1117	15	27.9	0.6	44	12.1
2008-06-03	00:00:00	23:59:00	Arnulf	1240	39	1279	14.6	21.9	0.2	88	3.2
2008-06-04	00:00:00	23:59:00	Arnulf	715	43	758	14.8	21.9	5.1	91	1.4
2008-06-05	00:00:00	23:59:00	Arnulf	569	37	606	14	20.4	14.2	91	0.6
2008-06-06	00:00:00	23:59:00	Arnulf	919	44	963	13.6	19.9	10	81	1.7

The following columns are present:

Table 2: Columns in the data set

ColumnName	Type	Description
datum	Date	Date of the measurement
uhrzeit_start	'hms' num	Time of the start of the measurement
uhrzeit_ende	'hms' num	Time of the end of the measurement
zaehlstelle	chr	Name of the position of the counting unit
richtung_1	num	Number of cyclist in the direction one
richtung_2	num	Number of cyclist in the direction two
gesamt	num	Total number of cyclists
min.temp	chr	Minimum temperature on the day in degree Celsius
max.temp	chr	Maximum temperature on the day in degree Celsius
niederschlag	chr	Precipitation on the day measured in mm
bewoelkung	num	Cloud cover on the day in %
sonnenstunden	chr	Sunshine hours on the day

Data types

We see that some of the data has not been read in properly, e.g. the temperature was converted to a character type, but should be numeric. Trying to convert these columns into numeric we generate NAs.

We can investigate the NAs using the following code.

```
Daily$min.temp[is.na(as.numeric(Daily$min.temp))]
Daily$max.temp[is.na(as.numeric(Daily$max.temp))]
Daily$niederschlag[is.na(as.numeric(Daily$niederschlag))]
Daily$sonnenstunden[is.na(as.numeric(Daily$sonnenstunden))]
```

It becomes evident that not all numbers are formatted correctly. Some of the numbers use a comma as decimal separator, which is in fact the standard in Germany and therefore a likely error. We can replace it with a point using a text search pattern or regex as follows.

The same issue applies for the table Daily_2020_2022, but only in niederschlag and sonnenstunden. I will correct the data here as well.

```
# Note that there might be negative temperatures (one or none '-'), one or two
# digits before the decimal and one after
Daily$min.temp <- Daily$min.temp %>%
    str_replace("^(-?\\d*),(\\d{1})", "\\1.\\2")
```

```
Daily$max.temp <- Daily$max.temp %>%
    str_replace("^(-?\\d*),(\\d{1})", "\\1.\\2")

# In very rare circumstances precipitation can be above 100 mm in one day and
# there is no negative precipitation. So I will have to adjust the regex for
# this.
Daily$niederschlag <- Daily$niederschlag %>%
    str_replace("^(\\d+),(\\d{1})", "\\1.\\2")
Daily_2020_2022$niederschlag <- Daily_2020_2022$niederschlag %>%
    str_replace("^(\\d+),(\\d{1})", "\\1.\\2")

# The longest day in Munich (summer solstice) is less than 20 hours and there
# are no negative sunshine hours. So the regex is again slightly different.
Daily$sonnenstunden <- Daily$sonnenstunden %>%
    str_replace("^(1?\\d),(\\d{1})", "\\1.\\2")
Daily_2020_2022$sonnenstunden <- Daily_2020_2022$sonnenstunden %>%
    str_replace("^(1?\\d),(\\d{1})", "\\1.\\2")
```

After this quick replacement we can convert the columns to numeric without any NAs.

```
Daily$min.temp <- as.numeric(Daily$min.temp)
Daily$max.temp <- as.numeric(Daily$max.temp)
Daily$niederschlag <- as.numeric(Daily$niederschlag)
Daily$sonnenstunden <- as.numeric(Daily$sonnenstunden)
Daily_2020_2022$niederschlag <- as.numeric(Daily_2020_2022$niederschlag)
Daily_2020_2022$sonnenstunden <- as.numeric(Daily_2020_2022$sonnenstunden)
```

For better usage, I will also change the location data to factor as there are only 6 locations in the data set (and in real life in Munich).

```
Daily$zaehlstelle <- as.factor(Daily$zaehlstelle)
Daily_2020_2022$zaehlstelle <- as.factor(Daily_2020_2022$zaehlstelle)
```

Summary and inconsistent data

Looking at the summary of the columns. I notice a few items that look odd / inconsistent.

Table 3: Summary of columns in the data set 01

datum	uhrzeit_start	uhrzeit_ende	zaehlstelle	richtung_1	richtung_2	gesamt	min.temp	max.temp	niederschlag	bewoelkung	sonnenstunden
Min. :2008-06-01	Length:23105	Length:23105	Arnulf :4200	Min. : 0.0	Min. : 0.0	Min. : 0	Min. :-23.100	Min. : -9.90	Min. : 0.000	Min. : 0.0	Min. : 0.000
1st Qu.:2012-01-29	Class1:hms	Class1:hms	Erhardt :3106	1st Qu.: 160.0	1st Qu.: 69.0	1st Qu.: 316	1st Qu.: 0.200	1st Qu.: 7.40	1st Qu.: 0.000	1st Qu.: 53.0	1st Qu.: 0.400
Median :2014-09-18	Class2:difftime	Class2:difftime	Hirsch :4231	Median : 520.0	Median : 278.0	Median : 890	Median : 5.600	Median : 15.30	Median : 0.000	Median : 79.0	Median : 4.200
Mean :2014-08-12	Mode :numeric	Mode :numeric	Kreuther :4231	Mean : 786.7	Mean : 613.2	Mean : 1400	Mean : 6.285	Mean : 16.53	Mean : 2.246	Mean : 69.7	Mean : 5.797
3rd Qu.:2017-05-08	NA	NA	Margareten:3106	3rd Qu.: 1157.0	3rd Qu.: 910.0	3rd Qu.: 1984	3rd Qu.: 11.100	3rd Qu.: 22.10	3rd Qu.: 2.000	3rd Qu.: 94.0	3rd Qu.: 9.000
Max. :2019-12-31	NA	NA	Olympia :4231	Max. :13861.0	Max. :14107.0	Max. :15009	Max. :191.000	Max. :352.00	Max. :71.200	Max. :100.0	Max. :156.000
NA	NA	NA	NA	NA	NA	NA	NA's :186	NA's :186	NA's :186	NA	NA's :186

We should investigate further:

1. There are days with no total count (gesamt = 0)
2. There are temperatures (min and max) above 100 degree celsius as
3. There are maximum daily sunshine hours of 156
4. There are 186 rows with missing weather data showing NAs (all weather data except cloud cover)

Days with no total count We can look at the days without any cyclist counted by month and location using the following code.

```
NullCounts <- Daily %>%
    filter(gesamt == 0) %>%
    group_by(year(datum), month(datum), zaehlstelle) %>%
    summarize(Count = n())
```

We can see there is a total of 1,544 rows with no cyclist counted on one day. However, for the majority of cases, we can see that there are several occurrences of no count in one month for that particular location. To be precise 1,505 cases fall into this category, when we set the limit at 5 or more occurrences of no count in one month.

```
sum(NullCounts$Count)
sum(NullCounts %>%
    filter(Count >= 5) %>%
    pull(Count))
```

As both the Median and the Mean are significantly higher than 0, I assume there was a technical fault or a local construction site in the location and therefore I will exclude this data.

There are also cases that seem to be real 0 counts. Looking at the Location Hirsch (which we will see later has the lowest mean and median of all locations) we can spot days with 0 counts that fall into winter and happen to be on very cold (and even rainy / snowy days). These should remain in the data set.

