Exploratory Analysis: StandyBy Data

Georg Grunsky

2025-06-04

Einleitung

Dieses Dokument beschreibt die vorbereitenden Abschnitte **Prüfung der Datenqualität** und **Explorative Datenanalyse** für den bestehenden Use Case. Ziel der exporativen Analyse ist dabei auch das Finden eines möglichen Ansatzes für die Vorhersage und die damit einhergehende Modifikation und Aufbereitung der Daten. Die Modellierungen selbst werden in den jeweiligen Modellverzeichnissen behandelt.

Prüfung der Datenqualität

Datensätze einlesen

Die Daten werden in zwei Dateien bereitgestellt: ***sickness_table.csv*** und ***sickness_table.xlsx***. Augenscheinlich handelt es sich dabei um den selben Datensatz,was jedoch eingangs noch zu überprüfen ist.

```
require(readr)
path_to_samples <- "../00_sample_data/"
sickness_csv <- read_csv(paste0(path_to_samples,"01_raw/sickness_table.csv"),
    col_types = cols(date = col_datetime(format = "%Y-%m-%d")))
head(sickness_csv)
```

```
# A tibble: 6 x 8
   ...1 date                n_sick calls n_duty n_sby sby_need dafted
  <dbl> <dttm>                <dbl> <dbl>  <dbl> <dbl>    <dbl>  <dbl>
1     0 2016-04-01 00:00:00     73  8154   1700    90        4       0
2     1 2016-04-02 00:00:00     64  8526   1700    90       70       0
3     2 2016-04-03 00:00:00     68  8088   1700    90        0       0
4     3 2016-04-04 00:00:00     71  7044   1700    90        0       0
5     4 2016-04-05 00:00:00     63  7236   1700    90        0       0
6     5 2016-04-06 00:00:00     70  6492   1700    90        0       0
```

```
require(readxl)
sickness_xls <- read_excel(paste0(path_to_samples,"01_raw/sickness_table.xlsx"),
    col_types = c("numeric", "date", "numeric",
        "numeric", "numeric", "numeric", "numeric",
        "numeric"))
head(sickness_xls)
```

```
# A tibble: 6 x 8
   ...1 date                n_sick calls n_duty n_sby sby_need dafted
  <dbl> <dttm>                <dbl> <dbl>  <dbl> <dbl>    <dbl>  <dbl>
1     0 2016-04-01 00:00:00     73  8154   1700    90        4       0
2     1 2016-04-02 00:00:00     64  8526   1700    90       70       0
3     2 2016-04-03 00:00:00     68  8088   1700    90        0       0
4     3 2016-04-04 00:00:00     71  7044   1700    90        0       0
5     4 2016-04-05 00:00:00     63  7236   1700    90        0       0
6     5 2016-04-06 00:00:00     70  6492   1700    90        0       0
```

Datensätze vergleichen

```
identical(sickness_csv, sickness_xls)
```

```
[1] FALSE
```

Die Überprüfung mittels *identical* besagt, dass die Datensätze nich identisch sind. Die in den jeweiligen Daten-Dictionaries beschriebenen Datenstrukturen lassen jedoch schon darauf schließen. Eine Summary der Wertebereiche der vorliegenden Daten schafft mehr Klarheit.

Daten Summary

sickness_csv

```
summary(sickness_csv)
```

```
      ...1             date                         n_sick      
 Min.   :   0.0   Min.   :2016-04-01 00:00:00   Min.   : 36.00  
 1st Qu.: 287.8   1st Qu.:2017-01-13 18:00:00   1st Qu.: 58.00  
 Median : 575.5   Median :2017-10-28 12:00:00   Median : 68.00  
 Mean   : 575.5   Mean   :2017-10-28 12:00:00   Mean   : 68.81  
 3rd Qu.: 863.2   3rd Qu.:2018-08-12 06:00:00   3rd Qu.: 78.00  
 Max.   :1151.0   Max.   :2019-05-27 00:00:00   Max.   :119.00  
     calls           n_duty         n_sby       sby_need          dafted      
 Min.   : 4074   Min.   :1700   Min.   :90   Min.   :  0.00   Min.   :  0.00  
 1st Qu.: 6978   1st Qu.:1800   1st Qu.:90   1st Qu.:  0.00   1st Qu.:  0.00  
 Median : 7932   Median :1800   Median :90   Median :  0.00   Median :  0.00  
 Mean   : 7920   Mean   :1821   Mean   :90   Mean   : 34.72   Mean   : 16.34  
 3rd Qu.: 8828   3rd Qu.:1900   3rd Qu.:90   3rd Qu.: 12.25   3rd Qu.:  0.00  
 Max.   :11850   Max.   :1900   Max.   :90   Max.   :555.00   Max.   :465.00  
```

sickness_xls

```
summary(sickness_xls)
```

```
      ...1             date                         n_sick      
 Min.   :   0.0   Min.   :2016-04-01 00:00:00   Min.   : 36.00  
 1st Qu.: 287.8   1st Qu.:2017-01-13 18:00:00   1st Qu.: 58.00  
 Median : 575.5   Median :2017-10-28 12:00:00   Median : 68.00  
 Mean   : 575.5   Mean   :2017-10-28 12:00:00   Mean   : 68.81  
 3rd Qu.: 863.2   3rd Qu.:2018-08-12 06:00:00   3rd Qu.: 78.00  
 Max.   :1151.0   Max.   :2019-05-27 00:00:00   Max.   :119.00  
     calls           n_duty         n_sby       sby_need          dafted      
 Min.   : 4074   Min.   :1700   Min.   :90   Min.   :  0.00   Min.   :  0.00  
 1st Qu.: 6978   1st Qu.:1800   1st Qu.:90   1st Qu.:  0.00   1st Qu.:  0.00  
 Median : 7932   Median :1800   Median :90   Median :  0.00   Median :  0.00  
```

```
 Mean   : 7920   Mean   :1821   Mean   :90   Mean   : 34.72   Mean   : 16.34
 3rd Qu.: 8828   3rd Qu.:1900   3rd Qu.:90   3rd Qu.: 12.25   3rd Qu.:  0.00
 Max.   :11850   Max.   :1900   Max.   :90   Max.   :555.00   Max.   :465.00
```

Gemäß den angezeigten Wertebereichen handelt es sich tatsächlich um die gleichen Datensätze. Anhand der .csv Datei wird noch überprüft ob der Datensatz fehlende Werte beinhaltet, bevor mit der weiteren Datenexploration fortgefahren wird.

NAs und “Missing Values”

```
alldays <- seq(from = as.Date("2016-04-01"),
               to = as.Date("2019-05-27"),
               by = 1)

if (identical(alldays, as.Date(sickness_csv$date))) {
  print("no missing dates")
  } else {
  print("you have to deal w/ missing dates")}
```

```
[1] "no missing dates"
```

```
if (!anyNA.data.frame(sickness_csv)) {
  print("no missing values")
  } else {
    print("you have to deal w/ missing values")}
```

```
[1] "no missing values"
```

In dieser Zeitreihe müssen weder mit fehlenden Zeitwerten noch NAs in den Datenpunkten behandelt werden. Die Prüfung der Datenqualität wird somit abgeschlossen.

