

Fallstudie

Model Engineering (DLMDWME01)

im Master Studiengang Data Science

**Vorhersagemodell für den täglichen Bedarf an Bereitschaftsfahrenden
beim DRK Berlin**

Verfasser: Georg Grunsky

Matrikelnummer: IU14072015

Tutorin: Nazil Andjic

Datum: 15. Juni 2025

Inhaltsverzeichnis

1	Einleitung	1
2	Projektorganisation	2
3	TDSP Projektabschlussbericht	5
3.1	Business Problem	5
3.2	Data Processing	5
3.2.1	Data Understanding	6
3.3	Modeling	9
3.3.1	Baseline Model	9
3.3.2	Advanced Model	10
3.3.3	Prophet Model	12
3.3.4	Modellresultate	14
3.4	Solution Architecture	15
3.5	Benefits	19
3.6	Learnings	20
4	Zusammenfassung	21
5	Literaturverzeichnis	22
6	Abbildungsverzeichnis	23
7	Tabellenverzeichnis	24
A	Annexes	25
A.1	Datenbeschreibung	25
A.2	explorative Datenanalyse	26
A.3	Baseline Model	41
A.4	Advanced Model	54
A.5	Prophet Model	66

1 Einleitung

Im gewählten Szenario für diese Fallstudie geht es um die Herausforderung, die Planungslogik des Bereitschaftsdienstplans für den Berliner Rotkreuz-Rettungsdienst zu verbessern. Ursprünglich aus dem Modul Model Engineering stammend (Pak 2024), wurde diese Aufgabenstellung, nach Rücksprache mit den jeweiligen Tutoren, für zwei weitere Module adaptiert und konnte so von der Use Case Analyse bis zur Umsetzung in thematisch aneinander gereihten Modulen bearbeitet werden.

1. **Data Science UseCase:** Analyse des Anwendungsfalls bis zur Präsentation für die Freigabe durch das Management.
2. **Technische Projektplanung:** Aufbauend auf dem PitchDeck vom Projektstrukturplan über eine Kostenschätzung und das Risikomanagement bis zur Stakeholder-Analyse.
3. **Model Engineering:** Die Umsetzung des UseCase als Projekt in einem Git-Repository mit bereitgestellten Daten bis zu einem fertiggestellten Vorhersagemodel.

Diese Arbeit behandelt den letzten Teil der Umsetzung dieses Anwendungsfalles.

Die Bearbeitung der Problemstellung aus mehreren Blickwinkeln hilft, ein durchgängiges Verständnis für die Abwicklung realer Szenarien zu erlangen. Die ganzheitliche Betrachtungsweise der Thematik lässt einen tieferen Einblick in die Prozesse der Data Science entstehen, erhöht die Identifikation mit dieser Aufgabenstellung und macht somit schlussendlich auch mehr Spaß. Da diese Arbeit den letzten der drei Abschnitte umfasst, kann an dieser Stelle bereits festgehalten werden, dass der Lerneffekt durch diese Vorgehensweise und Kombination mehrerer Module definitiv profitiert hat. Der Teufel steckt oft im Detail und, in dem Wissen, dass man, aufgrund der begleitenden Aufgabenstellung, von anfänglichen Fehlern und Unsauberkeiten wieder zu einem späteren Zeitpunkt eingeholt wird, beschäftigt man sich von Beginn an, intensiver mit dem Thema und versucht eben auch die Details zu lösen.

Bei dieser Aufgabe handelt es sich um ein Regressionsproblem in Zeitreihendaten und damit um die Vorhersage eines numerischen Wertes aufgrund historisch strukturierter Daten. Gelöst wurde die Aufgabe schlussendlich durch die lineare Abhängigkeit der Zielvariablen von einem modifizierten weiteren Merkmal, das besser durch saisonale Zeitreihenmuster zu beschreiben war.

2 Projektorganisation

Bei der Organisation des Projektes wurde auf Microsofts Team-Data-Science-Prozess zurückgegriffen. Dieser ist optimiert für agile Entwicklung und passt, unter anderem durch den definierten Anfang und das definierte Ende, besser zu diesem Projekt als CRISP-DM. Die flexiblere Gestaltung des Arbeitsprozesses durch Abbildung der Interdependenzen der großen „Überschriften“ des Projektlebenszyklus kam der Entwicklung ebenso zugute. Die Gestaltung des Arbeitsprozesses, vor allem mit weniger Erfahrung in diesem Bereich, wäre ohne häufigerem Wechsel zwischen den einzelnen Phasen nicht möglich gewesen.

Leider sind die Links zu TDSP, sowohl aus dem Studienscript, als auch von den referenzierenden GitHub-Seiten, auf Microsoft nicht mehr verfügbar. Das stattdessen angezeigte Dokument AI-Implementierung enthält zwar einen Verweis auf TDSP, zu dem Dokument selbst gelangt man jedoch nicht mehr. In diesem verhältnismäßig jungen Wirtschaftszweig folgen Weiterentwicklungen von Prozessen und Technologien Schlag auf Schlag.

Die Ordnerstruktur, die für die Organisation des Projektes angelegt wurde, orientiert sich an einer Azure TDSP Vorlage (Microsoft Azure 2023a), die zusätzlich um den Überbau einer Ordnerstruktur für eine gesamte Softwareentwicklung ergänzt bzw. erweitert wurde. Das Projekt steht weiters mit der gesamten bisherigen Entwicklung auf GitHub unter dem Link

<https://github.com/GGProjects/DLMDWME01>

zur Verfügung.

Die nachfolgende Abbildung zeigt die wesentlichen Elemente der bereitgestellten Ordnerstruktur. Auch wenn diese, in Teilen, für den Bedarf dieses Projektes zu umfangreich scheint, war die Intention, die Erstellung einer Vorlage für zukünftige Projekte zu schaffen, wo der Data Science Prozess in eine Softwareentwicklung eingebettet ist. Die einzelnen Modell- und Packageordner könnten auch als eigene Git-Repos angelegt werden. Die Nummerierung sorgt für gleichbleibende Anordnung und leicht zu verwaltende Strukturen.

Abb. 1: Struktur des Ordneraufbaus

```

DMLDWME01
|
+---00_docs                                # Dokumente
|   +---01_general_basics                  # unspezifische Grundlagen
|   +---02_project_basics                  # projektspezifische Grundlagen, zB:
|       +---Use Case Analyse               #   + Use Case Analyse
|       \---Projektplan                    #   + Projektplan
|   \---03_artifacts                       /* Projektprodukte
|       +---00_project_report               # Projektreport (diese Arbeit)
|       +---04a_model_standby              # Dokumentation eines Modells
|           +---01_dataanalysis             # Reports der Datenanalyse
|           \---02_modeling                 # Reports der Modellerstellung
|               \---[...models]             # Subfolder einzelner Modelle
|
+---01_data                                # Datenbereitstellung
|   +---01_lake                            # Data Lake – Rohdatenspeicher
|       |   \---01_use_case_2               # Rohdaten dieser Aufgabenstellung
|       |
|       \---02_warehouse                    # Data Warehouse – Aufbereitete Daten
|
+---02_config                             # Konfigurationsdateien
|
+---03_main                               # Hauptmodul der Anwendung
|
+---03[a-y]_...                           # einzelne Packages der Anwendung
|
+---03z_helpers                           # Hilfscode – wird bei Abschluss entfernt
|
+---04a_model_standby                     /* 04[a-z] Ordner der ML-Modelle
|   +---00_sample_data                     # Beispieldaten zur Dokumentation
|       |   +---01_raw                     # Rohdatenauszug
|       |   +---02_processed               # Auszug aufbereiteter Daten
|       |   \---03_for_modelling           # Auszug der Trainingsdaten
|       |
|       +---01_dataanalysis                 # Code explorativer Datenanalyse
|       +---02_modeling                     # Code der Modellerstellung
|       \---03_deployment                   # Code des Modeldeployments
|           \---[modelname].R              # projektspezifische Namenskonvention
|
+---05_output                             # Generierte Ergebnisse
|   +---01_documentation                   # Softwaredokumentation
|   +---02_data                            # Datenausgabe
|   +---03_figures                         # Grafiken
|   +---04_logs                           # Logausgaben
|   \---05_apps                           # User Interfaces
|
\---06_tests                              # Unittests

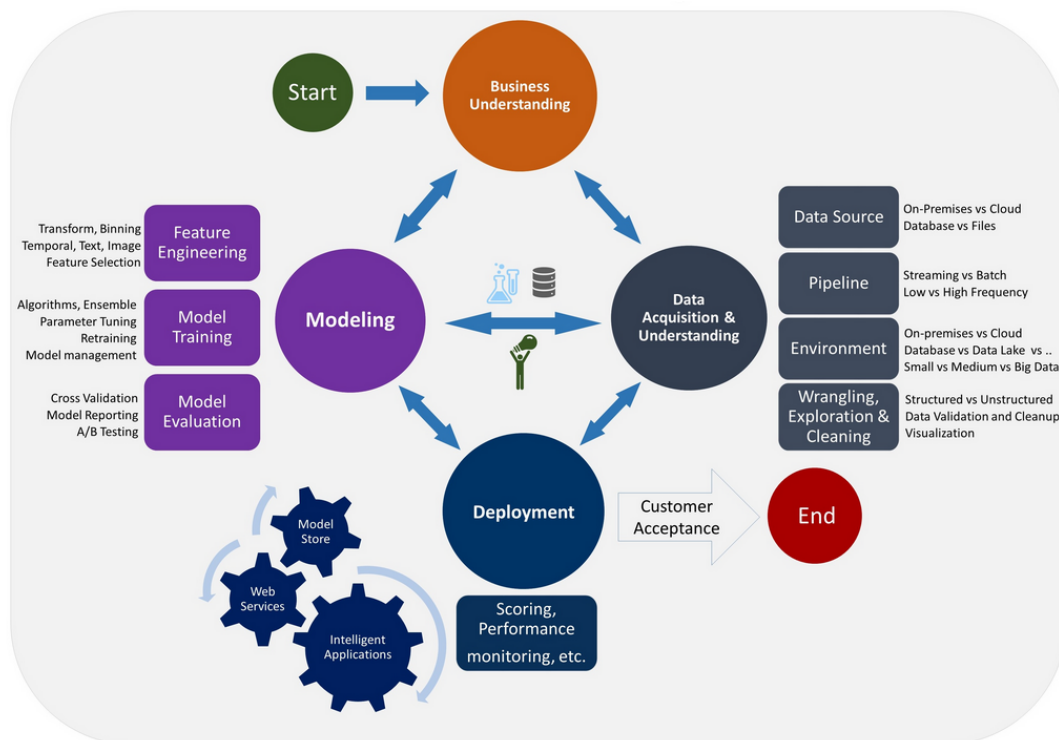
```

Quelle: Eigene Darstellung.

Die, in TDSP definierten Rollen und Aufgabenbereiche wurden, aufgrund der Projektgröße, nicht berücksichtigt, die benötigten Features waren gemäß Aufgabenstellung ebenso bereits vorgegeben. Der Use Case wurde in einem anderen Studiengangsmodul analysiert (Grunsky 2024). An der Stelle hätte es sich auch angeboten User Stories für die Implementierung der Features abzuleiten. The Use Case Handbook (microTool GmbH 2017) stellt dafür eine wunderbare Unterstützung dar. Zu diesem Zeitpunkt des Projektes lag der Fokus jedoch auf dem Machine Learning Canvas (Dorard 2022), der ebenfalls eine gute Unterstützung in diesem Bereich bietet. Für zukünftige Projekte wurde jedenfalls mitgenommen, die beiden Ansätze zu kombinieren, um im Vorfeld bereits die bestmögliche Vorbereitung für eine Projektplanung und Umsetzung zu erreichen.