Table 4: Selected rows with no cyclist counts

datum	uhrzeit_start	uhrzeit_ende	zaehlstelle	richtung_1	richtung_2	gesamt	min.temp	max.temp	niederschlag	bewoelkung	sonnenstunden
2010-01-04	00:00:00	23:59:00	Hirsch	0	0	0	-8.6	-0.4	0.0	39	7.6
2010-02-01	00:00:00	23:59:00	Hirsch	0	0	0	-3.6	-1.1	0.3	86	4.1
2010-12-30	00:00:00	23:59:00	Hirsch	0	0	0	-7.4	-3.4	0.0	95	0.0

So I will remove rows that seem to show an external influence (construction, defect etc.), with defining an external influence at 5 or more occurrences in one month. Then I will remove all other zero count rows that do not fall into winter (November to March). This leaves 20 rows with real 0 counts in our data set.

```
# I will flag the rows I have to delete in the following table
temp <- Daily %>%
    filter(gesamt == 0) %>%
    group_by(year(datum), month(datum), zaehlstelle) %>%
    mutate(Delete = ifelse(n() >= 5 | `month(datum)` %in% c(4:10), 1, 0)) %>%
    ungroup()
temp <- temp %>%
    select(datum, zaehlstelle, Delete)

# I will do a join with the original table and filter the false 0 rows out and
# remove the Delete column
Daily <- Daily %>%
    left_join(temp, by = c("datum", "zaehlstelle"))
Daily <- Daily %>%
    filter(Delete != 1 | is.na(Delete))
Daily <- Daily %>%
    select(-Delete)
```

Temperature outliers When we filter the data for the temperature outliers, we can see 156 cases with odd temperatures. However, they fall only on 26 days (all in June 2019) and relate to all 6 locations (6 x 26 = 156).

```
Daily %>% filter(min.temp > 50) %>% summarize(Count=n())
Daily %>% filter(min.temp > 50) %>% group_by(datum) %>% summarize(Count=n())
```

A look in the csv files reveals that the temperature for these days was wrongly formatted with a comma as decimal separator. Somehow during the read of the csv file this separator was lost. It seems all of June 2019 data is impacted, however, only temperatures, that have decimals in the first place (e.g. 12.7 degrees not 12 degrees).

I can correct this data by dividing by 10.

```
Daily <- Daily %>% mutate(min.temp = ifelse(min.temp>50, min.temp/10, min.temp))
Daily <- Daily %>% mutate(max.temp = ifelse(max.temp>50, max.temp/10, max.temp))
```

Sunshine hours outliers Looking at the outliers of sunshine hours - we discover that these are also all within June 2019. It seems to be the same error as with the temperature.

Again I can correct this data by dividing with 10 (as the minimun number of sunshine hours in that month according to the csv file was 1.8 the correction is easy).

Verification June 2019 As there were two errors when reading in June 2019 data, I went back to the base file and checked the other columns as well. The cyclists count columns are fine (no decimal separator). The same goes for the cloud coverage. Precipitation does have a decimal separator. However, the data was read in correctly - so there is no need for a correction.

Inconsistencies 2020-2022 data Looking into the 2020 to 2022 data we can identify the same issue with a wrong decimal separator in two months. Unfortunately one being December with very small actual temperatures, the correction formula was slightly more complex/ manual.

Weather NAs In the summary we can spot 186 NAs in the weather data. Using is.na we can identify that the month of December in the year 2018 is missing weather data for min and max temperature, precipitation and sunshine (31 days x 6 locations = 186 NAs).

As it is unclear what the precise source of the weather data is, the data for December 2018 cannot be used for modeling. I will hence remove the month from the data set.

Logic checks

Two final logic checks on the date.

Start and end time

As we know the data is on a daily basis start and end time should be the same for all rows.

```
Daily %>%
    group_by(as.factor(uhrzeit_start), as.factor(uhrzeit_ende)) %>%
    summarize(Count = n())
```

Table 5: Number of start times

Startzeit	Endzeit	Count
00:00:00	00:00:23.59	6539
00:00:00	23:59:00.00	14856

We see that there are two end times, I already noticed that, when looking at the CSV files. However it does look like another separator issue and since we know it is daily data, I will ignore start and end date in the analysis, so no need for correcting this.

Days in the year

We expect for each year 365 days (or 366 days) for each location that is running. Looking at the number of rows per year and location (each row represents one day) we can see that this holds mostly true. Some of the smaller differences can be explained as we already removed some rows, where we assumed the counting unit was not working. Also we removed a full month out of 2018 due to missing weather data. That leaves only one really odd data point - the location Kreuther in 2013, which can be explained by external effects.

Table 6: Number of rows per year and location

	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Arnulf	214	348	327	355	356	360	365	348	366	365	334	334
Erhardt	0	0	0	160	366	365	365	365	366	365	334	365
Hirsch	0	25	365	365	366	355	365	365	366	365	334	365
Kreuther	195	320	330	352	366	100	365	365	366	365	334	361
Margareten	0	0	0	177	366	340	365	365	366	365	334	365
Olympia	0	155	365	365	366	365	365	365	364	365	334	365

External effects

From the open data page we also get an information on two external effects.

1. At the location Arnulf

a. A construction site is influencing the numbers since early 2021
b. The counting unit didn't work from May to Juli 2019

2. At the location Kreuther

a. The unit has been upgraded in April 2020 doubling the width of the counting loop in the ground.
b. The counting unit didn't work from May to November 2013

For both locations the first effect listed has no impact on my base data (I am looking only at data till 2019). Nonetheless I will exclude this data for my planned comparison. The second effect listed for both locations (units not working) should have already been excluded by my data cleaning.

Enriching the data

For my analysis I would like to add some more predictors based on the date available in the data set. I start by adding school holidays and public holidays, as I expect they will have an impact on the number of cyclists. Then I will add the weekday - in Germany the office/ manufacturing working and the school week is from Monday to Friday, on Saturday shops are still open and on Sunday only restaurants/ hospitals/ police etc. are open. This will very likely have an impact as well. Finally for analysis purposes I will spread out the date into three columns year, month, day.

School holidays Unfortunately I couldn't find a curated open data set for school holidays in Germany. However I found the webpage https://kalender-365.de that list school holidays for each year. The dates for each year are on separate pages that follow a naming convention like https://kalender-365.de/schulferien.php?yy=2007. The data is then stored in the same table and structure, so that I can use data scraping, a process that extracts data presented on a web page for further processing, to collect the holidays.

Public holidays Public holidays fall into two groups, fixed holidays and flexible holidays. Fixed holidays always fall on the same day, flagging these in my data is relatively easy to set up. Flexible public holidays like Easter, that do fall on a different day every year, would be harder to flag. However, I can use the R function 'holiday' to determine the date in each year.

Day of the week The day of the week can be easily added with the function weekday in the package lubridate.

```
Daily <- Daily %>% mutate(weekday = wday(datum, label=TRUE))
Daily_2020_2022 <- Daily_2020_2022 %>% mutate(weekday = wday(datum, label=TRUE))
```

In a final step, I will split out the date to usable chunks year, month, day, yearmonth and monthday.

Now I am ready to explore the data.

Data exploration and visualization

Data summary

The cleaned and enriched data set now contains 21,395 rows of data. Each row representing the observation on one location on one day.

Location position

The city does provide more information on the position of the counting units, including the physical location and the cardinal direction of the two directions counted (direction one and direction two). A map of the locations and the direction one shows no common orientation.

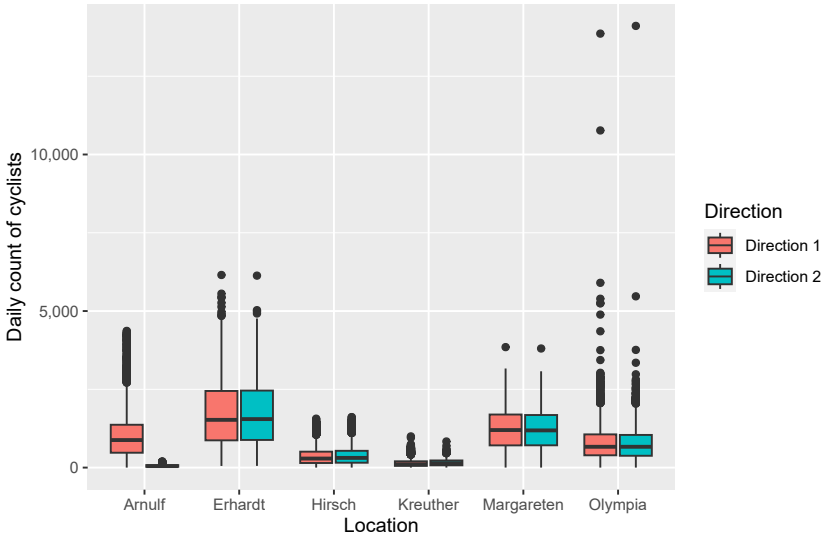
Map 1: Position of the cyclist counting units [2]

The common orientation I expected would have been into and out of the city center. While we could probably change direction one for most locations to fit this orientation, it is not clear which direction is into the city center for the location Erhardt.

Cyclist count by direction

The counting units provide data by direction. We can compare them using a boxplot.