Zusammenfassung

1. Die bereitgestellten Datensätze scheinen identisch zu sein. Es wird nur mit der .csv Datei weitergearbeitet.
2. Es gibt keine fehlenden Zeitwerte und/oder NAs in den Datenpunkten
3. Der bisherige Fixwert von 90 Bereitschaftspersonen (*n_sby*) scheint in den meisten Fällen im Vergleich zu *sby_need* zu hoch zu sein. Es gibt jedoch auf Tage wo mehr als 90 Personen gebraucht werden.
4. Die Zeitreihe behandelt den Zeitraum von 2016-04-01 bis 2019-05-27. Eine Periode von drei Jahren vor der COVID Pandemie. Es ist davon auszugehen, dass COVID die aktuellen Zahlen verändert hat. Denoch hilft eine erfolgreiche Umsetzung einer Vorhersage auf den Testdaten, das Projekt mit aktuellen Daten voranzutreiben und zukünftig Bereitschaftskosten sparen zu können.
5. Die Anzahl der Anrufe und die Anzahl des krankgemeldeten Personals scheint ähnliche Proportionen aufzuweisen, nur um etwa den Faktor 100 kleiner.

Explorative Datenanalyse

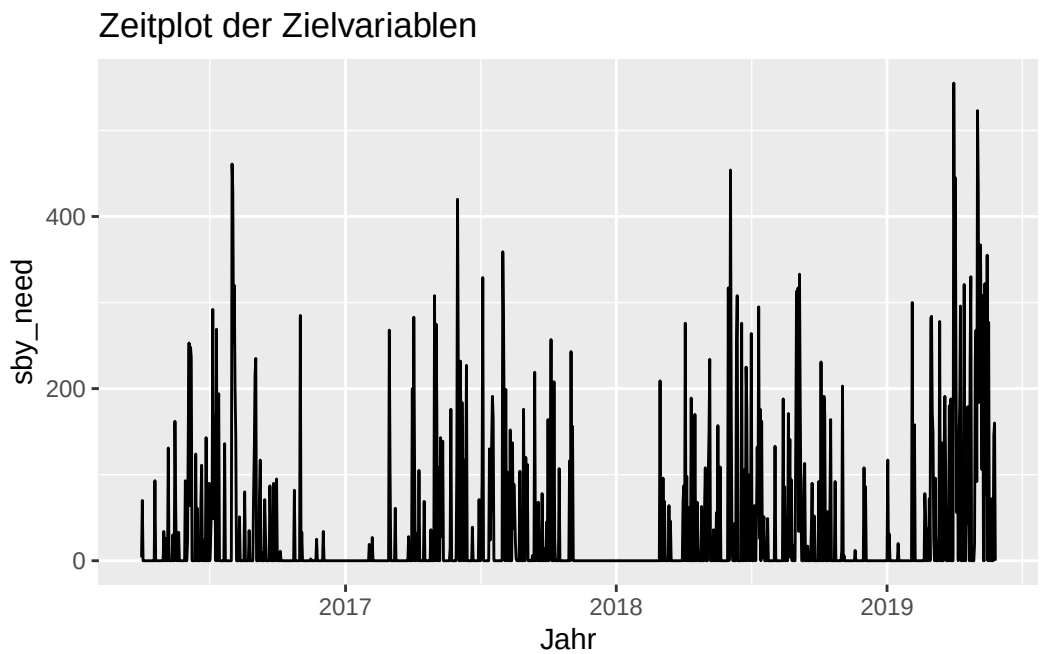
Hyndman und Athanasopoulos (2021) bietet von der Exploration bis zur Modellierung einen sehr ausführlichen Anhalt für die Arbeit mit Zeitreihendaten. Für die Analyse in diesem Dokument wird das dazugehörige R-Package *fpp3* verwendet.

Ein weiteres R-Package, das gute Werkzeuge für die explorative Analyse aber auch für die Erstellung von Models anbietet ist *caret*. Das Paket eignet sich zwar weniger gut für Zeitreihendaten, aus der Dokumentation alleine kann aber bereits viel gelernt werden. (Kuhn 2019).

Visualisierung der Zielvariable

Die Aufgabenstellung liegt in der Vorhersage des bereitzuhaltenden Bereitschaftspersonals auf Basis einer vermuteten Saisonalität im historischen Bedarf. Die Zielvariable für diese Vorhersage ist daher *sby_need*, die das tatsächlich benötigte Bereitschaftspersonal pro Tag ausdrückt. Die u.a. Darstellung der Zielvariablen gibt einen ersten Eindruck über deren Ausprägung und mögliche Herausforderungen. Um das R-Package *fpp3* für Forcasting in Zeitreihen zu verwenden, wird der Datensatz vorerst in ein entsprechendes Objekt konvertiert. Hierbei werden auch die erste Spalte (*id*), die bisher konstante Einteilung des Bereitschaftspersonals (*n_sby*), sowie die Anzahl der zusätzlich benötigten Fahrer:innen (*dafted*) entfernt.

```
ts_sby <- sickness_csv %>%
  select(-c("...1", "n_sby", "dafted")) %>%
  mutate(date = as.Date(date)) %>%
  as_tsibble(index = date)

ts_sby %>%
  autoplot(sby_need) +
  labs(x = "Jahr",
       title = "Zeitplot der Zielvariablen")
```

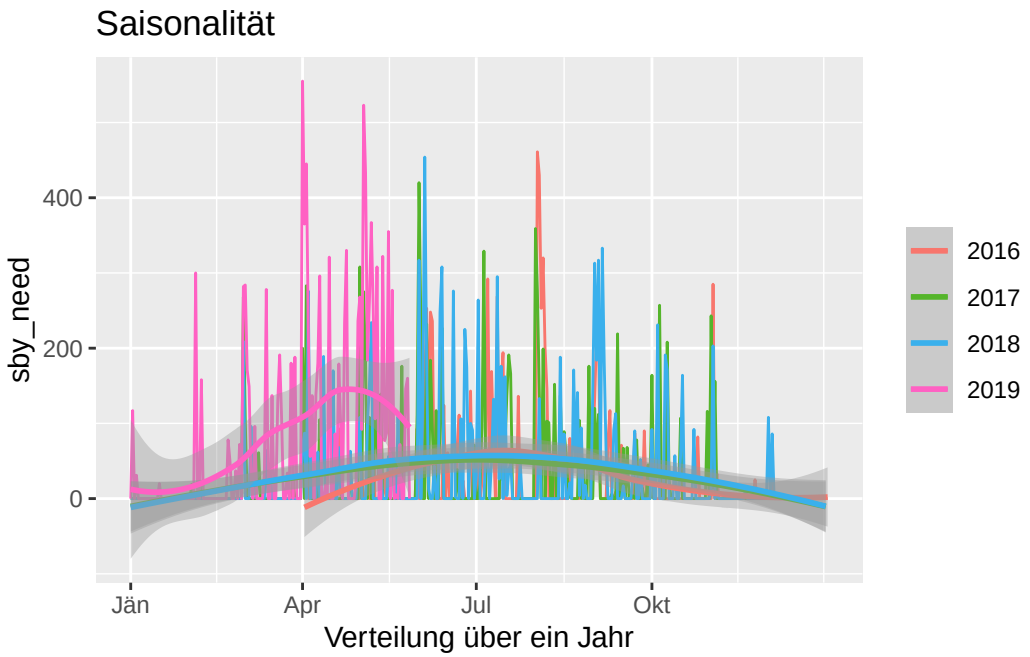


Die vermutete Saisonalität ist in der zeitlichen Darstellung des Bedarfs gut erkennbar. Augenscheinlich scheint es monatliche Spitzen in der Zielvariablen zu geben und um den Jahreswechsel ist oft nur