In den nachfolgenden Abschnitten dieser Arbeit wird auf die einzelnen Phasen des Lifecycle eingegangen, wobei sich die Struktur des Dokuments an der Vorlage eines Projektabschlussberichtes nach TDSP orientiert. (Microsoft Azure 2023b)

Abb. 2: Darstellung des Team Data Science Prozesses



Quelle: TDSP Data Science Lifecycle (Microsoft Azure 2017)

3 TDSP Projektabschlussbericht

3.1 Business Problem

Die Aufgabenstellung fokussiert auf einen klar abgegrenzten Aufgabenbereich innerhalb des Berliner Rot Kreuz Rettungsdienstes, nämlich auf eine Vorhersage des Bedarfes an Bereitschaftspersonal für Einsatzfahrten. Das aktuelle Vorgehen sah, bis dato, eine fixe vorgehaltene Anzahl von 90 Bereitschaftseinsatzfahrer:innen für jeden Tag vor. Dieser Fixwert nimmt keine Rücksicht auf bestehende Saisonalitäten, eine Effizienzsteigerung soll daher durch einen Machine Learning Ansatz erzielt werden.

Die, in den Projekt-Grundlagendokumenten abgelegte, Use Case Analyse (Grunsky 2024) versucht, durch Anwendung des Machine Learning Canvas zusätzliches Business Understanding aufzubauen. So wurde an dieser Stelle zum Beispiel eine Vorgabe des Projektes, nämlich die Bereitstellung der Vorhersage bis zum 15. Tag des Monats, abgeändert und bereits am 10. Tag des Monats verlangt. Auch wurden an dieser Stelle weitere Datenquellen diskutiert, die eine Forecast des benötigten Bereitschaftspersonals möglicherweise unterstützen und verfeinern könnten. Diese Features wurden im Projektplan jedoch auf eine niedrigere Priorität gestuft und könnten Verbesserungen für zukünftige Weiterentwicklungen darstellen.

3.2 Data Processing

Die Vorlage des TDSP Abschlussberichts stellt in diesem Punkt, der im Prinzip der Phase Data Acquisition and Understanding entspricht, den originalen Datensatz im Vergleich zu dem, für das Modell, prozessierten Datensatz dar. Die Vorlage bezieht sich in diesem Punkt auf ein erstelltes Machine Learning Modell. Aus Sicht der, in diesem Modul, gewonnenen Erkenntnisse ist dieser Zugang unvollständig.

Es wurden im Zuge dieser Arbeit ein Baseline-Model erstellt und zwei etwas ausgefeiltere Modelle. Die Vorgehensweise der ersten Erstellung einer Benchmark war nicht nur eine Vorgabe der Aufgabenstellung, sondern ist eine gängige Praxis. Auf dem Weg zu einem fortschrittlicheren Modell ist es sehr wahrscheinlich auf zusätzliche Ideen zu stoßen, Zwischenlösungen auszuprobieren oder einfach nur zwei vielversprechende Ansätze zu verfolgen ohne zu wissen, welcher davon sich als der Bessere entpuppt. Das Datenschema des originalen Datensatzes wird im Data Dictionary (Anhang A.1) dargestellt und in Anhang A.2, der explorativen Analyse, näher untersucht.

Hierbei wurden mögliche Vorhersageansätze diskutiert, wobei zwei davon weiter verfolgt wurden. Die vorherzusagende Zielvariable ist dabei stets der tageweise Bedarf von Bereitschaftsfahrenden. Der erste Ansatz verfolgt dabei eine direkte Vorhersage der Zielvariablen auf Basis der eigenen Saisonalität. Im zweiten Ansatz wird die Korrelation der Zielvariablen zu den eingehenden Notrufen

genutzt. Letztere sind für eine Vorhersage durch eine saisonale Modellierung besser geeignet. Die Datenschemata, die für die einzelnen Modelle verwendet wurden, weichen geringfügig von einander ab. Sample-Daten der einzelnen Modelle wurden im Rahmen der Modellentwicklung in der Ordnerstruktur hinterlegt. In der zukünftigen Arbeit mit einer Datenvorhersage werden die Tagesdaten gemäß dem Eingangsdatenschema weiterhin durch die Planungsstelle des Bereitschaftsdienstes elektronisch eingegeben. Diese werden als Grundlage für neuerliche Trainings und Vorhersagen genutzt. Diese Rohdaten werden jedoch nicht unmittelbar nach der Eingabe prozessiert, sondern erst bei der Vorhersage. Das ermöglicht eine einfache Implementierung neuer Modelle ohne Änderung der Basisdaten.

Eine Erweiterung der Datengrundlage wurde in der Use Case Analyse erläutert. Die Idee dabei war es, die vorliegenden Unternehmensdaten des Roten Kreuzes mit weiteren Daten anzureichern um diese als Prädiktoren für eine Vorhersage der benötigten Bereitschaftsfahrer:innen zu verwenden. Im konkreten ging es dabei um eine Einbeziehung von Wetterdaten, insbesondere Wetterereignisse wie zB Schneefall oder überdurchschnittliche Hitze, Wochenend- und Feiertage, sowie Daten geplanter Veranstaltungen im Raum Berlin. Eine Untersuchung dieser Daten auf Tauglichkeit als Prädiktoren wäre in einer Fortsetzung des Projektes zur Verbesserung des Modelles denkbar.

3.2.1 Data Understanding

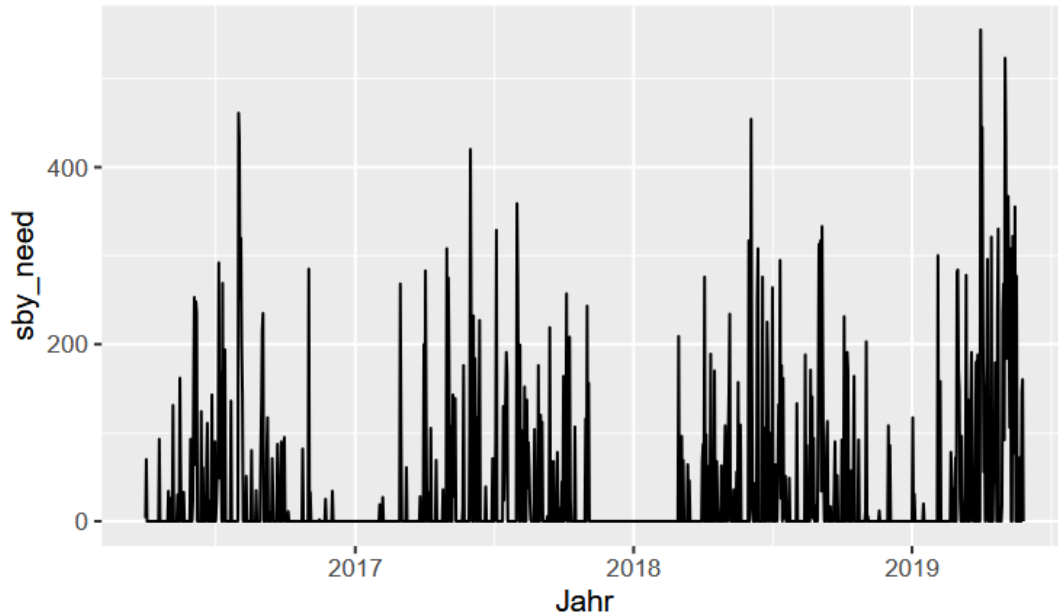
Der bereitgestellte Datensatz mit Messwerten des Einsatzfahrdienstes erstreckt sich über einen Zeitraum von etwa drei Jahren (von 01.04.2016 bis 27.05.2019) und enthält 1152 Datenpunkte zu je acht Merkmalen.

Tab. 1: Struktur der vorhandenen Daten

Variable	Datentyp	Beschreibung
...1	Numeric (double)	Zähler
date	Date (%Y-%m-%d)	Datum
n_sick	Numeric (double)	Anzahl erkrankte Einsatzfahrer:innen
calls	Numeric (double)	Anzahl der Notrufe
n_duty	Numeric (double)	Anzahl Einsatzfahrer:innen im Dienst
n_sby	Numeric (double)	Anzahl verfügbares StandBy Personal
sby_need	Numeric (double)	Anzahl aktiviertes StandBy Personal, Zielvariable
dafted	Numeric (double)	Zusätzlich aktiviertes Personal, da sby_need zu niedrig

Die erste Spalte (*id*), die bisher konstante Einteilung des Bereitschaftspersonals (*n_sby*), sowie die Anzahl der zusätzlich benötigten Fahrer:innen (*dafted*) wurden gleich zu Beginn wieder entfernt und die Daten in Tageszeitreihendaten konvertiert. Das Datum dient dabei als Schlüssel zur eindeutigen Identifikation des Datenpunktes.

Abb. 3: Zeitlicher Verlauf der Zielvariablen



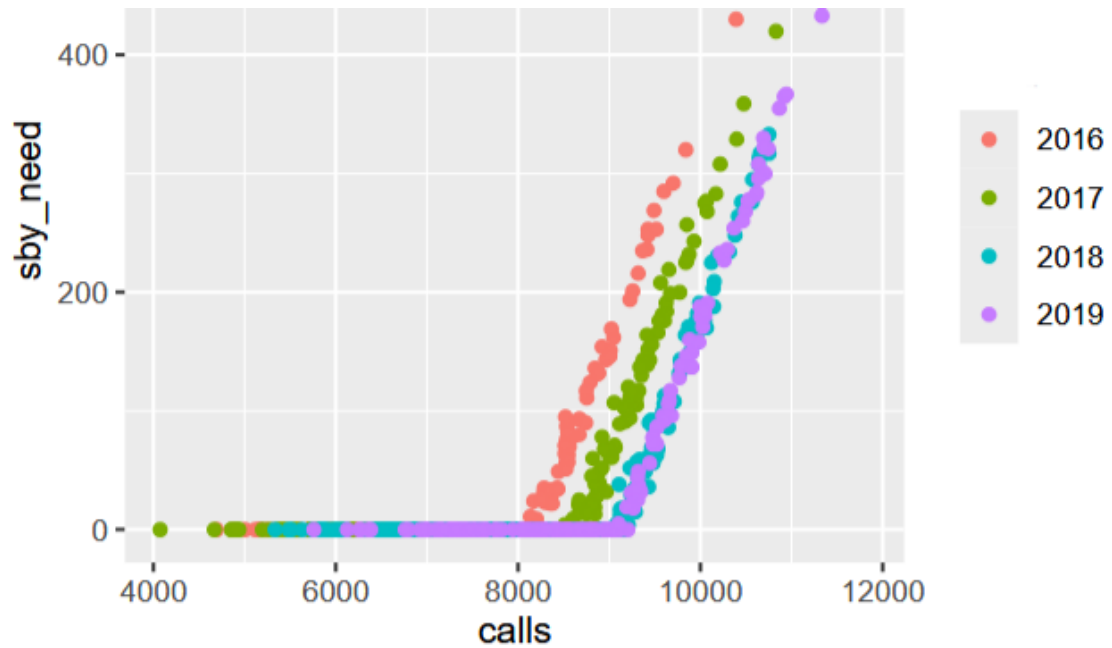
Quelle: Eigene Darstellung.

Die vermutete Saisonalität ist in der zeitlichen Darstellung des Bedarfs an Bereitschaftsfahrenden 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 diesem Zeitpunkt deutlich höhere Spitzen auf. Weiters werden auch die raschen Wechsel zwischen wenig bis kein Bedarf und sehr hoher Bedarf in der Darstellung deutlich. Diese Volatilität verlangt eine überlegte Generalisierung des Modells und einen klar definierten Fokus der Vorhersage. Gemäß Aufgabenstellung ist dieser auf eine Abdeckung der Spitzenwerte ausgerichtet.