Graph 1: Daily cyclist count by location and direction

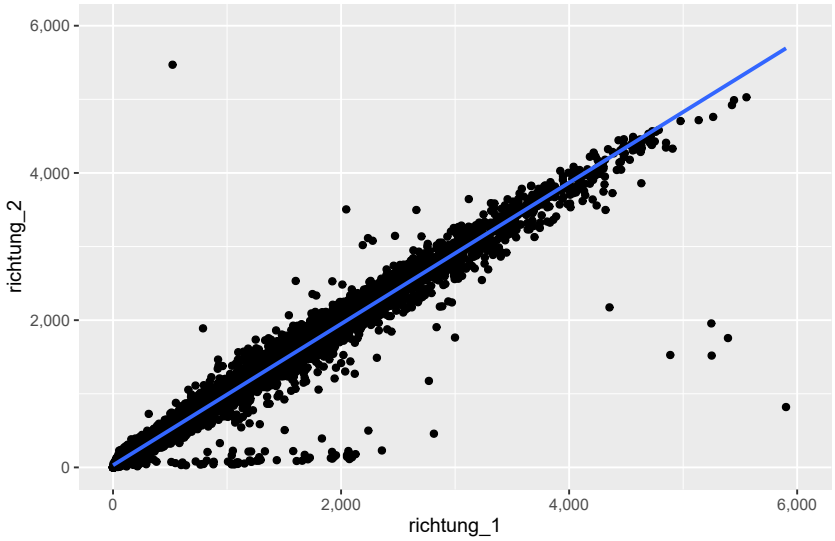


We can see that the mean, the 25th and the 75th percentile for each direction are on a similar level when grouped by location. This makes sense as we would expect cyclist travelling in one direction to return on the same day in the other direction.

The one exception is the location Arnulf. A bit of internet research shows [3] that all counting locations are based on cycle path with separate lanes per direction right next to each other, except Arnulf, which is at a cycle path that is going just in one direction. The data shown for Arnulf in Direction 2 are cyclists using the cycle path in the wrong direction. The huge difference to Direction 1 shows that German cyclist are (at least here) following the law most of the time.

Excluding Arnulf and plotting both directions in one graph, we can see a strong correlation.

Graph 2: Daily cyclist count in each direction



As I do not want to analyze the flow of cyclist in and out of the city and given the strong correlation between the cyclist count by direction, I will focus on total cyclists counted (column ‘gesamt’).

In graph 1, I also noticed that each location has a significant number of outliers - especially the location Olympia. Let’s have a quick look at the top 3 cyclist counts (all at the location Olympia).

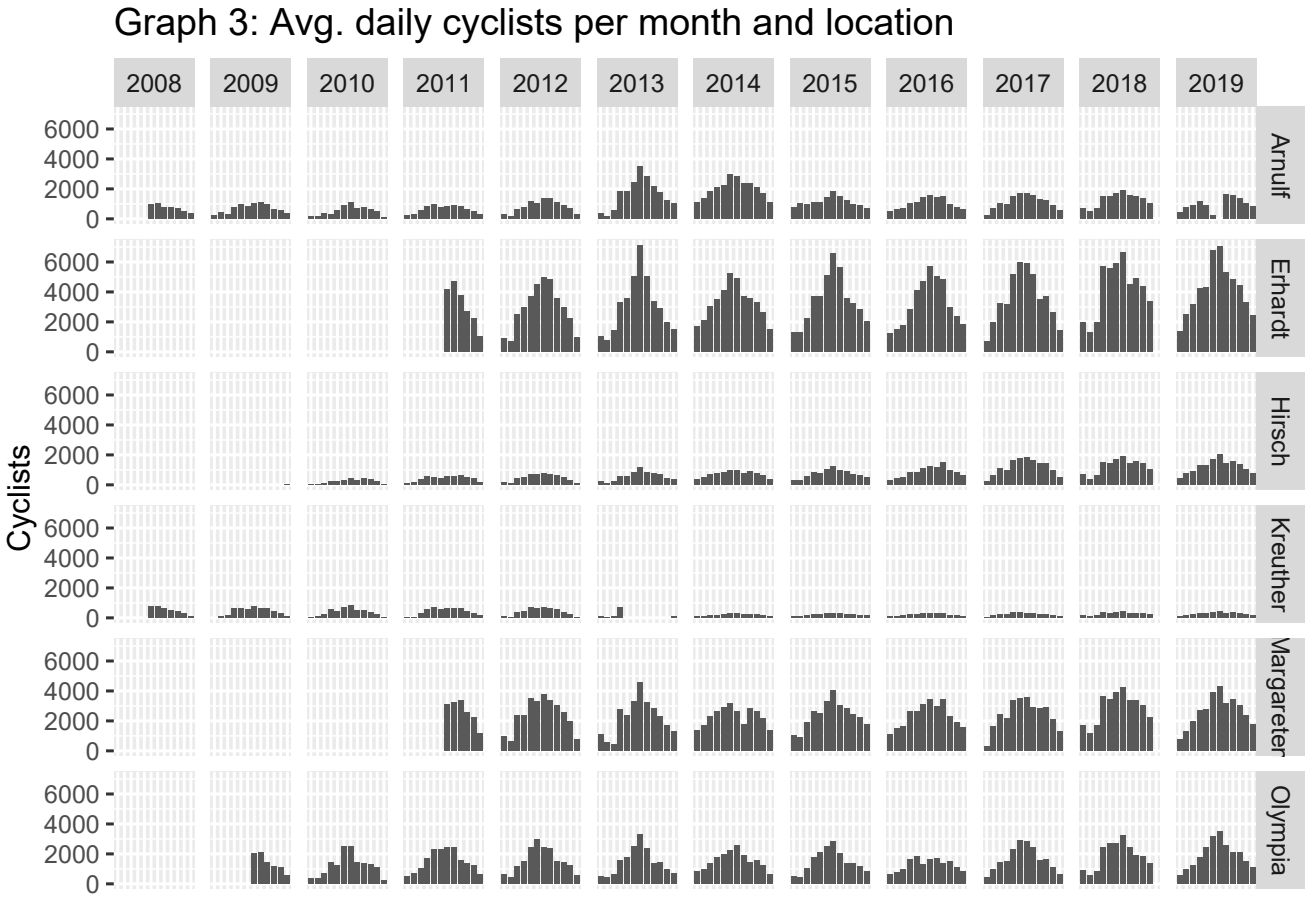
Table 7: Outliers in detail

datum	zaehlstelle	Count	subgroup
2012-06-09	Olympia	14107	Direction 2
2010-06-26	Olympia	13861	Direction 1
2010-06-27	Olympia	10768	Direction 1

We can see that top count is from the 09th of June 2012. A research on the webpage of the Olympic Park shows that on that day here was a 24 hours mountainbike race [4]. The second and third highest number were counted on the 26th and 27th of June 2010. Again looking at the webpage of the Olympic Park, we can see that on these two days there was the same event, a 24 hours mountainbike race [5]. I assume in 2012 the route was passing the counting location in one direction and in 2012 in both directions.

Monthly pattern

We can split out the total cyclist count by location and year for a visual review.

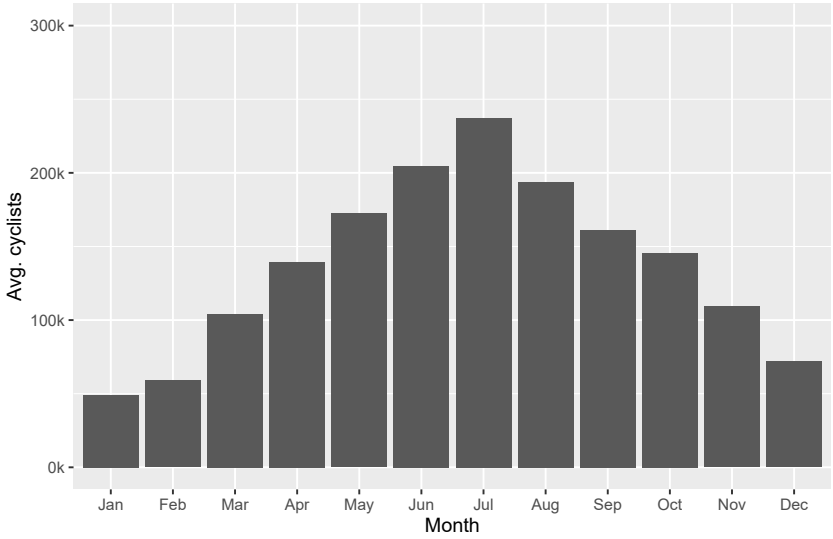


You can already detect a pattern in the monthly distribution. Also, as already noted, it is evident that we do not have data for all locations for all years or months. When summarizing this pattern - looking for an annual pattern it makes sense to look only at locations, which have values for every month in a year - i will create a logical fullyear column to filter for these locations.

```
fullyear <- Daily %>%
    group_by(zaehlstelle, year) %>%
    summarize(n = n_distinct(month)) %>%
    mutate(fullyear = if_else(n == 12, TRUE, FALSE)) %>%
    select(-n)
Daily <- left_join(Daily, fullyear, by = c("zaehlstelle", "year"))
```

Now I can look at the monthly cyclist in total (looking only at locations with full year data available).