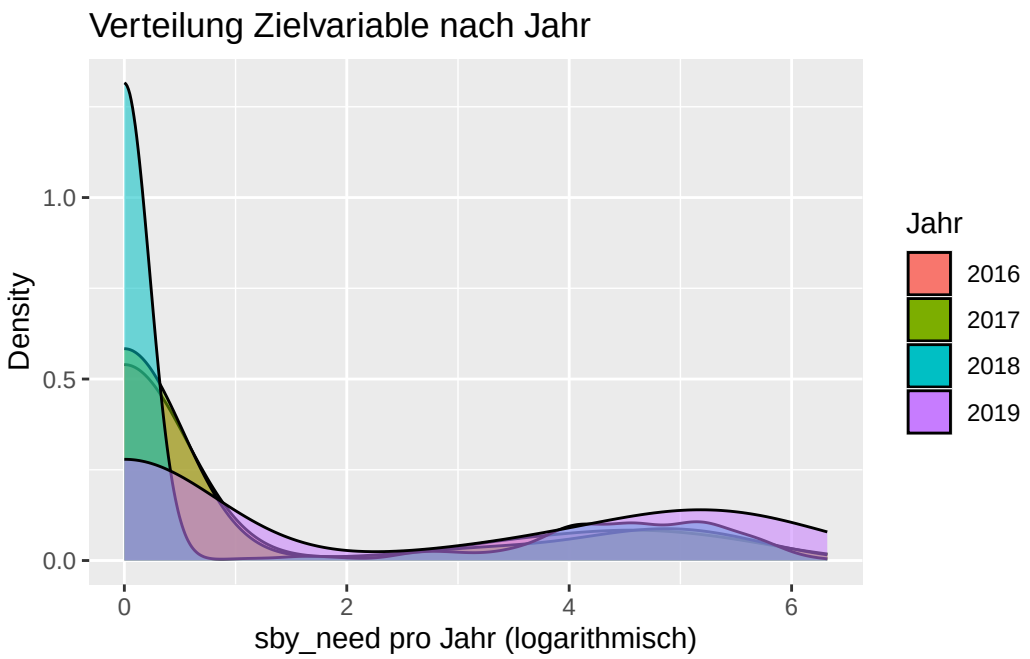
geringer Bedarf vorhanden. Mit dem Jahreswechsel auf 2019 ändert sich das Muster der Bedarf weist ab dem Jahreswechsel deutlich höhere Spitzen auf und auch um in der “sonst ruhigen” Zeit zwischen 2018 und 2019 sind *Peaks* erkennbar. Die rapiden Wechsel zwischen “kein Bedarf” und “sehr hoher Bedarf” an Bereitschaftspersonal könnten ein Vorhersagemodell vor eine Herausforderung stellen, vor allem da eine Vorgabe lautet, nie zu wenig Bereitschaftspersonal vorzusehen. Auch der Wechsel der Muster mit 2019 könnte zur Auswirkung haben, dass sich die historischen Daten nur schlecht als Prädiktoren für die zukünftige Entwicklung eignen.

Im u.a. Saisonalitätsplot werden die deutlich stärker ausgeprägten “Peaks” im Jahr 2019 auch durch die Smooth-Curve gut ersichtlich. Diese Spitzenwerte zeigen auch im Vergleich zwischen den Jahren immer wieder Abweichungen zu einander und treten nicht zu exakt den gleichen Zeitpunkten auf. Auch das dargestellte 99% Confidence Level der Smooth-Curve deckt diese nicht ab. Die explizite Vorgabe “immer ausreichend” Bereitschaftspersonal vorzusehen” wird dadurch erschwert.

```
ts_sby %>%
  gg_season(sby_need, period = "1y") +
  geom_smooth(level = 0.99) +
  labs(title = "Saisonalität",
       y = "sby_need",
       x = "Verteilung über ein Jahr")
```




```
ts_sby %>%
  ggplot() +
  geom_density(aes(x = log(sby_need + 1), fill = as.factor(year(date)), alpha = 0.3),
                   show.legend = c(fill = TRUE, alpha = FALSE)) +
  labs(title = "Verteilung Zielvariable nach Jahr",
       y = "Density",
       x = "sby_need pro Jahr (logarithmisch)",
       fill = "Jahr")
```

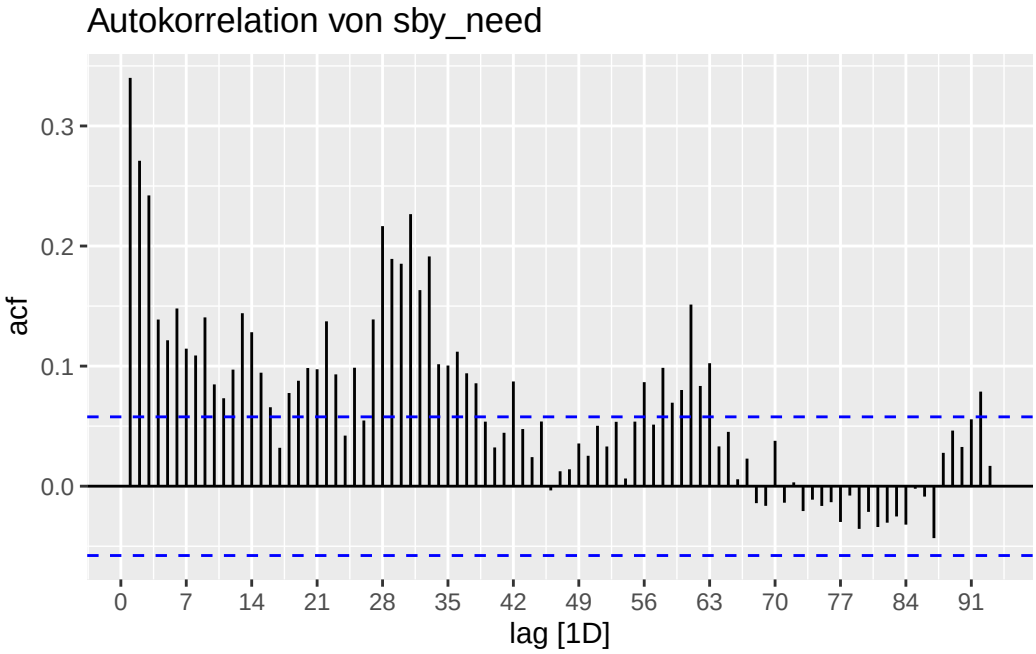


Wie bereits zuvor erläutert, sieht man auch in der Dichtefunktion des logarithmisch dargestellten Wertes von *sby_need* den Wechsel zwischem geringen und sehr hohem Bedarf. Eine Normalverteilung liegt logischerweise nicht vor, da der Modus zwar bei 0 zu liegen scheint, ein negativer Wert aber freilicherweise nicht vorkommen kann. Sieht man von der Häufung zum den Wert 0 ab, erkennt man eine zweite kleinere Akkumulation von Werten zwischen grob 3.5 und 6.5. Auch hier zieht das Jahr 2019 die Werte wieder etwas nach oben. In nicht logarithmische ausgedrückten Werten wäre das zwischen ca. 33 und 665 Fahrer:innen, die bereitgehalten werden müssen. Man könnte argumentieren, dass ein Vorhalten von 0 Standby-Fahrer:innen unrealistisch ist und aus dieser Darstellung bereits einen Mindestwert von ca. 35 Personen in Bereitschaft ableiten. Kostenmäßig wäre dies bereits eine Ersparnis gegenüber den derzeit 90 vorgehaltenen Bereitschaftsfahrenden. Die Herausforderung liegt abermals in der treffsicheren Vorhersage der, in augenscheinlich monatlichen Abständen auftretenden, Spitzenwerte.