Die Korrelation zwischen der Zielvariablen und dem Merkmal der eingegangenen Notrufe wurde bereits angesprochen. Die unten angeführte Grafik macht den, nahezu linearen, Zusammenhang anschaulich.

Die Abstände im augenscheinlichen Intercept (dem Schnittpunkt mit der x-Achse) betragen zwischen den Jahren jeweils etwa 500 Notrufe. Das ist jeweils das fünffache des jährlichen Anstiegs von diensthabendem Personal. Von 2018 auf 2019 wurde das diensthabende Personal nicht erhöht, weswegen auch im Intercept der linearen Korrelation keine Veränderung erkennbar ist. Wurden weniger Anrufe als der jeweilige Intercept getätigt wurde auch kein Bereitschaftspersonal aktiviert.

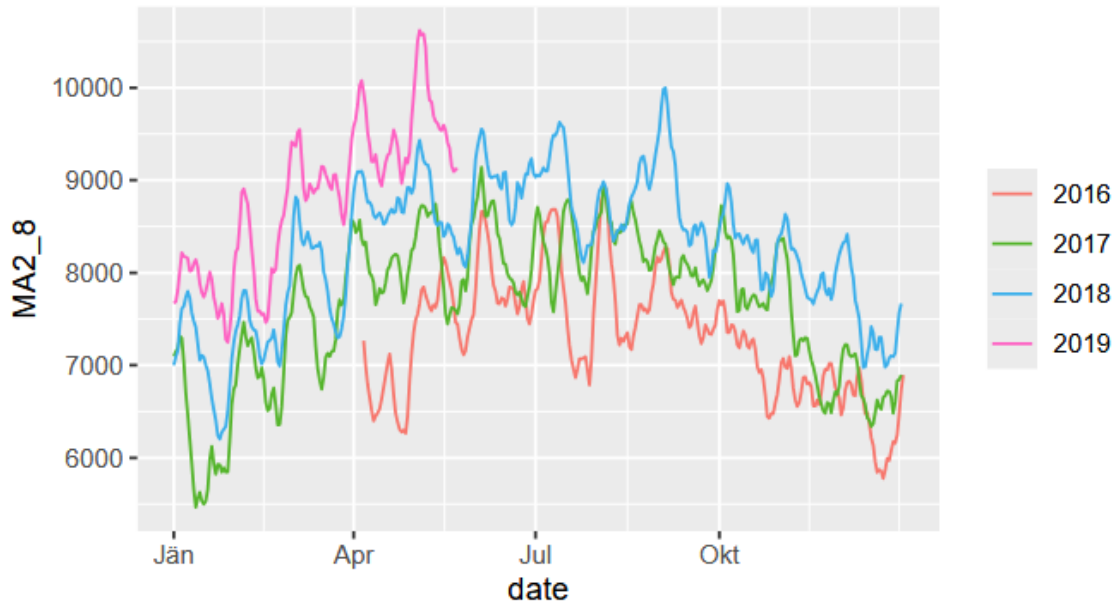
Abb. 4: Korrelation der Zielvariable zu den Notrufen



Quelle: Eigene Darstellung.

Die, auf der Shannon Entropy basierende, spektrale Entropy liefert eine erste Aussage über die Verteilung und Vorhersagbarkeit von Zeitreihendaten. Im vorliegenden Datensatz wurde der beste Entropy-Wert für den gleitenden Durchschnitt der eingehenden Notrufe erzielt. Dieser wird im nachfolgenden jährlichen Saisonalitätsplot dargesellt. Die, in den Daten auftretenden, Muster sind hier gut erkennbar, genauso wie ein augenscheinlicher Aufwärtstrend über die Jahre.

Abb. 5: Verteilung der Notrufe nach Jahr



Quelle: Eigene Darstellung.

Die beschriebene Korrelation der Zielvariable zu den eingegangenen Notrufen (unter Berücksichtigung des diensthabenden Personals) sowie die besser modellierbare Saisonalität der eingegangenen Notrufe dienen als Grundlage für die Entwicklung fortschrittlicherer Modelle.

3.3 Modeling

Die Entwicklung der Modelle wurde mit der Programmiersprache R umgesetzt. Besonders hilfreich war hierbei das online Buch sowie das dazugehörige R Package „Forecasting Principles and Practice“ (Hyndman / Athanasopoulos 2021). Für die Entwicklung der Modelle wurden, wie bereits beschrieben, zwei Ansätze gewählt. Im Vordergrund stand dabei stets die Anforderung, nie zu wenig Bereitschaftspersonal vorherzusagen. Gerade bei den vorliegenden, sehr stark und abrupt schwankenden Eingangsdaten der Zielvariablen, ist die Herausforderung, dass man zwischen Kosteneffizienz und Abdeckung der Maximalbedarfsspitzen wählen muss.

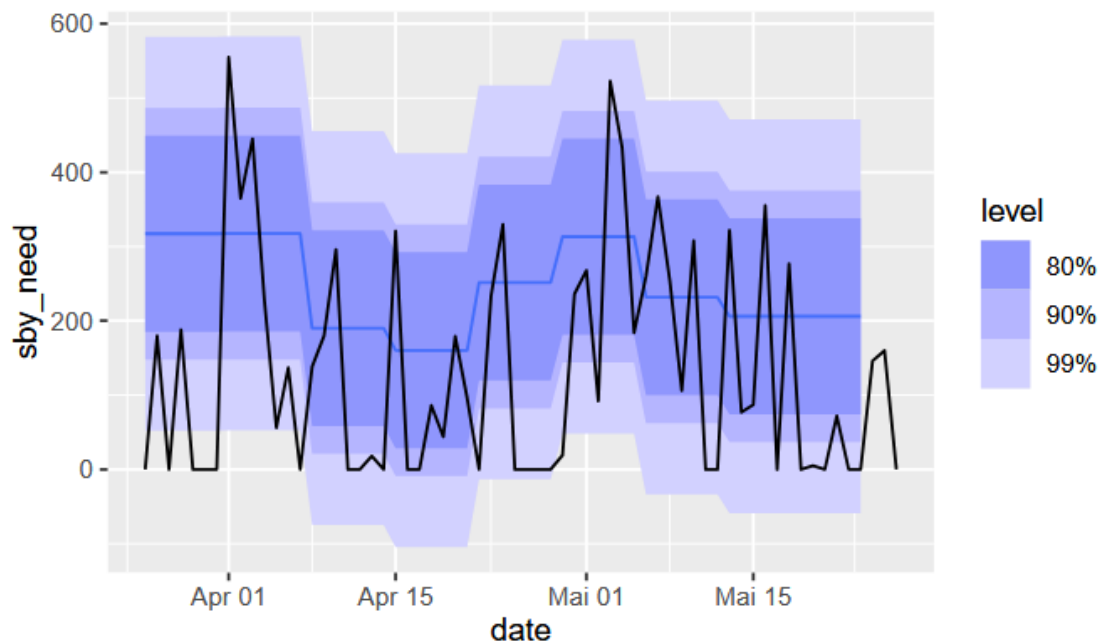
In der explorativen Datenanalyse (Anhang A.2) wurde festgehalten, dass 35 Personen in Bereitschaft für Tage mit niedrigem Bedarf einen guten Durchschnittswert bilden. Die Einteilung von null Personen in Bereitschaft scheint geschäftsseitig unrealistisch, der Minimalwert wird daher für alle Modelle mit 35 Personen festgelegt.

3.3.1 Baseline Model

Das Baseline-Model (Anhang A.3) wird direkt anhand der Saisonalität der Zielvariablen trainiert und verwendet dabei ein Lineares Model für Zeitreihen (TSLM), das auch ebenso Trends und saisonale Muster abbilden kann. Um eine ausreichende Generalisierung zu erreichen und die Vorhersage zuverlässiger zu machen, wurden die Daten mit einer, über sieben Tage, gleitenden 99% Quantile geglättet und anschließend zu wöchentlichen Daten aggregiert. Bei der Aggregation wurden die

jeweiligen Maxima der gleitenden Quantile als Wochenwert herangezogen. Nach der Vorhersage, werden die Wochendaten wieder in Tagesdaten (gem. Anforderung) aufgetrennt, wobei die Tage einer Woche entsprechend immer denselben Wert aufweisen. Die Bedarfsspitzen werden durch dieses Model nicht korrekt vorhergesagt, die Saisonalität wird jedoch modelliert. In der Darstellung werden die Vorhersagen (blau) den Daten des Testdatensatzes (schwarz), über einen Zeitraum von zwei Monaten, gegenübergestellt. Des Weiteren werden für die Vorhersage auch die 80%, 90% und 99% Konfidenzintervalle angezeigt.

Abb. 6: Vorhersage des Basline-Models



Quelle: Eigene Darstellung.

Für die spätere Evaluierung wurde, als zweites Baseline-Model, auch das aktuelle Vorgehen mit einem Fixwert von 90 Bereitschaftsfahrenden mit aufgenommen.

3.3.2 Advanced Model

Das Advanced-Model (Anhang A.4) basiert auf dem Modell „Seasonal and Trend decomposition using Loess“ (STL). Dieser Algorithmus wurde bereits in der explorativen Datenanalyse (Anhang A.2) verwendet um die saisonalen Komponenten der eingehenden Notrufe auf Basis eines gleitenden Durchschnitts zu untersuchen. Dabei werden die Daten in den Trend, verschiedene Saisonalitäten und den „Remainder“ (den Rest) aufgesplittet. So können saisonale Muster in den Daten gut erkannt, analysiert und beschrieben werden. In diesem Ansatz wird jedoch nicht die Zielvariable direkt vorhergesagt sondern die, schöner modellierbare, Saisonalität der eingehenden Notrufe. Für die weitere Vorhersage des Bedarfs an Bereitschaftspersonal wird die beschriebene lineare Korrelation zwischen den eingehenden Notrufen und dem StandBy-Bedarf genutzt. Diese wurde durch ein einfaches lineares Modell abgebildet.

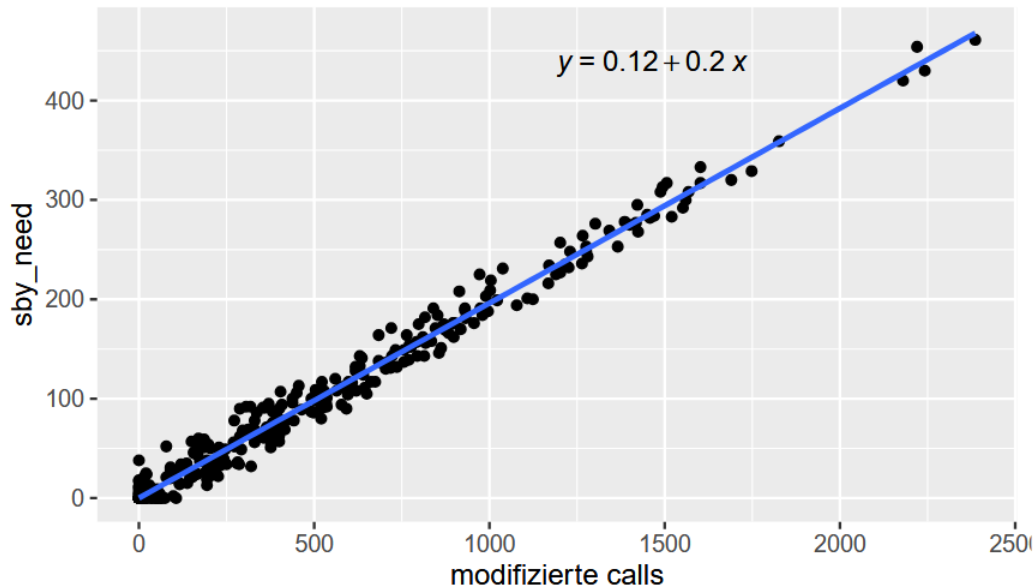
Die, in Punkt 3.2.1 Data Understanding, festgestellte Auswirkung der jährlichen Erhöhung des diensthabenden Personals mit einem Faktor 5 auf den Intercept der Korrelation wird im linearen Modell berücksichtigt. Bei der Anwendung des Modells muss daher die Anzahl des diensthabenden Personals

als Parameter mitgegeben werden.

$$\text{modifizierteNotrufe}_t = \text{Notrufe}_t - \text{Korrekturwert}_t \mid \text{Korrekturwert}_t = (n_duty_{year} - 1700) * 5$$

Der Intercept wird nach Korrektur fix auf 8150 gesetzt. Werden weniger Notrufe vorhergesagt wird durch das Model der fixierte Minimalwert von 35 Personen in Bereitschaft ausgegeben. Die u.a. Grafik zeigt die modifizierte lineare Abhängigkeit zwischen den eingehenden Notrufen und dem StandBy Bedarf mit dem fixen Intercept bei 8150.