Graph 4: Average cyclists per month and location (full years only)

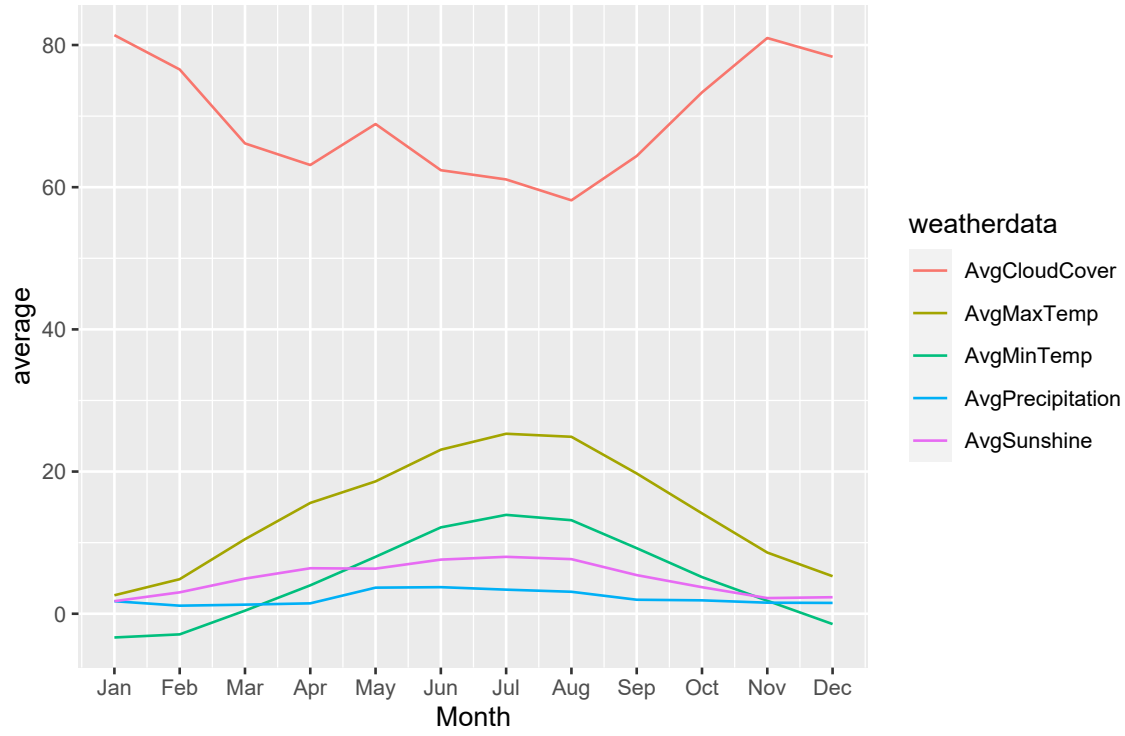


We can see there is a strong seasonal pattern with most cyclists in summer. This can probably be attributed to the weather conditions.

Weather pattern and impact

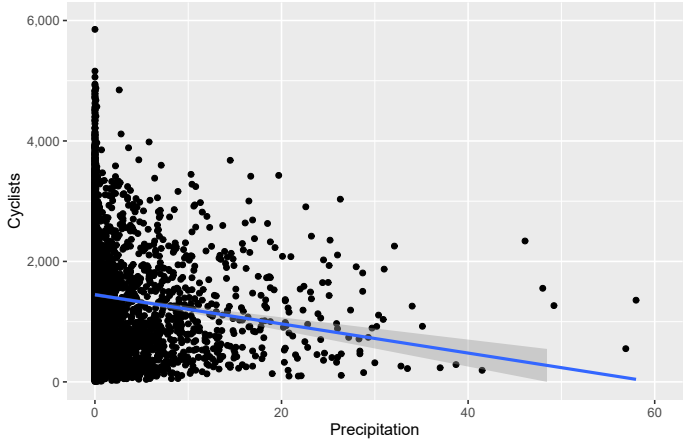
A look at the available weather data shows a similar pattern to the monthly pattern in graph 4.

Graph 5: Average monthly weather data for Munich

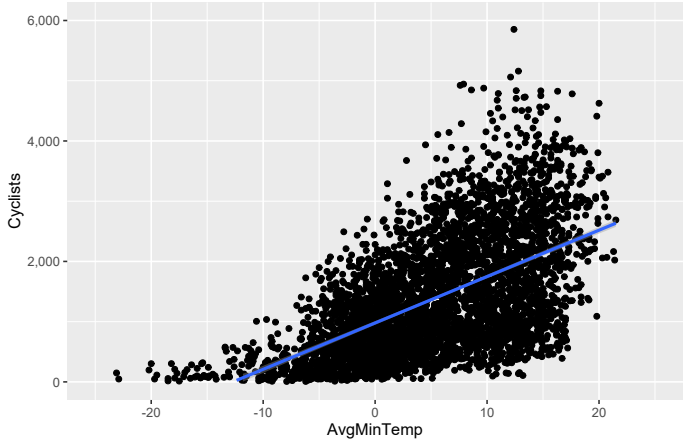


There seems to be a relationship between good weather (high temperatures, low precipitation, long sunshine) and the number of people riding their bike. Lets investigate the individual weather inputs and their impact on the average number of cyclist per day and location.

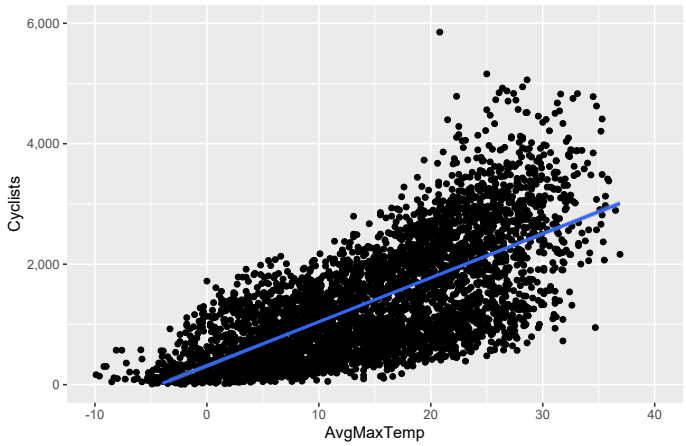
Graph 6: Avg. cyclists and Precipitation



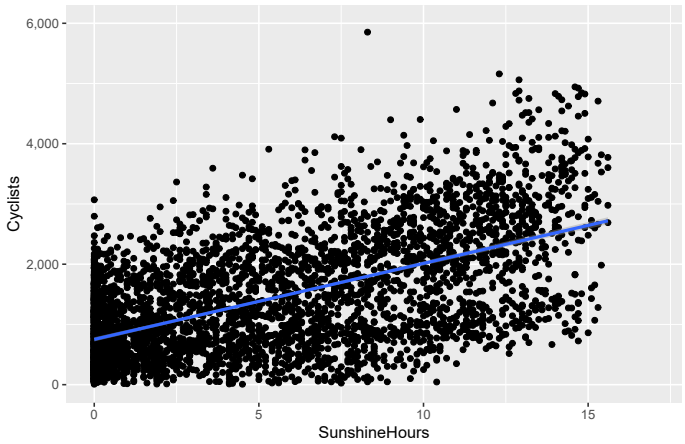
Graph 7: Avg. cyclists and Minimum Temperature



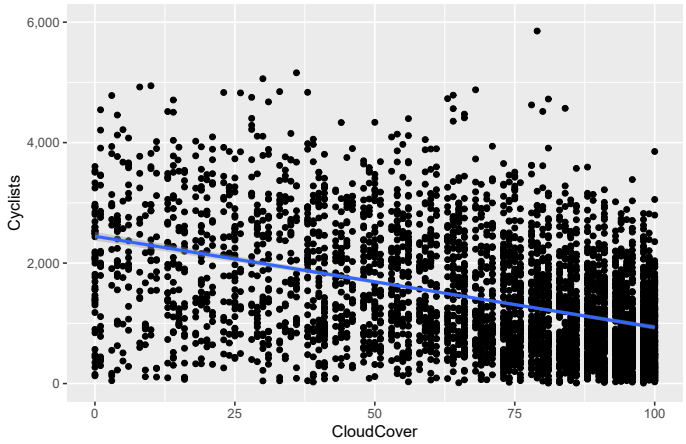
Graph 8: Avg. cyclists and Maximum Temperature



Graph 9: Avg. cyclists and Sunshine Hours



Graph 10: Avg. cyclists and Cloud Coverage



I added a regression line for orientation. However, focusing on the distribution (the dots), you can notice a significant spread in all graphs. I will focus on Temperature as it seems to be a more relevant factor.