Autokorrelation der Zielvariable

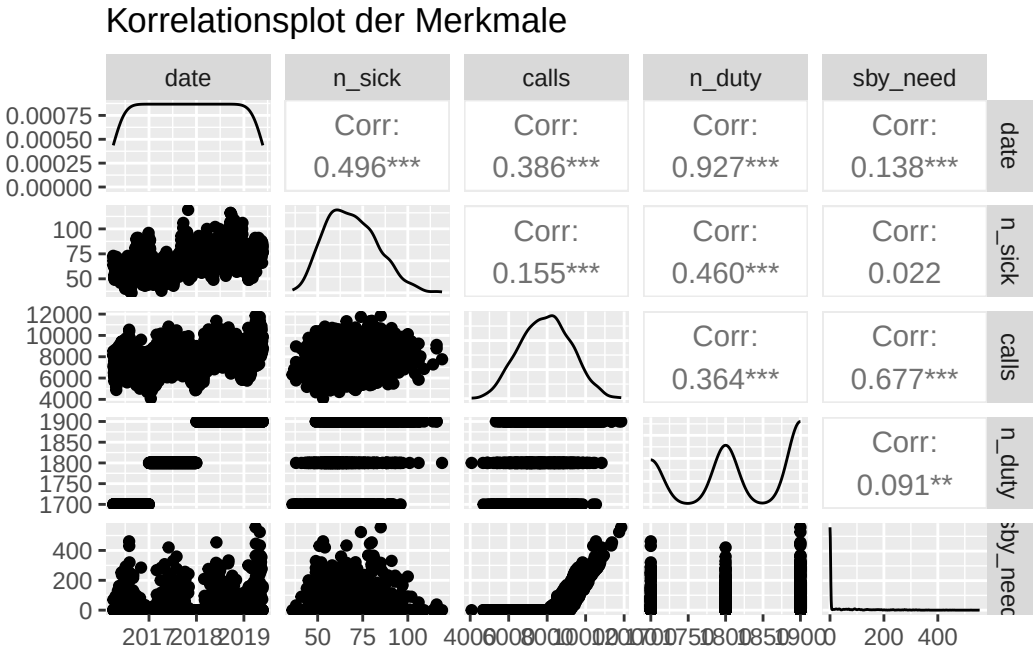
Die Autokorrelation der Zielvariable (s.u.) zeigt, neben der erwarteten monatlichen Korrelation in den *lags* 28 bis 33, auch wöchentliche Peeks Die rasche Abnahme der Korrelation zu Beginn des Plots lässt auf einen nur schwach ausgeprägten Trend in den Werten schließen, vmtl. da dieser augenscheinlich erst mit 2019 auftritt. Ab *lag* 63 zeigt die Autokorrelation teils negative Werte und wird daher mehr oder weniger unbrauchbar.

```
ts_sby %>%
  ACF(y = sby_need, lag_max = 93) %>%
  autoplot() +
  labs(title = "Autokorrelation von sby_need")
```



Übersicht über die weiteren Datenmerkmale

```
ts_sby %>%
  GGally::ggpairs(title = "Korrelationsplot der Merkmale")
```



sby_need (die Zielvariable) korreliert augenscheinlich am meisten mit der Anzahl der Notfallanrufe (*calls*). Hier liegt, ab einer gewissen Anzahl an *calls* nahezu eine lineare Korrelation vor. Das klingt durchaus plausibel. Die vermutete Saisonalität ist in der Übersicht der Variablen *sby_need*, *calls* und *n_sick* (der Anzahl der krankgemeldeten Bereitschaftsfahrer:innen), mit dem menschlichen Auge, gut zu erkennen. Bei dem Merkmal *calls* wirkt diese jedoch am stabilsten und auch die Wertevereilung scheint hier am besten einer Normalverteilung zu folgen.

Das Merkmal *n_duty* (die Anzahl der diensthabenden Bereitschaftsfahrer:innen) weist nur drei unterschiedliche Werte auf, die ausschließlich eine Abhängigkeit zum Datum haben. Hierbei wurde die Anzahl der Diensthabenden jedes Jahr zum ersten Januar um 100 Personen erhöht. Am 01.01.2019 ist dies jedoch nicht geschehen. Dieser Umstand muss vermutlich gesondert mit den Entscheidungsträgern besprochen und ggf. nachgezogen werden.

n_sick zeigt zwar die angesprochene Saisonalität, weist, aufgrund der o.a. Grafik, aber nur eine geringe Korrelation zur Zielvariablen auf. Es scheint plausibel, dass eine höhere Anzahl an Krankenständen, bei gleichbleibenden oder steigenden Notfällen zu einem höheren Bedarf an Bereitschaftspersonal führt. Möglicherweise unterstützt die Kombination dieser beiden Merkmale eine erfolgsversprechende Vorhersage.

Für die Vorhersage des Merkmals *sby_need* gibt es zu diesem Zeitpunkt daher fünf mögliche Ansatzpunkte.

1. Die direkte Vorhersage aufgrund der eigenen Saisonalität
2. Eine indirekte Vorhersage aufgrund der Saisonalität des Merkmals *calls* und der, ab einem gewissen Wert, nahezu linear anzunehmenden Korrelation mit *sby_need*. Letztere zeigt jedoch “drei Liniaritäten” und ist wahrscheinlich durch *n_duty* beeinflusst.
3. Wie in Punkt 2., nur dass zusätzlich eine jährliche Steigerung von *n_duty* berücksichtigt wird um aufgetretene Trends abzuflachen und mehr Fokus auf Saisonalität legen zu können. Dieses zu generierende Merkmal wird als regulierte calls, *reg_calls* bezeichnet.
4. Ein indirekte Vorhersage aufgrund der Saisonaltität eines neuen kombinierten Merkmals aus *calls* und *n_sick*, die jedoch zuerst zu evaluieren ist. Das neue Merkmal wird *calls_sick* genannt.
5. Wie in Punkt 4., aber wiederum unter Berücksichtigung einer jährlichen Steigerung von *n_duty* um aufgetretene Trends abzuflachen. Als Bezeichnung wird *reg_calls_sick* verwendet.

Die weitere Analyse konzentriert sich daher auf die ursprünglichen Merkmale *date*, *sby_need* und *calls* sowie auf die neu generierten Variablen.