Abb. 7: Korrelation modifizierter Notrufdaten mit der Zielvariablen

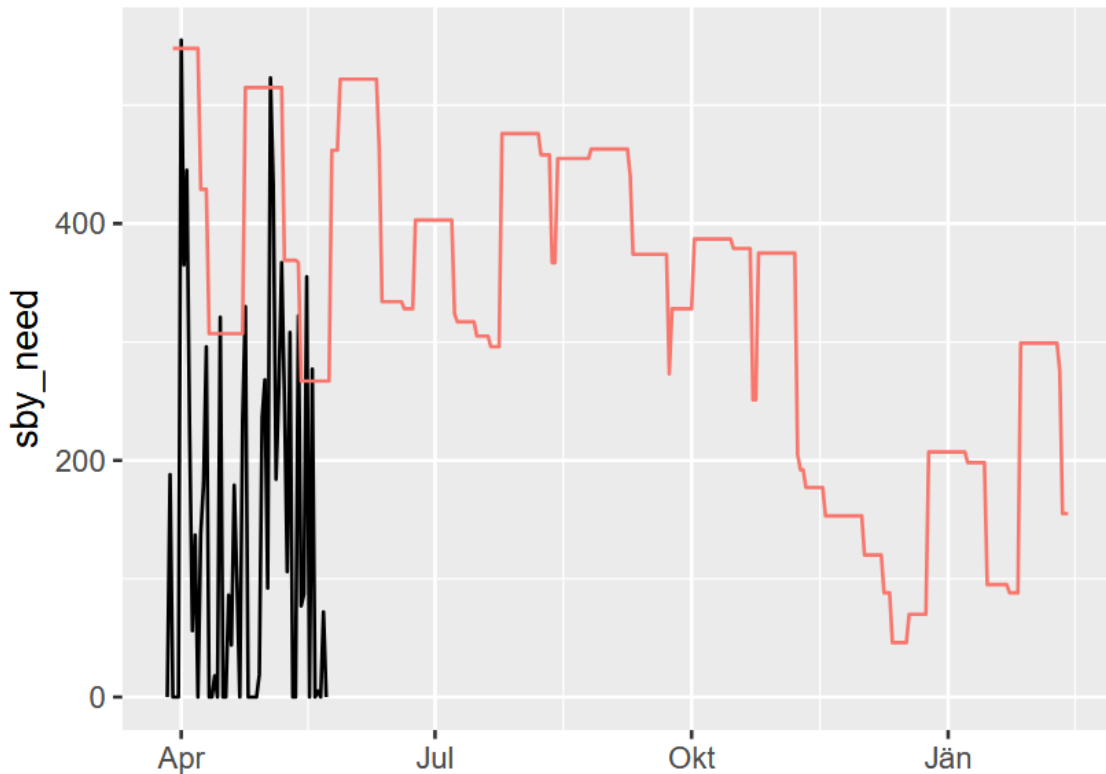


Quelle: Eigene Darstellung.

Der STL Algorithmus erkennt zwar die Spitzenaufkommen in den eingehenden Notrufen, im Vergleich mit dem Testdatensatz wurden diese allerdings oft wenige Tage verschoben prognostiziert. Die, allgemein fehlende, Generalisierung der Vorhersage wird durch Anwendung eines gleitenden Maximums auf die Prognose erreicht.

Das Advanced Model wurde in zwei Varianten evaluiert. In den finalen Programmcode wurde allerdings nur die Version 2 aufgenommen. Der Unterschied der beiden Modelle liegt in der Fenstergröße des gleitenden Maximums und (in der Version 2) einer Reduktion des Parameters „diensthabendes Personal“ um einen Wert von 100. Durch die Modifikation dieses Parameters kann die Höhe des vorhergesagten Bedarfs beeinflusst werden.

Abb. 8: Vorhersage Advanced Model V2



Quelle: Eigene Darstellung.

Die Darstellung zeigt die Ausgabe der Vorhersage (rot) über den Zeitraum von elf Monaten. Die Werte des Testdatensatzes (schwarz) sind nur in den ersten beiden Monaten der Prognose vorhanden. Die jährliche und monatliche Saisonalität werden von dem Modell gut aufgenommen. Durch die stark ausgeprägte Generalisierung wird jedoch überwiegend eine deutlich zu hohe Anzahl an Bereitschaftspersonal vorgesehen.

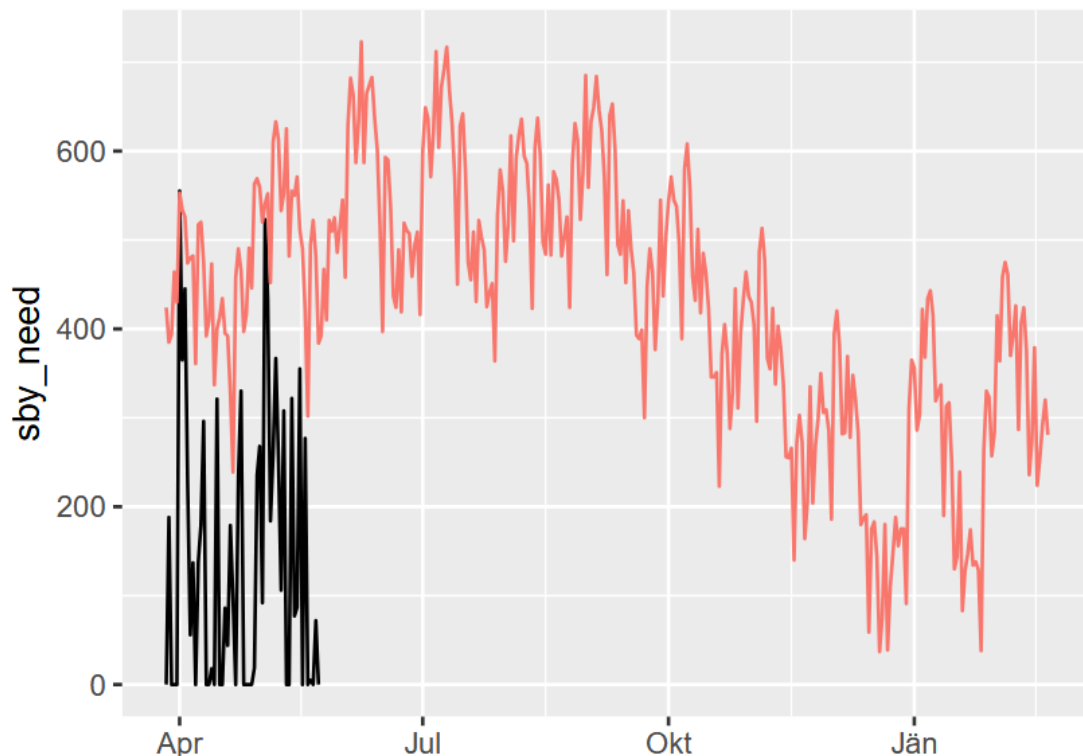
3.3.3 Prophet Model

In der Use Case Analyse wurde der Algorithmus „Facebook Prophet“ als möglicherweise passender Ansatz für die gegenständliche Machine Learning Aufgabe recherchiert. Dieser Algorithmus wurde für Zeitreihenvorhersagen entwickelt und verwendet dabei ein zusammengesetztes Modell aus Trend, Saisonalität und speziellen Ereignissen in Zeitreihen. (Taylor / Letham 2017). Der Vorteil von Facebook Prophet, oder kurz *Prophet*, liegt darin, dass in einer möglichen, zukünftigen Weiterentwicklung des Modells auch Tage mit besonderen Ereignissen, wie zB Veranstaltungen und speziellen Wetterverhältnissen, mit modelliert werden können. Diese würden dann a priori in eine Vorhersage miteinbezogen und in der Prognose berücksichtigt werden.

Im Advanced-Modell wurde versucht einen Puffer für die Spitzenwerte anhand eines gleitenden Maximums über die vorhergesagten Notrufe zu erreichen. Bei Prophet erscheint die Verwendung von Konfidenzintervallen möglich. Daher wird in diesem Ansatz der Puffer durch Verwendung der 90% Quantile als Vorhersagewert erreicht. Das Schöne hierbei ist, dass der zu verwendende Prozentsatz für die Berechnung im codierten Modell als Hyperparameter gesetzt werden kann. Wie auch im Advanced Modell V2 wird, zum Vergleich der Ergebnisse, die Höhe des diensthabenden Personals um

100 reduziert. Damit wird die Anzahl des vorhergesagten Bereitschaftspersonals etwas angehoben.

Abb. 9: Vorhersage Prophet Model



Quelle: Eigene Darstellung.

Die Darstellung zeigt wiederum die Ausgabe der Vorhersage (rot) über den Zeitraum von elf Monaten mit gleichzeitiger Anzeige der Testdaten (schwarz) für die ersten beiden Monate der Prognose. Bei Anwendung dieses Modells sind die einzelnen Komponenten der Saisonalität in der Vorhersage zu erkennen. Man sieht sowohl den jährlichen Verlauf mit einem Abfall des Bedarfs bis zum Jahreswechsel, als auch ein augenscheinlich relativ fixes Muster in den Monatsdaten. Der Grad der Flexibilität in den saisonalen Mustern kann in Prophet durch Hyperparameter gesteuert werden. In diesem Bereich bietet Prophet auch die meisten Möglichkeiten das Modell noch feiner zu „tunen“. Das Machine Learning Framework *Caret* (Kuhn 2019) bietet Unterstützung im Finetuning durch ein automatisiertes „Durchtesten“ mehrerer Kombinationen an Hyperparametern. Das war in dem, hier verwendeten, Package *fpp3* (Hyndman / Athanasopoulos 2021) nicht möglich, dafür ist dieses speziell für die Arbeit mit Zeitreihen entwickelt. In einer Weiterentwicklung des gegebenen Anwendungsfalles wäre ein Hyperparameter-Finetuning für den Prophet-Ansatz jedenfalls zielführend.

Dank Verwendung der 90% Quantile für die Prognose, wird auch bei diesem Modell überwiegend eine zu hohe Anzahl an Bereitschaftspersonal vorgesehen. Die Höhe der Vorhersage ist, wie auch beim Advanced Model, über gesetzte Parameter beeinflussbar und sollte mit den Entscheidungsträgern noch diskutiert werden.

3.3.4 Modellresultate

Tab. 2: Vergleich der ML-Modelle

Modell	RMSE	MAE	MeanError	% low	sum low	mean low	max low
Baseline	185,94	164,48	108,64	26,23	1703,19	106,45	237,44
Aktuelles Vorgehen	160,05	125,59	46,38	47,54	5245	180,86	465
Advanced Version 1	190,31	153,36	88,47	27,12	1914	119,62	271
Advanced Version 2	311,91	270,59	266,05	8,93	127	25,4	88
Prophet (Q90)	356,43	328,61	328,03	1,61	18	18	18
Prophet (Q80)	299,07	268,85	264,82	3,23	125	62,5	81

Quelle: Eigene Darstellung.