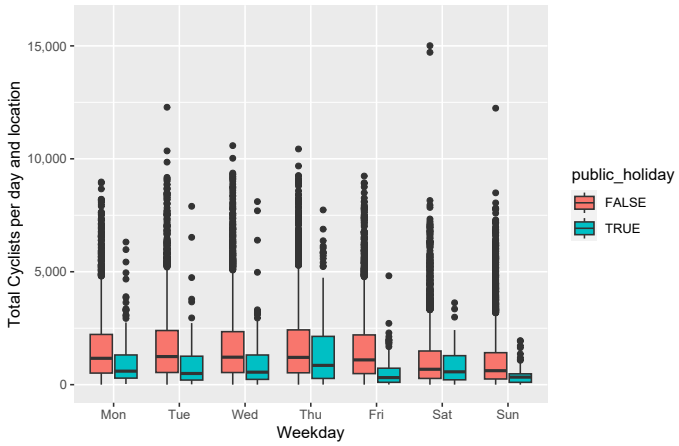
Weekday and holiday impact

I also expect an impact on the number of cyclists by holiday. And looking at the following graph, we can see that public holidays reduce cycling traffic for a given weekday. Two specific observations:

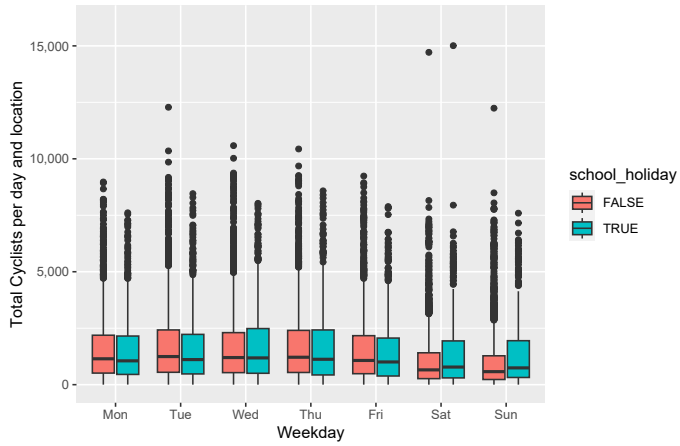
1. The impact on Thursdays seems to be the smallest. This could be due to Ascension Day and Corpus Christi. Both public holidays always fall on a Thursday and being in late spring (or even early summer - if you use the meteorological definition) usually have good weather. In addition Ascension Day is also known as fathers day, which is often combined with cycling day trips [6].
2. The impact on Sundays seems to be the biggest. This can probably be attributed to the timing of the holidays. As only fixed public holidays can fall on a Sunday (as Sunday's are in Germany by definition non working days, observed holidays like Easter Sunday are not considered as public holidays), this leaves only 2 public holidays in (meteorological) spring and summer that could fall on a Sunday, with 4 out of the remaining 6 in winter.

School holidays, the graph on the right, however, have only a limited impact during the working week (Monday to Friday), but increase cycling traffic on the weekends. This again might be correlation but not causation, as the largest school holiday block falls into summer and thus correlates with the months with highest number of cyclists overall.

Graph 11: Total Cyclists by Weekday and Public holiday



Graph 12: Total Cyclists by Weekday and School holiday



Data model set up and testing

Selection of predictors and data setup

Based on the data exploration and visualization, I will focus on the following data:

1. Basic location information (zaehlstelle)
2. Weather data - even precipitation (min.temp, max.temp, niederschlag, bewoelkung, sonnenstunden)
3. Calendar data (year, month, weekday, yearmonth, monthday, school_holiday, public_holiday)
4. And total cyclist count per day (gesamt)

With the last being the outcome and the other the predictors. I will also exclude one full year (I picked by random the year 2016) .

For training and testing the data models, I split the data set in a training and a test data set.

```
set.seed(1982)
test_index <- createDataPartition(temp$gesamt, time = 1, p = 0.9, list = FALSE)

Daily_train <- temp[test_index, ]
Daily_test <- temp[-test_index, ]
```

Accuracy measures

For measuring the accuracy of my data model I will use the Residual Mean Square Error (RMSE) and as it is easier to understand also the Mean Absolute Percentage Error (MAPE).

When we define $y_{d,l}$ as the actual number of cyclist for a given day (d) and a specific location (l) and our prediction in the same manner as $\hat{y}_{d,l}$ the RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{N} * \sum_{d,l}(y_{d,l} - \hat{y}_{d,l})^2}$$

The MAPE is putting the error in relation to the actual value thus creating an easy to understand percentage value of the average error the model creates. Using the same definitions as above the MAPE is defined as[7]:

$$MAPE = \frac{100\%}{N} \sum_{d,l} \left| \frac{y_{d,l} - \hat{y}_{d,l}}{y_{d,l}} \right|$$

As I will be using both several times, I create a function in R to ease the computation.

```
    RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
    }

    MAPE <- function(true_ratings, predicted_ratings){
      mean(abs((true_ratings-predicted_ratings)/true_ratings))*100
    }
```

We noticed earlier that 20 actual values are 0 (and we assume them to be true 0), as we cannot divide by 0. This will create a problem for the MAPE. Hence I will exclude them when calculating the MAPE. There are alternatives like Symmetric mean absolute percentage error, which however, have their own shortcomings. Excluding 0.09% of the data seems easier, especially since I will just use the MAPE to better understand the error.

```
Daily_test_MAPE <- Daily_test %>% filter(gesamt!=0)
```

Mean

In a first and very basic approach, we assume the same number of cyclists come across each location on each day. Differences in the number of cyclists are explained by random variation. Our model will now look like this.

$$Y_{d,l} = \mu + \epsilon_{d,l}$$

We set this one rating as the mean of the training data set, as the mean is the least square estimate of μ the actual number of cyclists per day and location.

```
avg <- mean(Daily_train$gesamt)
basic_rmse <- RMSE(Daily_test$gesamt, avg)
basic_MAPE <- MAPE(Daily_test_MAPE$gesamt, avg)
```

The mean of our data set is 1,510 cyclist per day and location. The results look like this.

Table 8: Data Model results 01

Method	RMSE	MAPE
Mean	1462.112	642.1363

We see that both the RMSE and the MAPE show a huge error in our model. We can probably do a lot better, by adjusting for the location.

Location effect

We can accommodate for the effect that that some locations have significantly higher cyclist counts than others. We will call this effect location bias and denote it with a small b. We can therefore enhance our model by adding the factor b_l for the location bias.

$$Y_{d,l} = \mu + b_l + \epsilon_{d,l}$$

As we know that the least squares estimate is the average of the actual cyclist per location counts minus the average over all counts, we can compute b_l as follows

```
    location_avg <- Daily_train %>% group_by(zaehlstelle) %>% summarize(b_l = mean(gesamt - avg))
```

Our prediction changes therefore to

```
prediction <- avg + Daily_test %>%
    left_join(location_avg, by = "zaehlstelle") %>%
    pull(b_l)
predictionMAPE <- avg + Daily_test_MAPE %>%
    left_join(location_avg, by = "zaehlstelle") %>%
    pull(b_l)
```

Table 9: Data Model results 02

Method	RMSE	MAPE
Mean	1462.112	642.1363
Mean + Loc	1116.696	279.2076

We can see the error has already come down significantly, however (and that's why I included the measure) the MAPE is still above 200% (!).