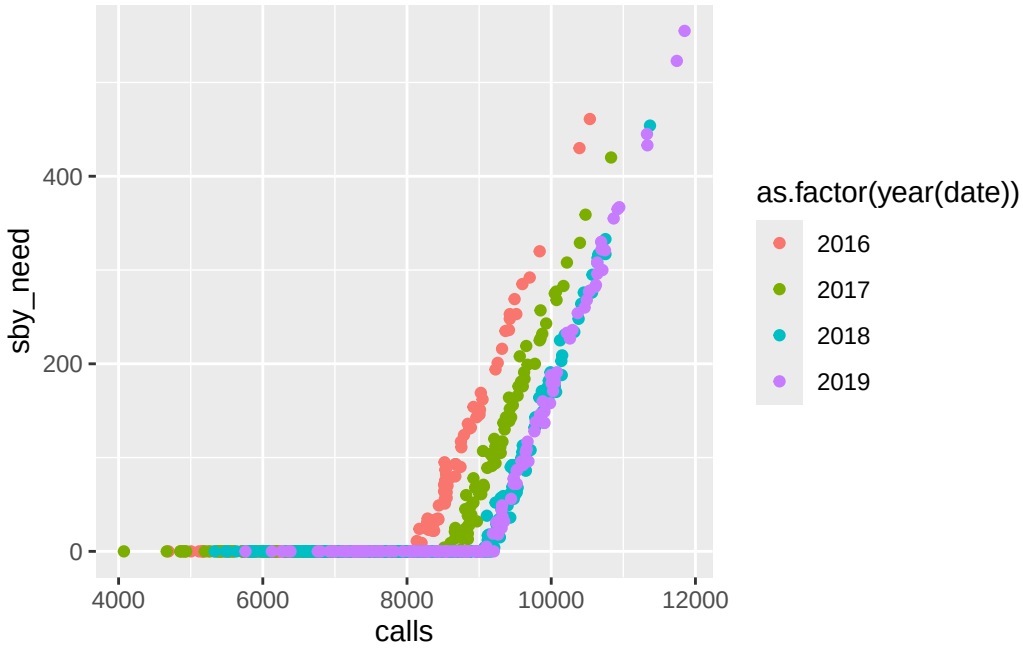
Merkmalsgenerierung

Die in Punkt 4. beschriebene kombinierte Variable *calls_sick* aus *calls* und *n_sick* wird als Anrufe je krankgemeldetem/r Einsatzfahr:in verstanden und daher berechnet als

$$calls\_sick_t = \frac{calls_t}{n\_sick_t}$$

Wie der u.a. Korrelationsplot von *calls* und *sby_need* zeigt, hatten die Erhöhungen des diensthabenden Personals Auswirkungen auf die Notwendigkeit Bereitschaftspersonal zu aktivieren. Die Tatsache, dass 2018 und 2019 genauso wie bei dem Merkmal *n_duty* eine gleiche Korrelation vorliegt, bestätigt die Annahme der Abhängigkeit zu *n_duty* vorliegt.

```
ggplot(ts_sby,
            aes(x = calls,
                y = sby_need,
                colour = as.factor(year(date))
                )) +
  geom_point()
```



Die Abstände im augenscheinlichen “Intercept” betragen zwischen den Jahren jeweils etwa 500 (2016: ~8.000 *calls*, 2017: ~8.500 *calls*, 2018 und 2019: ~9.000 *calls*). Das ist jeweils das fünffache des jährlichen Anstiegs von *n_duty*.

Wurden weniger Anrufe als der jeweilige “Intercept” getätigt wurde auch kein Bereitschaftspersonal aktiviert. Sollte ein Ansatz gewählt werden, in dem *calls* oder *reg_calls* für die Vorhersage verwendet wird, wird dieser Intercept, vmtl. in einem linearen Modell zu bedenken sein.

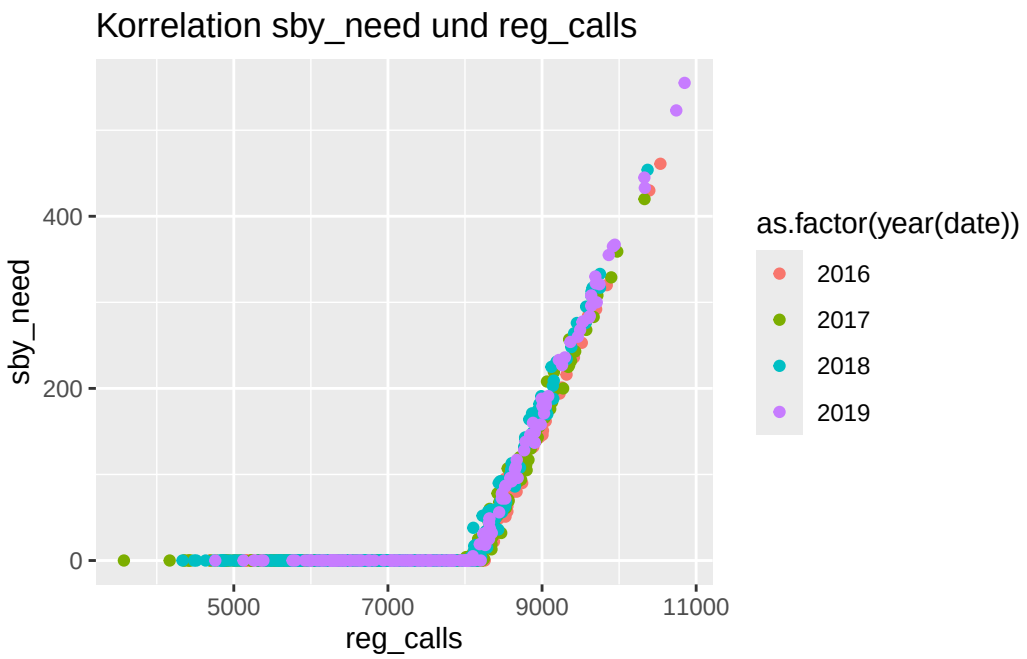
Die oben festgestellte Auswirkung der Erhöhung mit einem Faktor 5 wird in der Merkmalsgenerierung berücksichtigt.

$$reg\_calls_t = calls_t - n\_duty_t \mid n\_duty_t = (n\_duty_{year} - 1700) * 5$$

Das neue Merkmal *reg_calls_sick* leitet sich davon ab.