Die angeführte Tabelle zeigt einen Vergleich der beschriebenen Modelle (inklusive des aktuellen Vorgehens) anhand unterschiedlicher Messwerte. Diese sind folgendermaßen zu interpretieren:

- **RMSE:** Gängiger Messwert. Berechnet die Wurzel der mittleren quadrierten Abweichung
- **MAE:** Gängiger Messwert. Berechnet die durchschnittliche absolute Abweichung
- **MeanError:** Die durchschnittliche Abweichung. Positive Werte deuten auf eine durchschnittliche Überschätzung durch das Modell hin
- **% low:** Prozentanteil der Vorhersagen, wo das Modell zu niedrig geschätzt hat
- **sum low:** Die Summe zu niedrig geschätzten Personals, gleichzeitig Personal, das zusätzlich hätte aktiviert werden müssen.
- **mean low:** Der Durchschnitt jener Schätzungen, die unter dem Bedarf lagen.
- **max low:** Die maximale Abweichung jener Schätzungen, die unter dem Bedarf lagen.

Die, in der obigen Tabelle, dargestellten Messwerte beziehen sich auf eine Gütebewertung der Modelle in einem Testzeitraum von zwei Monaten. Hierbei wird ersichtlich, dass beim bisherigen Vorgehen etwa bei der Hälfte der Tage zu wenig Bereitschaftspersonal vorgesehen wurde. In den zwei Monaten Testzeitraum hätten in Summe 5245 zusätzliche Bereitschaftsfahrer:innen aktiviert werden müssen. Hinsichtlich einer durchschnittlichen Abweichung (RMSE, MAE oder MeanError) schneidet das aktuelle Vorgehen jedoch am Besten ab. Eine erste Verbesserung in Bezug auf Unterschätzungen des Bedarfs an Bereitschaftspersonal zeigen das Baseline Modell und das Advanced Modell in der Version 1. Die beiden Modelle liefern ähnliche Messwerte und schaffen es Unterschätzungen auf ein Viertel der Vorhersagen zu senken, wohingegen RMSE und MAE noch mit dem aktuellen Vorgehen vergleichbar bleiben. Das Advanced Modell in Version 2, sowie zwei Varianten des Prophet Modells (einmal mit 80% Konfidenzintervall und einmal mit 90% Konfidenzintervall) steigen bei RMSE und MAE deutlich in die Höhe, was sich, im Gegenzug, in einer ebenso deutlichen Verringerung der Fälle von Unterschätzung widerspiegelt.

Auf welches Modell, die Entscheidung schlussendlich fällt ist abzuwägen und wird mit den Entscheidungsträgern diskutiert werden müssen. Durch „Finetuning“ von Hyperparametern ist gegebenenfalls noch eine Verbesserung einzelner Modelle zu erreichen. Derzeit sind die Modelle „Baseline“, „Advanced V2“ und „Prophet Q90“ ausprogrammiert.

Für alle Modelle ist die Rechendauer als sehr positiv zu bewerten. Die Zeiten für Training und Vorhersage betrugen, in der Entwicklung, beim Baseline Modell in Summe 0,28 Sekunden, bei den Advanced Modellen in Summe 0,97 Sekunden und den Prophet Modellen in Summe 13,60 Sekunden. Für die Entwicklung wurde ein Notebook vom Typ Dell Latitude 5490, Intel i7 vPro (8th Generation) mit 16 Gb RAM (aus dem Jahr 2018) verwendet. Für eine Umsetzung der modellbasierten Vorhersage im Bereitschaftsdienst ist daher nicht mit großen Anschaffungskosten hinsichtlich Hardwarekapazitäten zu rechnen.

Fehleranalyse

Die Schwachstellen der vorgestellten Modelle, liegen, wie bereits im Vergleich erkennbar, in der Unfähigkeit sowohl niedrigen als auch hohen Bedarf korrekt vorherzusagen. Hier wird eine Entscheidung getroffen werden müssen, worauf der Fokus liegen soll. Eine korrekte Vorhersage von Tagen mit niedrigem Bedarf spart Bereitschaftskosten, wohingegen der Einsatz von zu wenig Bereitschaftsfahrenden möglicherweise einen Reputationsschaden nach sich ziehen kann. An dieser Stelle wäre es hilfreich mit den Entscheidungsträgern gemeinsam eine Use Case spezifische Kostenfunktion für die Bewertung der Modelle zu entwerfen. Diese kann dann die tatsächlichen Kosten der Vorhersage ermitteln.

Weiters ist es für Modelle des maschinellen Lernens schwierig, allgemeine Änderungen im Muster der Daten (Datendrift) abzubilden. Beim vorliegenden Datensatz ist mit dem Jahreswechsel 2018/2019 eine solche Änderung erkennbar. Diese fällt in einen Zeitraum, der bereits die Testdaten für die Gütebewertung der Modelle beinhalten und daher nicht für das Training herangezogen werden sollte. Das Sammeln weiterer Trainingsdaten, die das Aufkommen nach 2019 beschreiben könnten die Modellgüte maßgeblich verbessern. Bei den vorgestellten Modellen, ist der Prophet Algorithmus auch in der Lage, vorgegebene *Changepoints* für Änderungen im Trend zu berücksichtigen. So könnten zB auch maßgebliche Trendwenden wie COVID-19 modelliert werden.

3.4 Solution Architecture

Die Implementierung des Vorhersagemodells in die tägliche Arbeit der Planungsstelle könnte dermaßen erfolgen, dass die Mitarbeitenden weiterhin täglich die Betriebsmetriken (wie auch im vorliegenden Datensatz), über ein Formular festhalten. Dieses kann, wie in der unteren Grafik dargestellt auch in Form einer Web-Applikation bereitgestellt werden.

Am 10. Tag jedes Monats wird über eine weitere Seite der Web-Applikation das Training und die Vorhersage für einen angegebenen Zeitraum (zB zwei Monate) angestoßen und die Mitarbeiter:innen erhalten die prognostizierten Werte als Liste und grafisch ausgegeben.

Abb. 10: mögliches Dateneingabe-GUI

The image shows a web application interface for 'Einsatzfahrten' (Emergency Trips). The interface has a red header bar with the title 'Einsatzfahrten' and a hamburger menu icon. Below the header, there is a dark grey sidebar with a pencil icon and the text 'Dateneingabe'. The main content area is white and features a red header for the form titled 'Neuen Eintrag hinzufügen'. The form contains several input fields with labels: 'Datum' (containing '2025-06-02'), 'Anzahl diensthabender Einsatzfahrer:innen' (containing '1900'), 'Anzahl der Krankmeldungen' (containing '0'), 'Anzahl eingegangener Notrufe' (containing '0'), 'Anzahl des eingeteilten Bereitschaftspersonals' (containing '0'), 'Bedarf an Bereitschaftspersonal' (containing '0'), and 'Zusätzlich aktivierte Fahrer:innen' (containing '0'). At the bottom of the form is a red button labeled 'Eintrag speichern'. On the left side of the sidebar, there is a red button with a checkmark icon and the text 'Vorhersage StandBy'.

Einsatzfahrten

Dateneingabe

Vorhersage StandBy

Neuen Eintrag hinzufügen

Datum

2025-06-02

Anzahl diensthabender Einsatzfahrer:innen

1900

Anzahl der Krankmeldungen

0

Anzahl eingegangener Notrufe

0

Anzahl des eingeteilten Bereitschaftspersonals

0

Bedarf an Bereitschaftspersonal

0

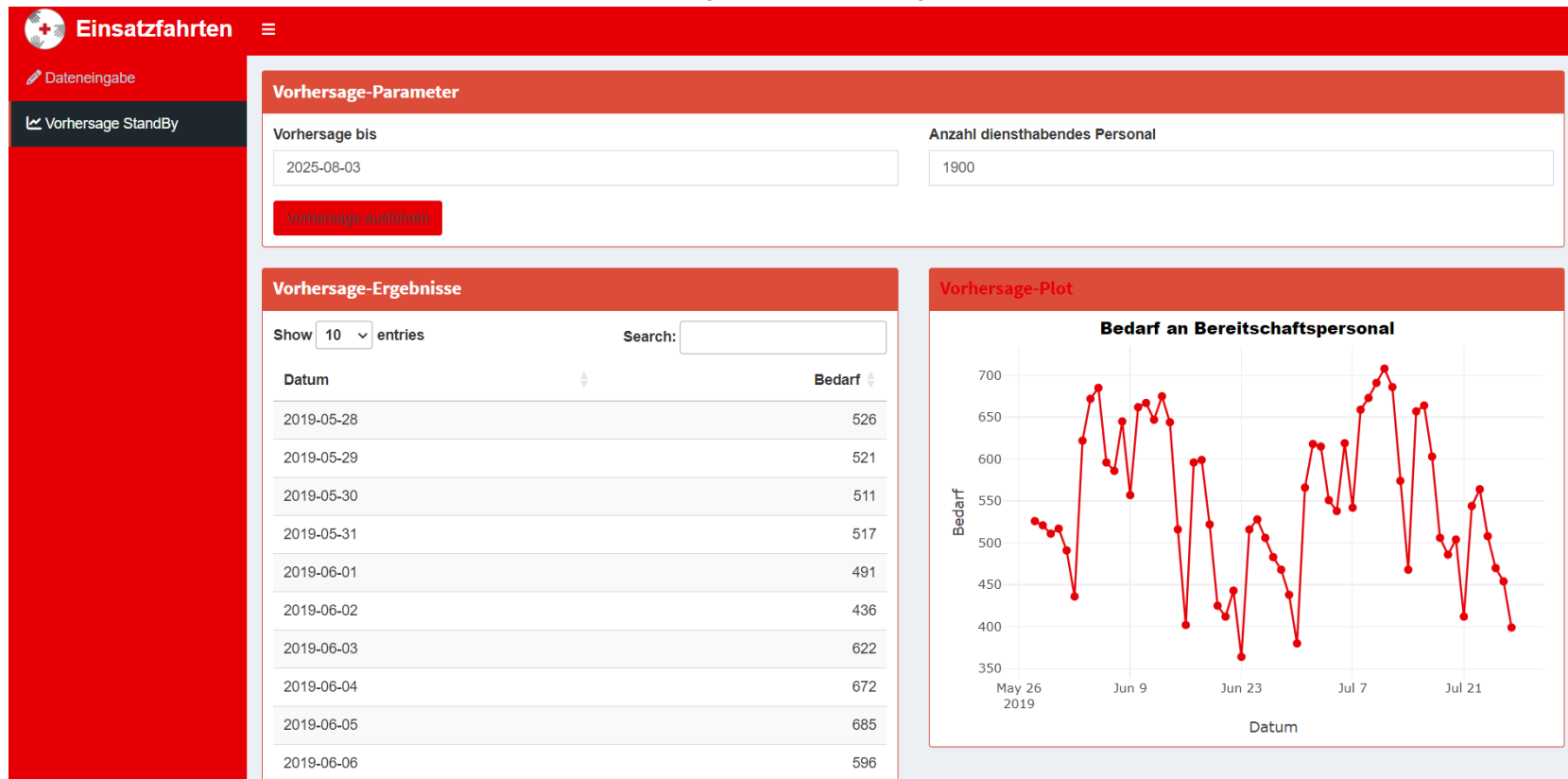
Zusätzlich aktivierte Fahrer:innen

0

Eintrag speichern

Quelle: Eigene Darstellung.

Abb. 11: mögliches Datenausgabe-GUI



Quelle: Eigene Darstellung.