Month, temperature, weekday, school holiday effect

In the same way I added the location effect, I can add more effects/ predictors. Based on the data exploration I will add in four steps a month effect, a temperature effect (I pick the minimum daily temperature), a weekday effect and a school holiday effect. Our final model is then:

$$Y_{d,l} = \mu + b_l + b_m + b_t + b_{wdy} + b_{shl} + \epsilon_{m,u}$$

The code becomes more cumbersome with each effect added. The code for the final step/ effect looks like this.

```
schoolholiday_avg <- Daily_test %>%
    left_join(location_avg, by = "zaehlstelle") %>%
    left_join(month_avg, by = "month") %>%
    left_join(temperature_avg, by = "min.temp") %>%
    mutate(b_t = replace_na(b_t, 0)) %>%
    left_join(weekday_avg, by = "weekday") %>%
    group_by(school_holiday) %>%
    summarize(b_shl = mean(gesamt - avg - b_l - b_m -
        b_t - b_wdy))

prediction <- Daily_test %>%
    left_join(location_avg, by = "zaehlstelle") %>%
    left_join(month_avg, by = "month") %>%
    left_join(temperature_avg, by = "min.temp") %>%
    mutate(b_t = replace_na(b_t, 0)) %>%
    left_join(weekday_avg, by = "weekday") %>%
    left_join(schoolholiday_avg, by = "school_holiday") %>%
    mutate(pred = avg + b_l + b_m + b_t + b_wdy + b_shl) %>%
    pull(pred)

predictionMAPE <- Daily_test_MAPE %>%
    left_join(location_avg, by = "zaehlstelle") %>%
    left_join(month_avg, by = "month") %>%
    left_join(temperature_avg, by = "min.temp") %>%
    mutate(b_t = replace_na(b_t, 0)) %>%
```

```
    left_join(weekday_avg, by = "weekday") %>%
    left_join(schoolholiday_avg, by = "school_holiday") %>%
    mutate(pred = avg + b_l + b_m + b_t + b_wdy + b_shl) %>%
    pull(pred)
```

Looking at the results from this modelling approach. We can see a decreasing RMSE with every predictor added, but an increase in the MAPE after adding the weekday. I assume this is due to the way the MAPE is calculated.The MAPE as an indicator has several shortcomings, for example underforecasts are treated better than overforecasts [8]. As mentioned earlier I will focus on optimizing RMSE and consider the MAPE just as an easy way to explain model results.

Table 10: Data Model results 03

Method	RMSE	MAPE
Mean	1462.1121	642.1363
Mean + Loc	1116.6965	279.2076
Mean + Loc + Month	946.4407	178.0922
Mean + Loc + Month + Temp	840.2679	165.1314
Mean + Loc + Month + Temp + Wday	809.5285	189.1418
Mean + Loc + Month + Temp + Wday + SchoolHld	803.4224	189.3187

Regression Tree

To better understand the prediction process and the relevant predictors we can use a regression tree. A regression tree is a decision tree for a continuous variable (like our cyclist count). Decision trees are easy to visualize and help understanding the data model. I will create a basic regression tree.

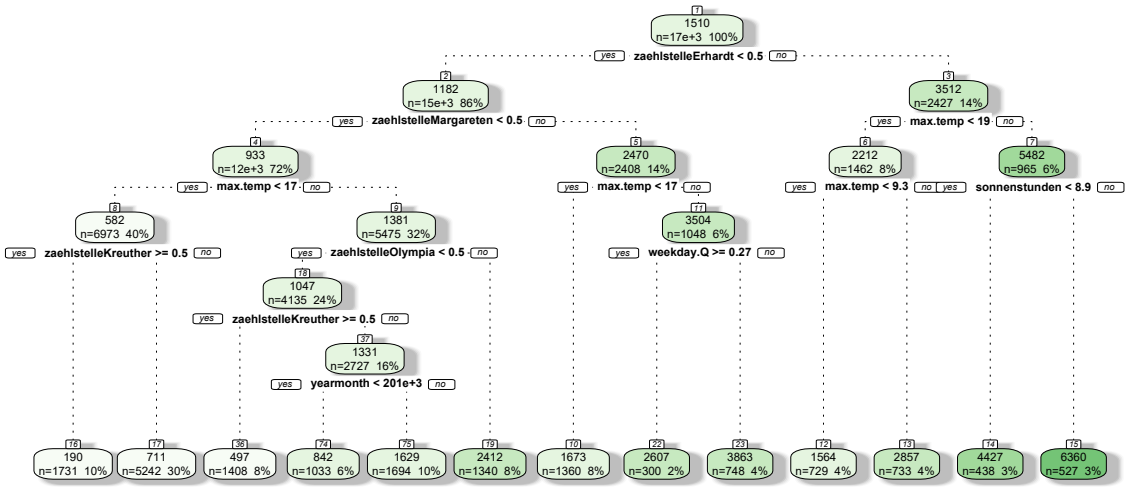
```
RegressionTree <- train(gesamt ~ ., method = "rpart", tuneGrid = data.frame(cp = seq(0,
    0.01, len = 25)), data = Daily_train)
```

The lowest level of a regression tree showing the result is called a leaf.For better visualization I look only at the first nodes of the decision tree, thus creating new leafs on a higher level (this is called very fittingly pruning).

```
pruned_RegressionTree <- prune(RegressionTree$finalModel, cp = 0.007)
```

It can be visualized like this.

Graph 13: Regression Tree visualization



When looking at the visualization it becomes relatively easy to understand how the algorithm works and what a regression tree is. The top number in each box represents the estimate on that node level (on the top node the estimate equals the mean across the training data set), the two lower numbers in the box represent the number of observations falling under that node and their percentage of the total data set. For each node we then see the decision to be made to split the tree.

The first decision shown here is “Is the location not Erhardt?”. A human decision maker would have probably left out the “not” in the decision, resulting in a position swap of the yes and no boxes, but obviously both lead to the same result. Also notice how some predictors are used several times for example you can see the location Kreuther on the lower left side of the tree twice.

We can look at the RMSE and MAPE for the decision tree (for this I will use the full tree not only the pruned/visualized one) and notice a huge improvement.

Table 11: Data Model results 03

Method	RMSE	MAPE
Mean	1462.1121	642.13629
Mean + Loc	1116.6965	279.20762
Mean + Loc + Month	946.4407	178.09220
Mean + Loc + Month + Temp	840.2679	165.13145
Mean + Loc + Month + Temp + Wday	809.5285	189.14179
Mean + Loc + Month + Temp + Wday + SchoolHld	803.4224	189.31871
RegressionTree	489.9534	49.74427

Random Forest

While regression trees are easy to understand and to visualize, they are easily overtrained. The visualization above was just a pruned tree, the actual tree used several thousand nodes. Also regression trees are highly depending on the training data.

We can use Random Forest to overcome some of the deficiencies of the Regression Tree.

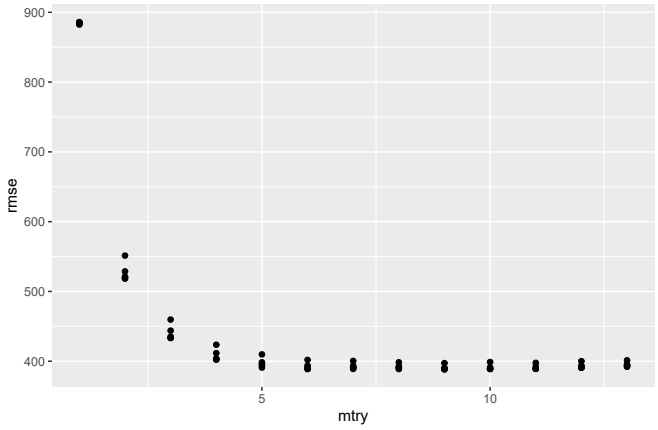
The concept of the Random Forest bases the prediction not on a single regression tree, but on several trees (a forest), which are then averaged. The random part relates to the fact that we take random samples from our test set to build the tree.