$$reg\_calls\_sick_t = \frac{reg\_calls_t}{n\_sick_t}$$

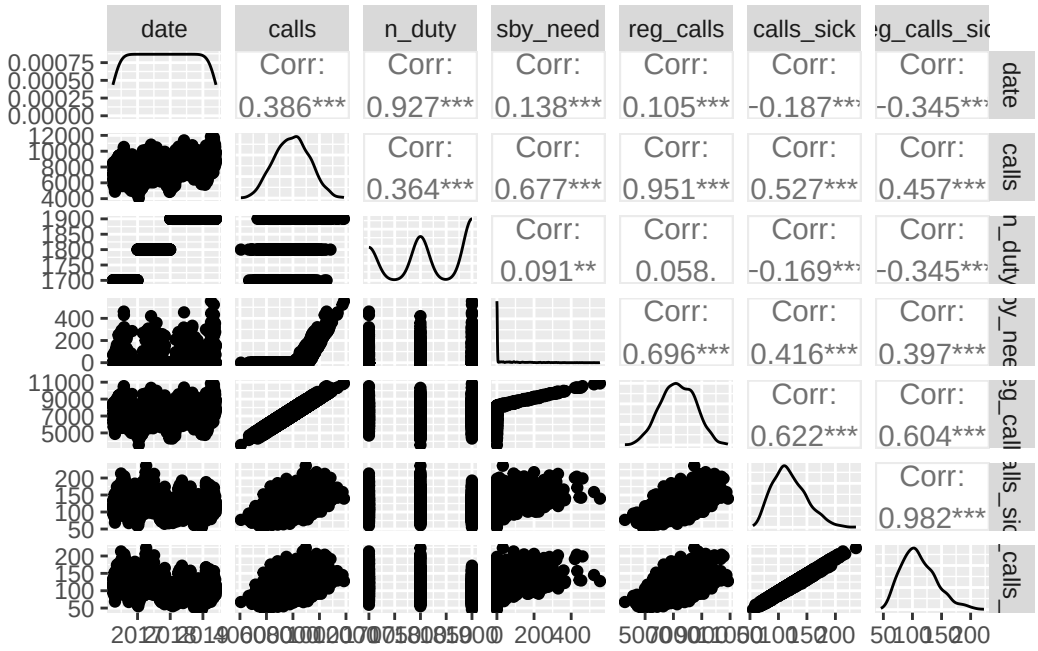
```
ts_sby <- ts_sby %>%
  mutate(reg_calls = calls - (n_duty - 1700) * 5,
         calls_sick = calls/n_sick,
         reg_calls_sick = reg_calls/n_sick) %>%
  select(-c("n_sick"))

ggplot(ts_sby,
            aes(x = reg_calls,
                y = sby_need,
                colour = as.factor(year(date))
                )) +
  geom_point() +
  labs(title = "Korrelation sby_need und reg_calls")
```



Die Auswirkung der vorgenommenen Anpassungen an *reg_calls* sind in der Grafik gut erkennbar.Der mittlere Intercept liegt bei etwa 8150 Anrufen. Bei weniger Anrufen an einem Tag wird kein zusätzliches Bereitschaftspersonal benötigt.

Wie die nachgestellte Grafik zeigt konnte die Korrelation von *reg_calls* mit *sby_need* gegenüber *calls* leicht gesteigert werden. Bei den neuen Merkmalen *calls_sick* und *reg_calls_sick* ist diese jedoch vergleichsweise nicht ausreichend gegeben.

```
ts_sby %>%
  GGally::ggpairs()
```



Um festzustellen welcher Ansatz möglicherweise erfolgversprechender ist, lohnt es sich die Variablen auf die vermeintliche Vorhersagbarkeit ihrer Saisonalität hin zu überprüfen. Hyndman und Athanasopoulos (2021) verweist hierbei auf die spektrale Entropy (Shannon) einer Zeitreihe, die einen Wert zwischen 0 und 1 ausgibt. Je niedriger der Wert, desto stärker ist die Saisonalität und der Trend der Zeitreihe.

```
ts_sby %>%
  select(-c("n_duty")) %>%
  features_all(feat_spectral)
```

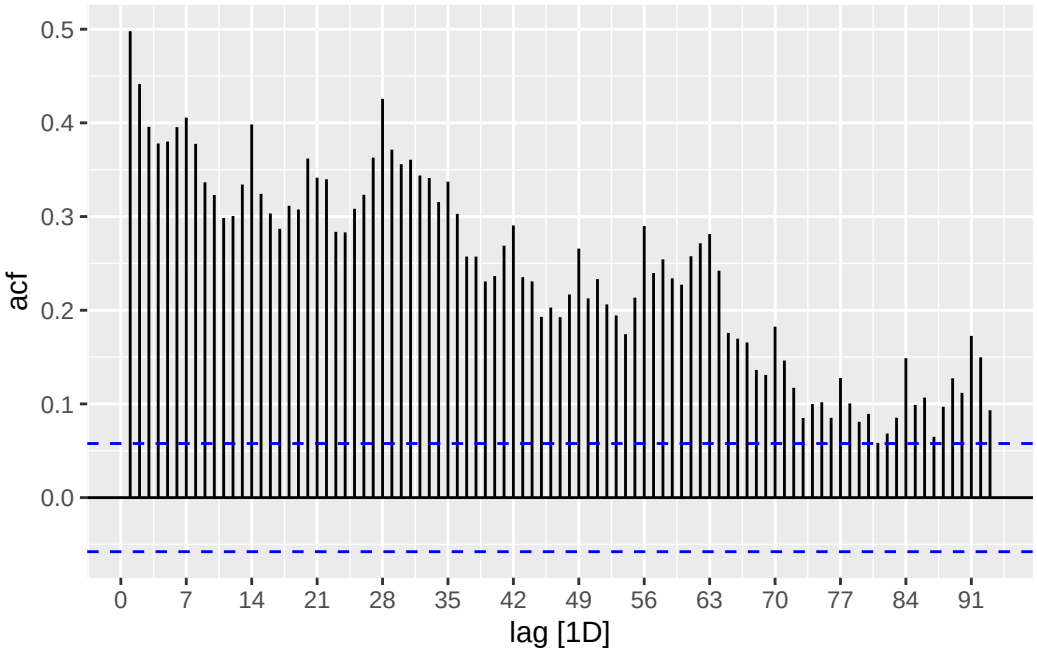
```
# A tibble: 1 x 5
  calls_spectral_entropy sby_need_spectral_entropy reg_calls_spectral_entropy
                   <dbl>                     <dbl>                       <dbl>
1                  0.812                     0.945                       0.880
# i 2 more variables: calls_sick_spectral_entropy <dbl>,
#   reg_calls_sick_spectral_entropy <dbl>
```

Interessanterweise, ist trotz einer augenscheinlich gut erkennbaren Saisonalität der Entropie-Wert durchgehend relativ hoch. Am ehesten scheint sich an dieser Stelle das unregulierte Merkmal der Notfallanrufe (*calls*) für eine saisonal-bedingte Vorhersage zu eignen. Dieses Merkmal scheint annähernd normalverteilt zu sein und die Korrelation zur Zielvariablen *sby_need* liegt nur leicht unter dem regulierten Wert *reg_calls*. *calls_sick* und *reg_calls_sick* schneiden bei der Betrachtung der letzten beiden Faktoren deutlich schlechter ab.

Untersuchung des Merkmals “calls”

Der Autokorrelationsplot der Variablen *calls* soll weiteren Aufschluss über dieses Merkmal geben. Hierfür werden 93 lags, also etwa drei Monate betrachtet.