Die Planungsstelle hat nun noch weitere fünf Tage Zeit, diese Werte

1. auf Plausibilität zu prüfen und eigene Erfahrung mit einfließen zu lassen. Prognostizierte Werte dürfen vom Personal der Planungsstelle abgeändert werden, das Bedarf allerdings klaren Regeln und einer hinreichenden Dokumentation, wie in Punkt Customer Acceptance (3.5) noch erläutert wird.
2. mit Namen von Bereitschaftspersonal zu hinterlegen und so einen Bereitschaftsdienstplan zu erstellen.

Das Format der Ausgabe durch die Web-Applikation kann noch den Wünschen Planungspersonals angepasst werden. Zur nachvollziehbaren Modellbewertung sollte die Prognose nicht veränderbar (read-only) gespeichert werden. Abänderungen durch die Planungsstelle müssen gesondert dokumentiert und abgelegt werden.

Die derzeit im Repository verfügbare Web-Applikation bildet die Grundfunktionalität des beschriebenen Ablaufes dar. Dennoch sind hier noch einige Weiterentwicklungen und Verbesserungen für einen Betrieb vorzusehen. Hierzu gehören zum Beispiel die Änderung und Bearbeitung bereits angelegter „Betriebsmetriken“, oder auch die angesprochene Speicherung und Dokumentation von Prognosemodifikationen durch die Planungsstelle. Weiters wäre es sinnvoll, zusätzlich zu Training und Vorhersage, auch einen Trainings- und Testlauf, mit allen verfügbaren Modellen, anhand der gespeicherten historischen Daten durchführen zu können (zB durch Aktivieren einer Check-Box) und die Modellbewertungen analog zu Tabelle (2), oder mit eigener Kostenfunktion, anzeigen lassen zu können. So kann in periodischen Abständen überprüft werden ob die Vorhersagealgorithmen in einem zuverlässigen Rahmen arbeiten, oder ob sich die Güte einzelner Modelle (zB aufgrund eines Datendriffs) verschlechtert.

Die Auswahl des anzuwendenden Vorhersagemodells wurde in der Web-Applikation bewusst weggelassen. Die Mitarbeitenden der Planungsstelle sollen sich nicht damit beschäftigen müssen, welches Modell, wie funktioniert und auch nicht begründen, warum sie welches Modell gewählt haben. Es wäre jedoch keine Schwierigkeit ein solches Feature in die Applikation zu implementieren, da beim Training derzeit das Prophet Modell standardmäßig beim Funktionsaufruf als Parameter mitgegeben wird, die Funktion aber so gestaltet ist, dass durch Veränderung dieses Parameters jedes implementierte Modell gewählt werden kann.

Für die Implementierung neuer Modelle ist demnach auch einfach nur der Code für das Modell, analog zur Struktur und Namenskonvention bestehender Modelle im Ordner *04a_model_standby* *03_deployment* abzulegen und in der Applikation mit dem Parameter des Modellnamens aufzurufen. Sollten neue Modelle weitere zusätzliche Hyperparameter benötigen sind diese jedoch applikationsweit zu ergänzen.

Die, diesem Projekt vorangegangene, Use Case Analyse sieht für die geschäftszentrierte Messung des Erfolgs ein Monitoring der Vorhersagemodelle anhand von zwei Key Result Indikatoren (KRI) vor. Diese sind zwar nicht ausschließlich von der Arbeit der Vorhersage abhängig, spiegeln jedoch die Messung des, durch Implementierung der Modelle, gewünschten Effekts wider. Die beiden Werte werden in der Use Case Analyse (Grunsky 2024) wie folgt beschrieben:

- Eine statistisch aufbereitete Messung der „time-to-target“, also der Zeit vom Anruf bis zum Eintreffen am Einsatzort bzw. der „no-shows“, das sind Notrufe, die nicht bedient werden konnten. Natürlich hängen diese Werte von viel mehr Faktoren ab, als den verfügbaren Be-

- reitschaftsfahrer:innen. Die Vermeidung von Totalausfällen (no-shows) aufgrund mangelnden Bereitschaftspersonals wird aber eine wesentliche Ziffer für den Erfolg des Modells darstellen.
- Die mittleren Personalkosten pro Notruf sollten mit einer akkuraten Planung der Bereitschaftsfahrenden eigentlich abnehmen. Wenn man den o.a. Wert als K.O.-Kriterium für die Modellbewertung erachtet, so sagt dieser Wert (im Vergleich mit Vorangegangenen) aus, wie präzise das Modell arbeitet.

In der Use Case Analyse wurde davon ausgegangen, dass ein Vorhersagemodell sowohl niedrigen als auch erhöhten Bedarf korrekt vorhersagen würde. Die Modellentwicklung hat hier bis dato etwas anderes ergeben. Aus aktueller Sicht scheinen die beiden angeführten KRIs für das Monitoring des Modells im gegenständlichen Anwendungsfall trotzdem Gültigkeit zu besitzen und könnten noch zusätzlich in die Applikation mit aufgenommen werden. In diesem Fall müsste nur noch die Frage geklärt werden, wer für die Dokumentation der erforderlichen Metriken verantwortlich ist.

3.5 Benefits

Sowohl im Rahmen der Use Case Analyse (Grunsky 2024) als auch der Projektplanung (Grunsky 2025) wurden die Stakeholder des Projektes beleuchtet.

Den **Bereitschaftsfahrenden** kommt der eine erfolgreiche Umsetzung des Vorhersagemodells zugute. Durch eine ausreichende Abdeckung werden kurzfristige Anrufe, ob man nicht evtl. doch Dienst versehen kann vermieden und die persönliche Freizeit dadurch planbarer.

Für den **Rettungsdienst** bedeutet eine erfolgreiche Umsetzung möglicherweise eine Kostenersparnis, wenn geringerer Bedarf korrekt vorhergesagt würde, aber in jedem Fall die Vermeidung eines Reputationsschadens aufgrund mangelnden Bereitschaftspersonals.

Die **Planungsstelle** sieht sich im täglichen Betrieb sogar mit zusätzlichem Aufwand und größerer Verantwortung konfrontiert. Dank des Vorhersagemodells wird nun nicht mehr, wie bisher, ein vorgegebener Tages-Fixwert angenommen sondern der von einem System ermittelte wahrscheinliche Bedarf an Bereitschaftspersonal. Die Mitarbeiter:innen der Planungsstelle können diesen Wert nun übernehmen oder dokumentiert und begründet Abändern. In jedem Fall übernehmen Sie, anders als bisher, die Verantwortung für die Höhe des eingeteilten Bereitschaftspersonals. Das Präsidium hat mit dem Auftrag für dieses Modell bereits angeordnet, dass eine zu geringe Besetzung des Bereitschaftsdienstes unbedingt zu vermeiden ist. Diese neue Verantwortung könnte dazu führen, dass die Mitarbeitenden der Planungsstelle den Einsatz des Systems ablehnen, da sie dadurch nur verlieren können. Dieses Risiko wurde in der Projektplanung auch als eines der Hauptrisiken identifiziert.

Eine klar geregelte Vorgehensweise, um den Mitarbeiter:innen der Planungsstelle auch in ungewissen Fällen Sicherheit zu geben, ist zwingend notwendig. Auch die Einbeziehung der Nutzer in das Projekt, gleich von Beginn an, kann dabei helfen, die Identifikation dieser mit dem „selbst mitgestalteten“ System zu erhöhen und ihnen die Angst vor der Benutzung oder deren Konsequenzen zu nehmen (oder diese zumindest frühzeitig zu erkennen).

Abb. 12: Risiko Nutzerakzeptanz

Risikofaktor	Beschreibung
Risikonommer	2
Risikobeschreibung	Ablehnung des Systems durch die Nutzer
Risikoanalyse	Höhere Verantwortung für Ergebnisse eines „Black-Box-Systems“
Eintrittswahrscheinlichkeit	Hoch
Auswirkung	Kritisch; Das System wird nicht verwendet und es werden weiterhin weit überschießende Fixwerte eingetragen; Die Mitarbeiter:innen der Planungsstelle verändern sich oder machen schlechte Stimmung
Risikoniveau	Kritisch
Risikosteuerung	Verminderung
Steuerungsmaßnahmen	Erhöhung der Identifikation mit dem System bei Mitarbeitenden der Planungsstelle (Workshop); klare Regelung der Verantwortung (nicht ausschließlich bei Planungsstelle); Angeordneter Wert (wie bisher) bei unplausiblen Werten, Zweifel oder Ausfall des Systems (zB 90 Bereitschaftsfahrende, Dokumentation erforderlich)
Steuerungsverantwortung	Projektleitung
Risikoüberwachung	„Führung halten“ mit Planungsstelle, Wöchentliche Dokumentation von Bedenken und Stimmung der Mitarbeitenden (Planungsstelle) während der Entwicklungsphase
Überwachungsverantwortung	Projektleitung

Quelle: Grunsky (2025)

3.6 Learnings

Durch die Umsetzung dieses Projektes von der Use Case Analyse über die Projektplanung bis hin zu einer ersten Entwicklung konnte ein tieferes Verständnis für das Zusammenspiel bestehender Werkzeuge, die Bedeutung von Schnittstellen und Struktur in der Arbeit mit diesen Werkzeugen aber auch die Unzulänglichkeiten mit der Prozesse und Frameworks teils spezifiziert sind, erlangt werden.

Speziell auf die Entwicklung der Modelle bezogen, erweiterte die Arbeit mit Machine Learning Frameworks in R wie *caret* (Kuhn 2019) und *fpp3* (Hyndman / Athanasopoulos 2021), aber auch mit Werkzeugen wie Quarto (Posit 2025) das eigene Repertoire der Entwicklung und half dabei eine eigene Vorlage und Arbeitsweise für zukünftige Projekte anzulegen.

Nächste Schritte in diesem Projekt sollten

- eine Diskussion mit dem Präsidium über die Definition einer fallspezifischen Kostenfunktion,
- das Finetuning von Hyperparametern, sowie
- die Umsetzung eines modell- und geschäftszentrierten Monitorings der Vorhersagemodelle

enthalten.

4 Zusammenfassung

Die Herausforderung der gegenständlichen Aufgabenstellung in der Entwicklung eines Modells zur Lösung eines Regressionsproblems für Zeitreihendaten, die

1. einer sehr hohen Volatilität unterliegen und beide Extreme bestmöglich vorhergesagt werden sollten und
2. an der optimalen Grenze zwischen Trainings- und Testdaten einem Datendrift unterliegen

Zur Begegnung dieser Herausforderung wurden verschiedene Modellansätze gewählt wobei schlussendlich der *Facebook Prophet* Algorithmus am erfolgversprechendsten erschien. In der Umsetzung wurden durch den Algorithmus die Saisonalitäten der eingehenden Notrufe gelernt und diese anschließend durch die lineare Korrelation mit der Zielvariablen zur Vorhersage derselbigen herangezogen. Zur Erreichung der linearen Korrelation wurden die eingehenden Anrufe erst ab einer gewissen Anzahl gewertet und durch die Höhe des diensthabenden Personals modifiziert. Eine Generalisierung des Modells konnte über die Verwendung des 90% Konfidenzintervalls erzielt werden, das eine tendenziell zu hohe Vorhersage unterstützt und der Anforderung, nie zu wenig Bereitschaftspersonal vorzusehen, entgegen kommt.

Prophet bietet weiters die Möglichkeit, Trendwenden und besondere vergangene und zukünftige Ereignisse in eine Prognose mit einzukalkulieren und schafft somit die Voraussetzung in der Weiterentwicklung der Modelle diese Fähigkeiten für die Einbindung weiterer denkbarer Prädiktoren zu nutzen. Solche wurden in Punkt 3.2 Data Processing und der vorangegangenen Use Case Analyse diskutiert.