Before training the random forest I will optimize two model parameters. I will look at

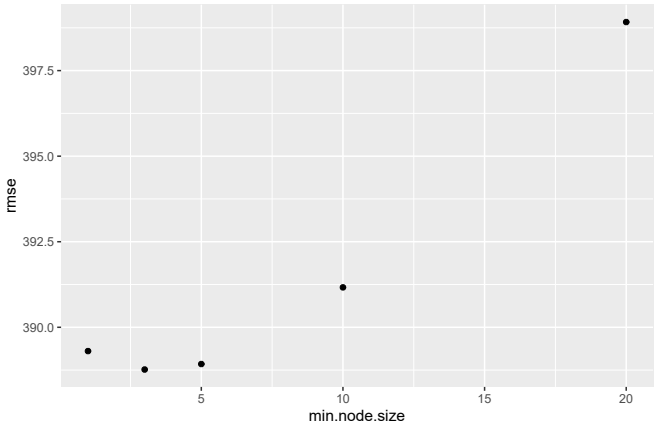
1. m_{mtry}- the number of randomly selected parameter at each node
2. Minimum node size - the minimum number of observations to fall under one node

Visualizing first the m_{mtry} I decide to set m_{mtry} to 10. With m_{mtry} fixed at 10, I look at the minimum node.size. The default value for regression problems is 5, which seems to work well with the given data, so I stick to it.

graph 14: Impact of mtry on the model rmse



graph 15: Impact of the minimum node size on the model rmse



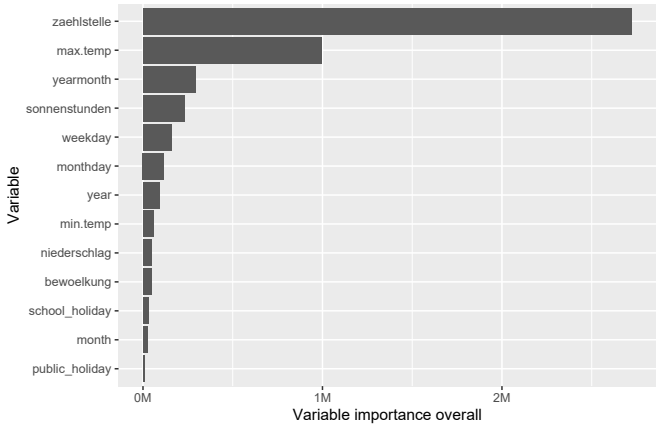
With m_{mtry} and minimum node size set, I can now train the random forest (I will use the ranger package for that) and calculate our predictions.

```
train_rf <- ranger(gesamt ~ ., data = Daily_train, num.tree = 300, mtry = 10, replace = TRUE,
    min.node.size = 5, sample.fraction = 1, seed = 1982, importance = "permutation")

prediction <- predictions(predict(train_rf, data = Daily_test))
predictionMAPE <- predictions(predict(train_rf, data = Daily_test_MAPE))
```

While we can no longer look at a visual representation of the data model as for the regression tree, we can still look at the importance of each predictor in the model (called variable importance). We can see that after the most important variable location (zaehlstelle), the weather does play a greater role than the weekday.

graph 16: Variable importance of the final model



Looking at the result we see an even better RMSE - using the MAPE as interpretation the average error is still over 30%, but that does not deter me from using the model.

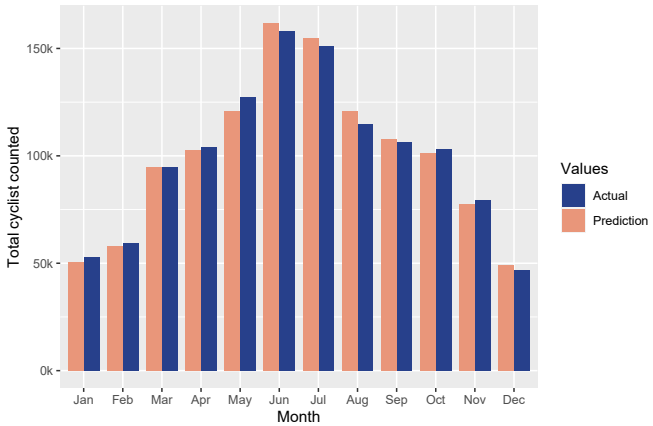
Table 12: Data Model results 03

Method	RMSE	MAPE
Mean	1462.1121	642.13629
Mean + Loc	1116.6965	279.20762
Mean + Loc + Month	946.4407	178.09220
Mean + Loc + Month + Temp	840.2679	165.13145
Mean + Loc + Month + Temp + Wday	809.5285	189.14179
Mean + Loc + Month + Temp + Wday + SchoolHld	803.4224	189.31871
RegressionTree	489.9534	49.74427
RandomForest	382.8965	36.92412

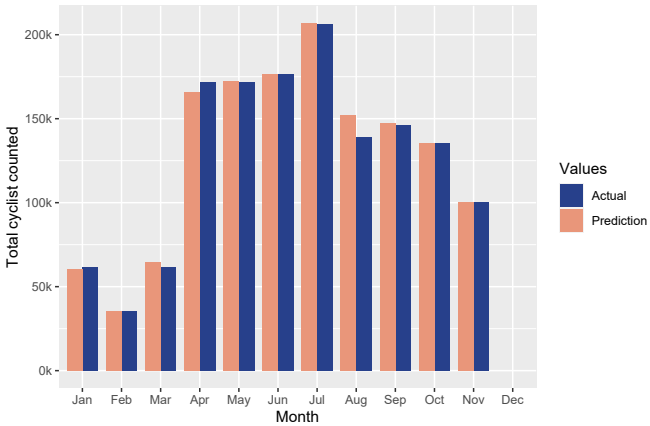
As my intention of the model is to interpret the cyclist data for 2020 to 2022 I will not look at individual daily data, but on monthly aggregates. By aggregating the error should reduce even more as negative and positive errors equal each other out.

A visual comparison for two years at the location Erhardt shows a relatively good match.

graph 17: Actual and Predicted Cyclists for location Erhardt 2014



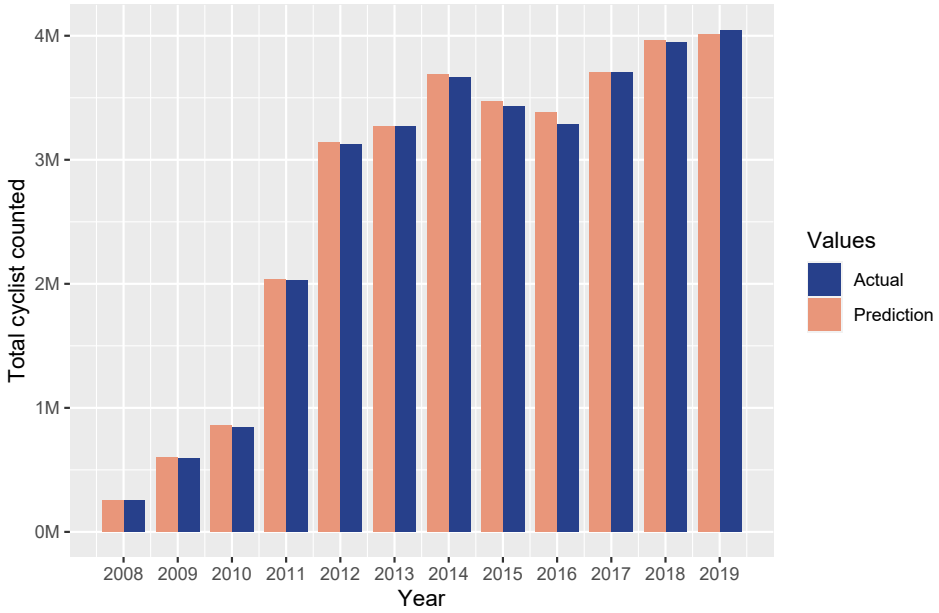
graph 18: Actual and Predicted Cyclista for location Erhardt 2018



Aggregating even further, we also see a relatively good match on an annual level across all locations. The error for 2019 is only 32k cyclist. A high number, but still a lot less than expected by the RMSE (RMSE x 6 locations x 365 days)

= ~800k, so as expected aggregating reduces the error. Also given that the actual count was over 4 million cyclist this error is less than 1% based on the actual results.

graph 19: Predicted and actual Cyclist Count per year



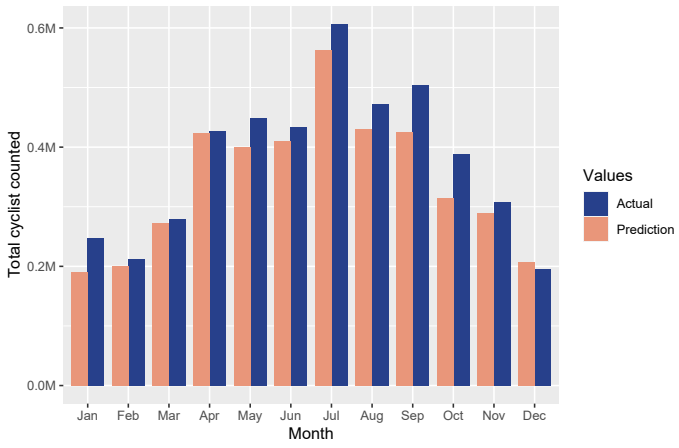
However, when looking at the year we randomly excluded (2016), we can see that this year has the highest error (with around 3%). Given the complexity of the data, I would consider an error of around 3% still good, but it does show potential for further optimization of the model.