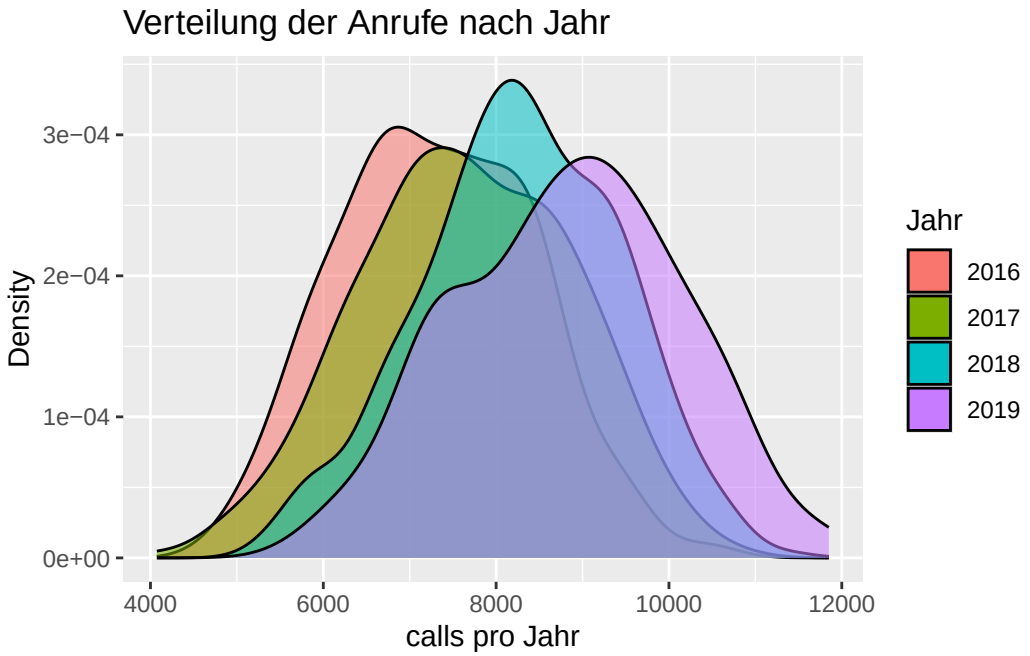
```
ts_sby %>%
  ACF(y = calls, lag_max = 93) %>%
  autoplot()
```



Die höhere Autokorrelation in den niedrigen “lags”, die im Verlauf abnimmt, zeigt, dass die Daten einem Trend unterliegen. Die “kleinen” Peaks bei 7, 14, 21, etc. zeigen eine leichte wochenweise Saisonalität, der etwas höhere Peak bei lag 28 weist auf eine ähnliche, aber augeschneinlich stabilere, monatliche Saisonalität wie die Zielvariable hin. Auch hier nimmt der Autokorrelationswert ab lag 63 schneller ab, rutscht jedoch nicht ins Negative sondern zeigt weiters verwendbare Werte.

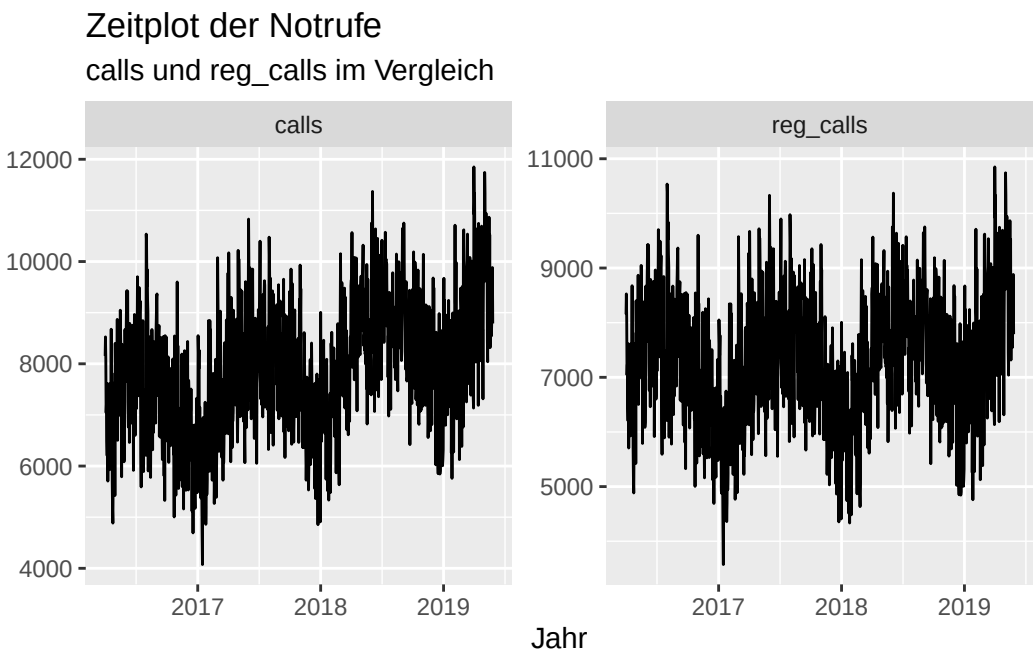
Die Verteilung des möglichen Prädiktors gibt Auschluss, bzw. einen Überblick über die zu erwartenden Werte.

```{r}
ggplot() +
  geom_density(data = ts_sby, aes(x = calls, fill = as.factor(year(date)), alpha = 0.3),
                   show.legend = c(fill = TRUE, alpha = FALSE)) +
  labs(title = "Verteilung der Anrufe nach Jahr",
       y = "Density",
       x = "calls pro Jahr",
       fill = "Jahr")
```



Wie auch bereits im Autokorrelationsplot ist hier der Trend über die Jahre in der Verschiebung der Kurven nach rechts gut erkennbar. Die Verteilung des Merkmals nähert sich einer Normverteilung deutlich besser an als die Zielvariable (von den kleinen Seitenhügeln einmal abgesehen).

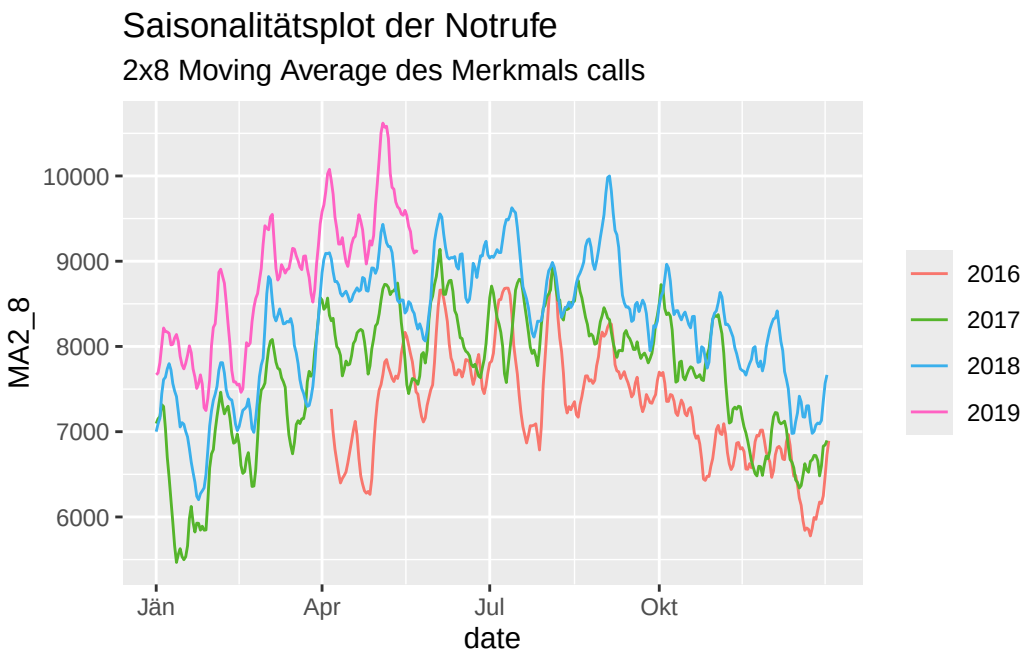
```
ts_sby %>%
  autoplot(vars(calls, reg_calls)) +
  labs(x = "Jahr",
       title = "Zeitplot der Notrufe",
       subtitle = "calls und reg_calls im Vergleich")
```