Für die Implementierung der Umsetzung wurde die erhöhte Verantwortung des Personals der Planungsstelle und damit einhergehende eine möglicherweise fehlende Nutzerakzeptanz als ein wichtiges und zu behandelndes Risiko dargelegt.

Der Finalisierung des Projektes sollte die Gütebewertung der Modelle auf Basis einer, mit dem Präsidium gemeinsam definierten, Kostenfunktion erneut durchgeführt werden und danach über ein Hyperparametertraining das beste Setting für das anzuwendende Modell gefunden werden. Außerdem wäre es ratsam auch ein automatisiertes modell- und geschäftszentriertes Monitoring umzusetzen.

5 Literaturverzeichnis

Dorard, L. (2022): *Machine Learning Canvas*.

Grunsky, G. (2024): *Rettungsdienst Berliner Rotes Kreuz MachineLearning-Einsatz in der Bereitschaftsplanung - Use Case Analyse*.

Grunsky, G. (2025): *Rettungsdienst Berliner Rotes Kreuz MachineLearning-Einsatz in der Bereitschaftsplanung - Projektplan*.

Hyndman, R. J./ Athanasopoulos, G. (2021): *Forecasting: Principles and Practice*. Melbourne, Australia, 3. auflage edition.

Kuhn, M. (2019): *The Caret Package*.

Microsoft Azure (2017): *TDSP Lifecycle*.

<https://github.com/Azure/Microsoft-TDSP/blob/master/Docs/lifecycle-detail.md>.

Microsoft Azure (2023a): *Azure-TDSP-ProjectTemplate*. Microsoft Azure.

Microsoft Azure (2023b): *Azure-TDSP-ProjectTemplate/Docs/Project/Exit Report.md at master · Azure/Azure-TDSP-ProjectTemplate*. <https://github.com/Azure/Azure-TDSP-ProjectTemplate/blob/master/Docs/Project/Exit%20Report.md>.

microTool GmbH (2017): *The Use Case Handbook*.

Pak, M. (2024): *Aufgabenstellung_DLMDWME01*.

Posit (2025): *Quarto*. <https://quarto.org/>.

Taylor, S. J./ Letham, B. (2017): *Forecasting at scale*.

6 Abbildungsverzeichnis

1	Struktur des Ordneraufbaus	3
2	Darstellung des Team Data Science Prozesses	4
3	Zeitlicher Verlauf der Zielvariablen	7
4	Korrelation der Zielvariable zu den Notrufen	8
5	Verteilung der Notrufe nach Jahr	9
6	Vorhersage des Basline-Models	10
7	Korrelation modifizierter Notrufdaten mit der Zielvariablen	11
8	Vorhersage Advanced Model V2	12
9	Vorhersage Prophet Model	13
10	mögliches Dateneingabe-GUI	16
11	mögliches Datenausgabe-GUI	17
12	Risiko Nutzerakzeptanz	20

7 Tabellenverzeichnis

1	Struktur der vorhandenen Daten	6
2	Vergleich der ML-Modelle	14

A Annexes

A.1 Datenbeschreibung

Dateiname: **sickness_table.csv**

Dateigröße (Kb): **48.09**

Originaldaten gespeichert in: **01_data/01_lake/01_use_case_2/**

Aufgrund der geringen Größe wurde der gesamte Datensatz ebenso in ***** 00_sample_data/01_raw*** hinterlegt.

Der Datensatz enthält **1152** Datenpunkte mit jeweils **8** Merkmalen und wurde als Datentyp **spec_tbl_df, tbl_df, tbl, data.frame** angelegt.

Die Daten weisen folgende Struktur auf:

```
cols(  
  ...1 = col_double(),  
  date = col_date(format = "%Y-%m-%d"),  
  n_sick = col_double(),  
  calls = col_double(),  
  n_duty = col_double(),  
  n_sby = col_double(),  
  sby_need = col_double(),  
  dafted = col_double()  
)
```

Die u.a. Beschreibung der Daten wurde dem Dokument "Zusatzinformationen_DLMDWME01 (**pak_zusatzinformationen_dlmdwme01_2024?**)" entnommen. Hiernach geben die enthaltenen Daten Auskunft über Metriken (auf Tagesbasis) des Einsatzfahrten-Bereitschaftspersonals des Berliner RotKreuz Rettungsdienstes. Diese tageweisen Zeitreihendaten umfassen den Zeitraum von **2016-04-01** bis **2019-05-27**.

Beschreibung der einzelnen Datenmerkmale

Variable	Beschreibung
...1	Zähler
date	Datum
n_sick	Anzahl der Einsatzfahrer, die einen Krankenstand angemeldet haben
calls	Anzahl der Notrufe
n_duty	Anzahl der Einsatzfahrer im Dienst
n_sby	Anzahl der verfügbaren Ersatzfahrer
sby_need	Anzahl der Ersatzfahrer, die aktiviert werden
dafted	Anzahl der zusätzlichen Einsatzfahrer, die aktiviert werden müssen, wenn die Anzahl der Einsatzfahrer im Bereitschaftsdienst nicht ausreicht

A.2 explorative Datenanalyse

Einleitung

Dieses Dokument beschreibt die vorbereitenden Abschnitte **Prüfung der Datenqualität** und **Explorative Datenanalyse** für den bestehenden Use Case. Ziel der explorativen 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> <dtm>          <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> <dtm>          <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 nicht 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. Dennoch 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).

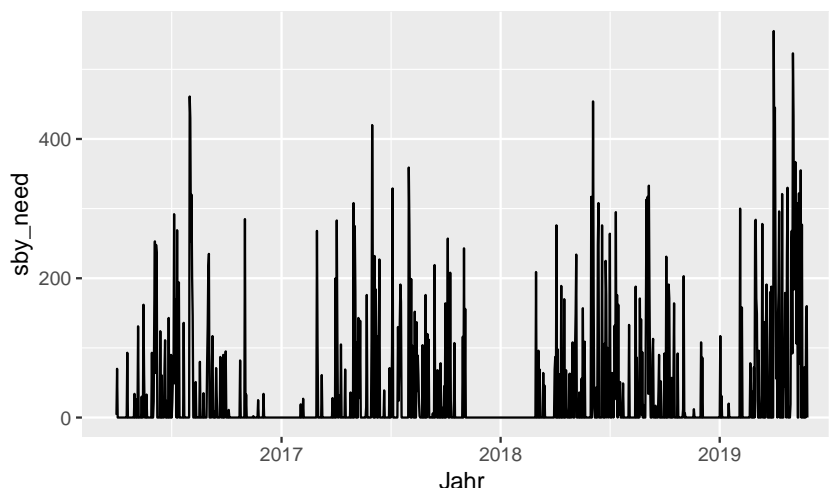
Visualisierung der Zielvariable

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 Forecasting 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")
```

Zeitplot der Zielvariablen

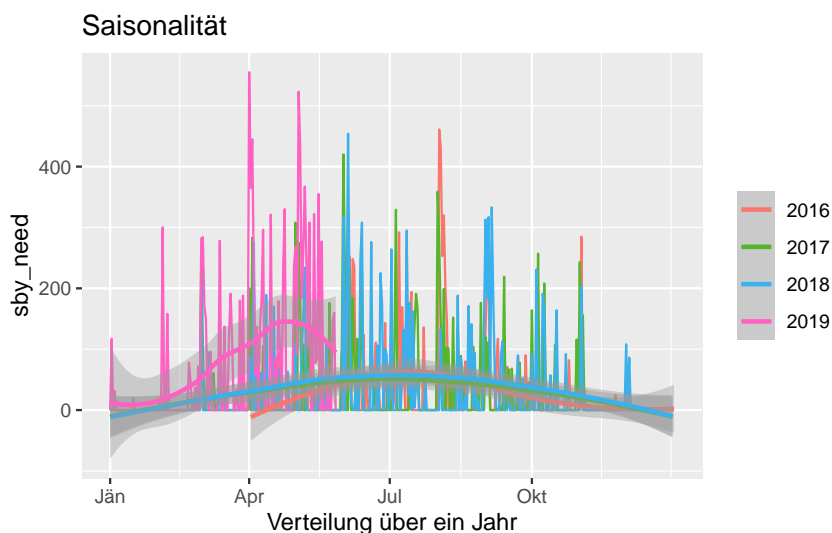


Die vermutete Saisonalität ist in der zeitlichen Darstellung des Bedarfs gut erkennbar. Augenscheinlich scheint es monatliche Spitzen in der Zielvariablen zu geben und um den Jahreswechsel ist oft nur

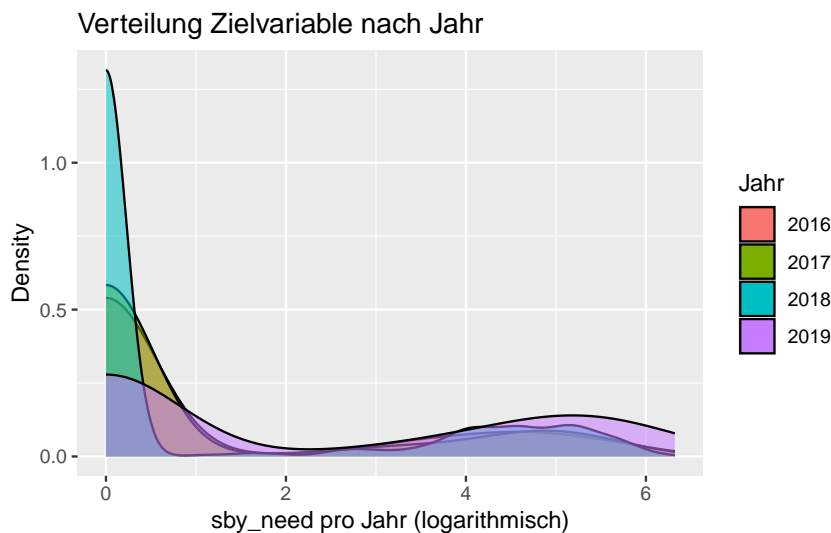
geringer Bedarf vorhanden. Mit dem Jahreswechsel auf 2019 ändert sich das Muster der Bedarf weist ab dem Jahreswechsel deutlich höhere Spitzen auf und auch um in der "sonst ruhigen" Zeit zwischen 2018 und 2019 sind *Peaks* erkennbar. Die rapiden Wechsel zwischen "kein Bedarf" und "sehr hoher Bedarf" 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")
```