Bicycle traffic 2020 to 2022

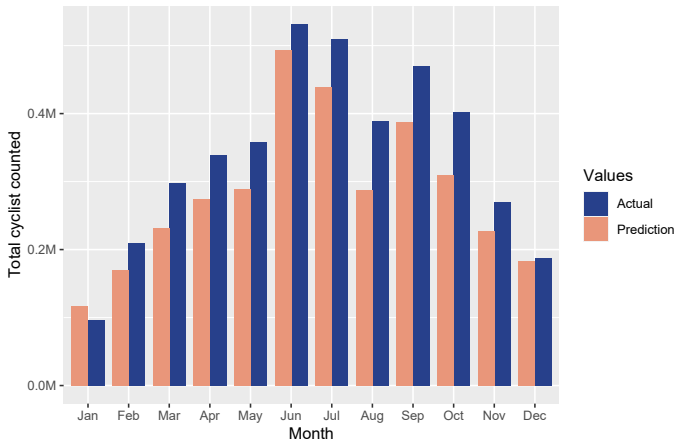
With the model established we can now look at the data which we have left purposely untouched. The values for the year 2020 to 2022. With the global covid pandemic, lock-downs and the rise of home office I expected a reduction in cyclist traffic.

However, using my model we can see a significantly higher number of cyclists than predicted.

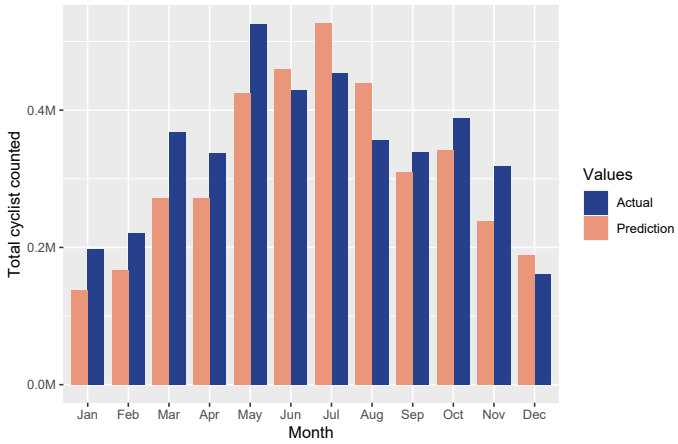
graph 20: Actual and Predicted Cyclist Count for 2020



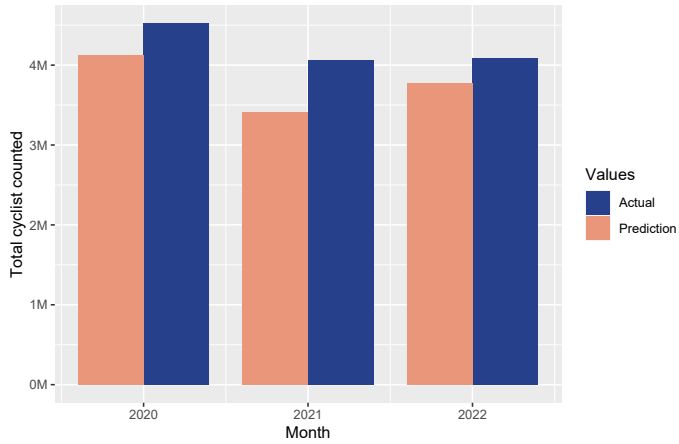
graph 21: Actual and Predicted Cyclist Count for 2021



graph 22: Actual and Predicted Cyclist Count for 2022



graph 23: Actual and Predicted Cyclist Count per year



Even when aggregated to an annual level the data is showing more cyclists than predicted (the error of the prediction being 9%, 16% and 8% of the actual number).

Please be aware the annual data is not prepared to show a clean annual comparison e.g. same number of locations on each day. As mentioned earlier, there is a construction site at the location Arnulf (we excluded it from 2021 onwards) and a change in the counting system at the location Kreuther from March 2020 onwards (also excluded), this certainly helps explaining the decrease in the total cyclists predicted (from around 4M in 2018 and 2019).

It seems my expectation/ hypothesis was wrong. The data shows that cycling became a boost during the pandemic. One of the reasons I did not consider is that the pandemic came along with a strong decline in public transport usage (minus 40% in 2020 [9]). Given the good public transport system in Munich this could probably explain a lot of the uplift and might (over-)compensate the effects, I expected to see regarding home office use and lock-downs (there was no fully enforced lockdown in Munich at any time).

Summary and Outlook

I have downloaded the open data available on daily bicycle usage in the city of Munich from 2008 onward. Quite a bit of data cleaning and preparation had to be done - with a wrong/ different decimal separator being the biggest issue.

With the data cleaned and enriched, I was able to identify a seasonal pattern in each year, that non-surprisingly overlapped with the pattern visible in the available weather data. Looking at the individual weather data and the impact on the number of cyclist, I could see a trend for each factor, but each with a high spread (the strongest trend was in temperature - the higher the temperature, the more cyclists).

Weekdays, public holidays and school holidays also seemed to have impacted the number of cyclists. However, the impact might have also been correlated to the weather / timing of the holidays.

I then developed four models to predict the daily data.

1. Mean
2. Mean corrected by several effects/ biases
3. Regression Tree
4. Random Forest

The random forest produced the best results.

I then looked at the model performance on an annual basis comparing it to all data available till 2019, including one year, which I selected by random and excluded from training the data model. Overall the data model produced good results when aggregating the data to a higher level (maximum error on an annual level around 3%).

Finally I used the model to predict the number of cyclists for the years 2020 to 2022 to see the impact of the Covid pandemic. I was surprised to see a boost in number of cyclists (the prediction error increased to 16% in 2021). A bit of research showed that might be due to a significant lower usage of public transport that overcompensates other effects.

The data is highly interesting and I will investigate it further. On top of my list is further optimization of the current model and inclusion of the 15 minute data sets to structure daily patterns.

References

[1] http://opendata.muenchen.de

[2] Map tiles by Stamen Design http://stamen.com under CC BY 3.0 http://creativecommons.org/licenses/by/3.0 Data by OpenStreetMap http://openstreetmap.org under ODbL http://www.openstreetmap.org/copyright

[3] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiB6JeF-eT9AhVGRvEDHfs3ApEQFnoECAoQAQ&url=https%3A%2F%2Fmedia.frag-den-staat.de%2Ffiles%2Ffoi%2F75110%2FFahrrad_Muenchen_Strassenverkehrstechnik_Heft_01_2011_geschwaerzt.pdf&usg=AOvVaw0F2RwQ4n_fY8hJn4K9fZ9X

[4] https://www.olympiapark.de/en/no_cache/events-tickets/overview/?tx_event_pi4%5Bstart%5D=1338501600&tx_event_pi4%5Bstop%5D=1341093600&tx_event_pi5%5Bm%5D=1338501600

[5] https://www.olympiapark.de/en/no_cache/events-tickets/overview/?tx_event_pi4%5Bstart%5D=1275343200&tx_event_pi4%5Bstop%5D=1277935200&tx_event_pi5%5Bm%5D=1275343200

[6] https://www.kaiserslauternamerican.com/germans-observe-ascension-day-fathers-day-thursday/

[7] https://en.wikipedia.org/wiki/Mean_absolute_percentage_error

[8] https://stats.stackexchange.com/questions/299712/what-are-the-shortcomings-of-the-mean-absolute-percentage-error-mape

[9] https://www.sueddeutsche.de/muenchen/muenchen-mvv-corona-fahrgaeste-bus-app-1.5202785 unfortunately in German - please use a translate function if necessary