Im Vergleich der Variablen *calls* und *reg_calls* ist die Modifikation der letzteren gut an dem fehlenden Trend zu erkennen. Wie auch bereits die spektrale Entropy zeigte, ist das unregulierte Merkmal vermutlich leichter vorherzusagen. Schließlich ist der Trend ein wesentliches Element eines Forcasting. Gleichzeitig muss die Regularisierung in Bezug auf das diensthabende Personal (*n_duty*) in der Korrelation mit der Zielvariablen berücksichtigt werden, da dies wiederum Einfluss auf den Bedarf an Bereitschaftsfahrenden hat. Andernfalls wäre es falsch, bereits jetzt rechnerisch das diensthabende Personal der zukünftigen Zeiträume in einem Vorhersagemodell festzulegen. Vielmehr wäre es sinnvoll das flexibel zu gestalten. Über diese Punkte wird jedenfalls in der Modelerstellung nachgedacht werden müssen.

Auch wenn in den oberen beiden Plots eine jährliche und monatliche Saisonalität gut erkennbar ist, zeigt der Verlauf dennoch eine hohe Volatilität (vmlt. durch die leichte wochenweise Autokorrelation, wie weiter oben beschrieben). Ein 2x8 Moving Average würde die wöchentliche Saisonalität glätten, ohne die monatliche Saisonalität zu sehr zu beeinflussen. Dazu wird das Merkmal *calls* nochmals modifiziert.

```
ts_sby <- ts_sby %>%
  mutate(
    MA8 = slider::slide_dbl(calls, mean,
                            .before = 3, .after = 4,
                            .complete = TRUE),
    MA2_8 = slider::slide_dbl(MA8, mean,
                              .before = 1, .after = 0,
                              .complete = TRUE)
        )

ts_sby %>%
  gg_season(MA2_8, period = "1y") +
  labs(title = "Saisonalitätsplot der Notrufe",
       subtitle = "2x8 Moving Average des Merkmals calls")
```



Das Ergebnis zeigt, wie erwartet, die monatliche und jährliche Saisonalität im direkten Vergleich der Jahre. Auch hier ist der Trend wiederum gut erkennbar.

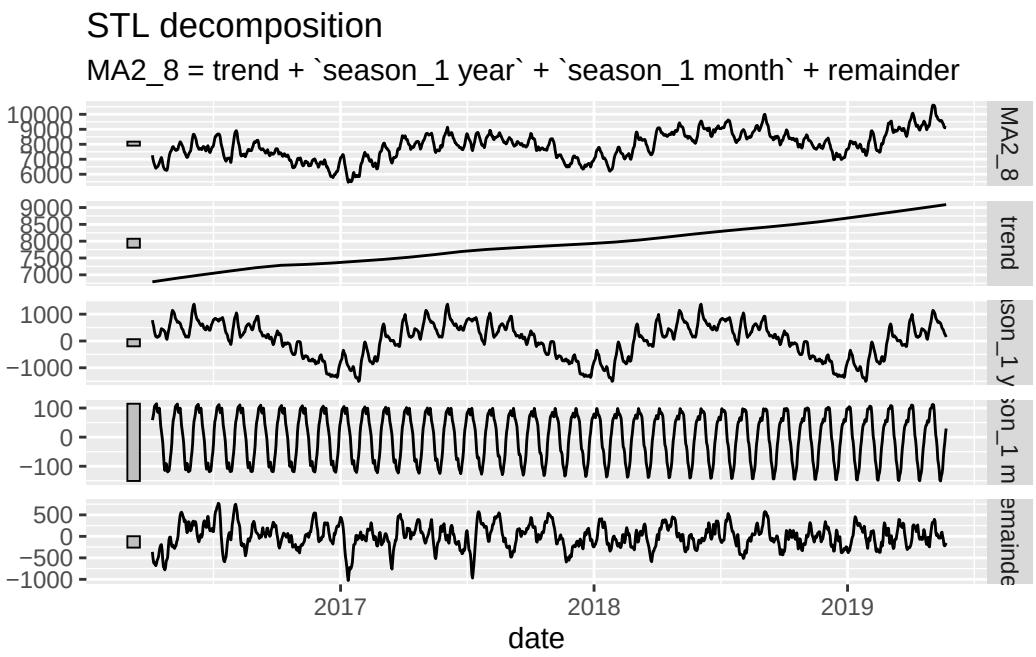
```
ts_sby %>%
  features(MA2_8,feat_spectral)
```

```
# A tibble: 1 x 1
  spectral_entropy
             <dbl>
1            0.494
```

Auch die spektrale Entropy des Merkmals wird durch den gleitenden Durchschnitt deutlich verbessert. Wie sehr darunter die Tauglichkeit als Prädiktor leidet, wird in der Modellerstellung zu evaluieren sein.

Decomposition von calls

```
ts_sby %>%
  filter(!is.na(MA2_8)) %>%
  model(
    STL(MA2_8 ~
          trend(window = 365) +
          season(period = "1 year", window = 540) +
          season(period = "1 month", window = 61))) %>%
  components() %>%
  autoplot()
```



Mit Hilfe der Methode “Seasonal and Trend decomposition using Loess” (STL), zeigt eine erste Dekomposition des Merkmals die besprochenen Komponenten. Der Trend ist nahezu linear, und die monatliche Saisonalität regelmäßig. In der Jährlichen Saisonalität scheint noch ein modifizierender Teil der Monatsperioden enthalten zu sein. Augenscheinlich könnte auch im *remainder*, dem Restanteil, noch eine Periodizität herauszuarbeiten sein. Dies wird jedoch ggf. die Aufgabe einer Modellerstellung.

Fazit

Der in der Merkmalsübersicht angesprochene zweite Ansatz zur Vorhersage von *sby_need* scheint aufgrund der explorativen Analyse am vielversprechensten. Bei diesem Ansatz wurde eine indirekte Vorhersage aufgrund der Saisonalität des Merkmals *calls* und der, ab einem gewissen Wert, nahezu linear anzunehmenden Korrelation mit *sby_need* untersucht, wobei letztere bewiesen durch *n_duty* beeinflusst ist. Die Qualität dieses Ansatzes wird in der Modellerstellung geprüft.

Für ein Benchmarking mit einem einfachen Baselinemodel kann möglicherweise auf Basis der Zielvariablen eine brauchbare Lösung gefunden werden,

Speichern aufbereiteter Daten

```
save(ts_sby, file = "../00_sample_data/02_processed/data_explorative.rda")
```

Literaturverzeichnis

Hyndman, Rob J., und George Athanasopoulos. 2021. *Forecasting: Principles and Practice*. 3. Auflage. https://otexts.com/fpp3/accuracy.html.
Kuhn, Max. 2019. *The caret Package*. https://topepo.github.io/caret/.