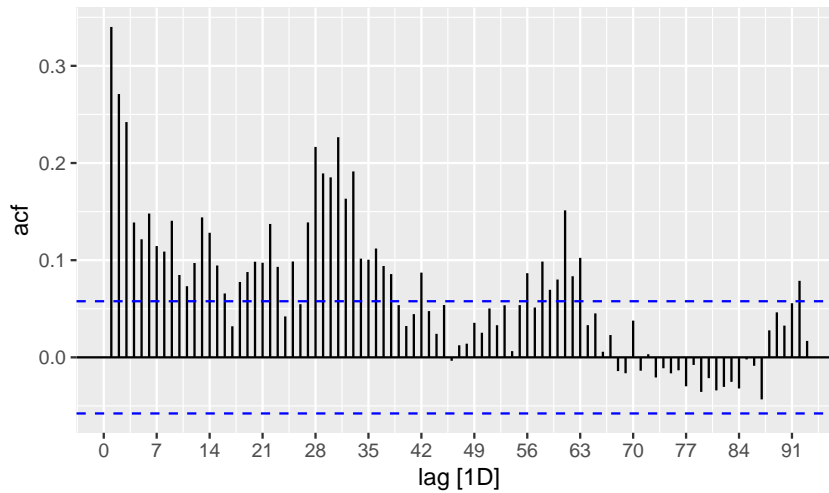
Wie bereits zuvor erläutert, sieht man auch in der Dichtefunktion des logarithmisch dargestellten Wertes von *sby_need* den Wechsel zwischen geringen und sehr hohem Bedarf. Eine Normalverteilung liegt logischerweise nicht vor, da der Modus zwar bei 0 zu liegen scheint, ein negativer Wert aber freilicherweise 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 Peaks. 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")
```

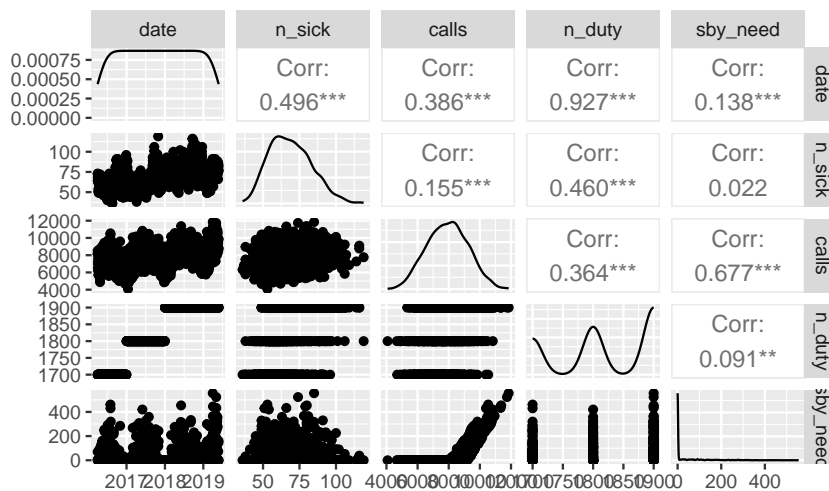
Autokorrelation von sby_need



Übersicht über die weiteren Datenmerkmale

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

Korrelationsplot der Merkmale



sby_need (die Zielvariable) korreliert augenscheinlich am meisten mit der Anzahl der Notfalleinsätze (*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 Werteverteilung 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 Linaritäten" 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 Saisonalität 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.

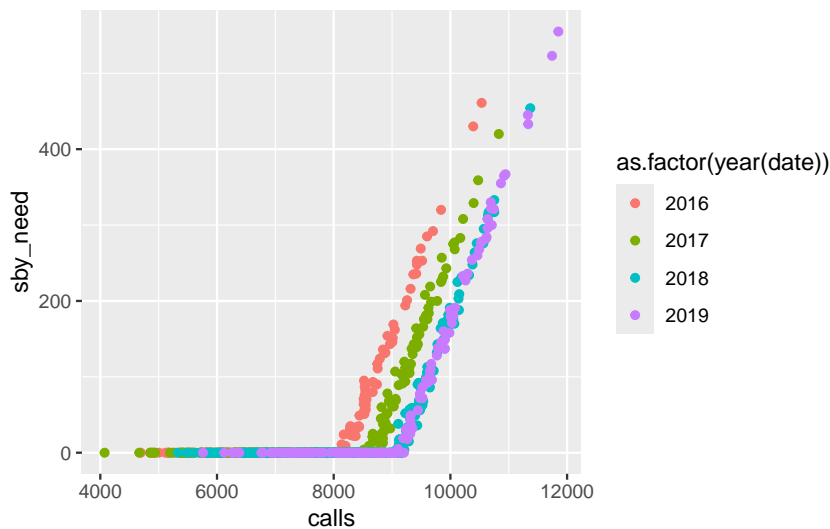
Merkmalsgenerierung

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.

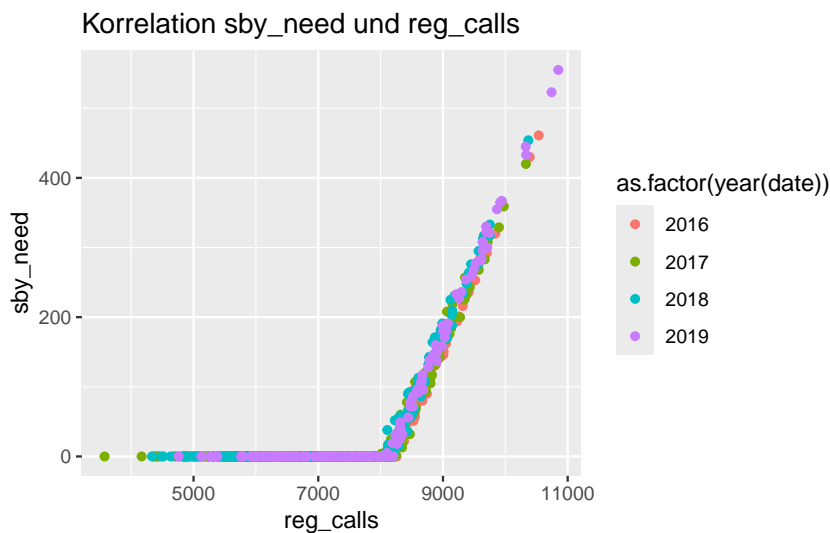
$$reg_calls_t = calls_t - n_duty_t \mid n_duty_t = (n_duty_{year} - 1700) * 5$$

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"))

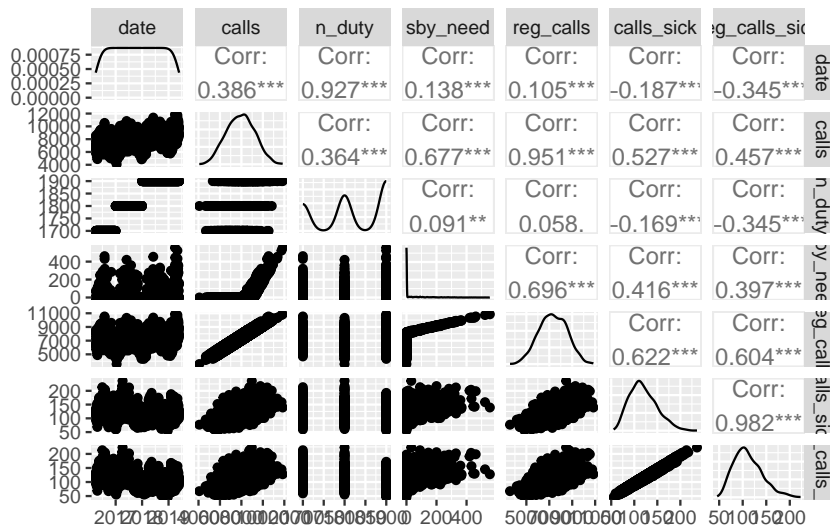
ggplot(ts_sby,
       aes(x = reg_calls,
           y = sby_need,
           colour = as.factor(year(date))
       )) +
  geom_point() +
  labs(title = "Korrelation sby_need und reg_calls")
```



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 Entropie (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(feats_spectral)

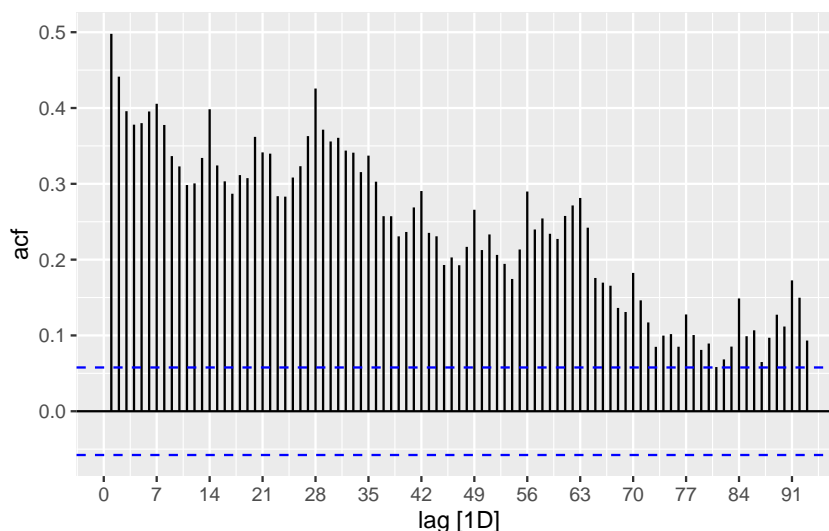
# A tibble: 1 x 5
  calls_spectral_entropy sby_need_spectral_entropy reg_calls_spectral_entropy
          <dbl>                <dbl>                <dbl>
1          0.812                0.945                0.880
# i 2 more variables: calls_sick_spectral_entropy <dbl>,
#   reg_calls_sick_spectral_entropy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

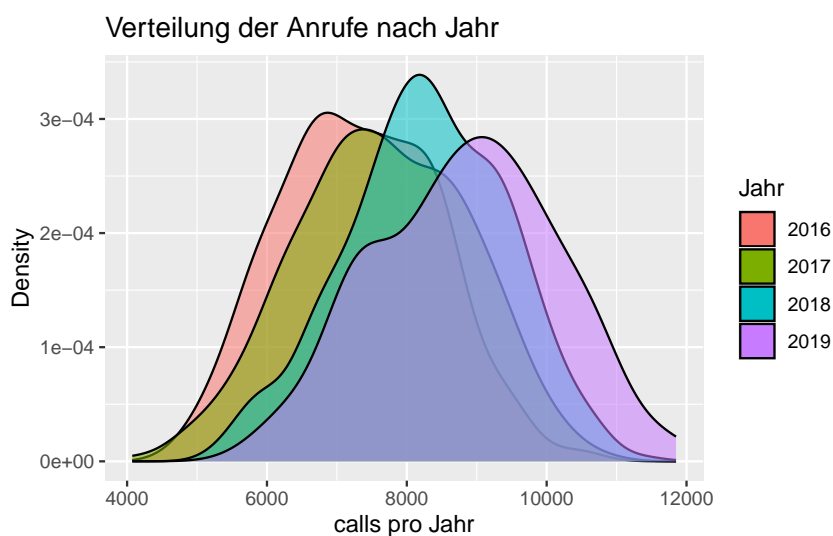
```
ts_sby %>%
  ACF(y = calls, lag_max = 93) %>%
  autoplot()
```



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 augenscheinlich 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.

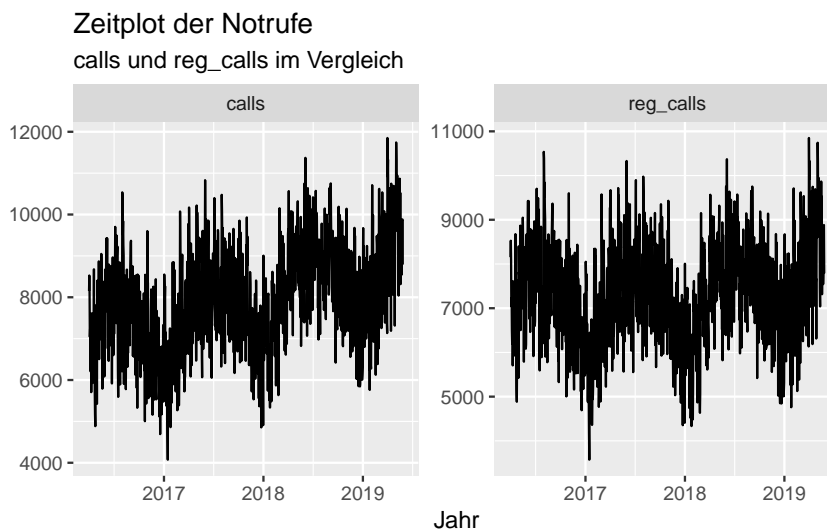
Die Verteilung des möglichen Prädiktors gibt Ausschluss, bzw. einen Überblick über die zu erwartenden Werte.

```
```{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 Forecasting. 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.

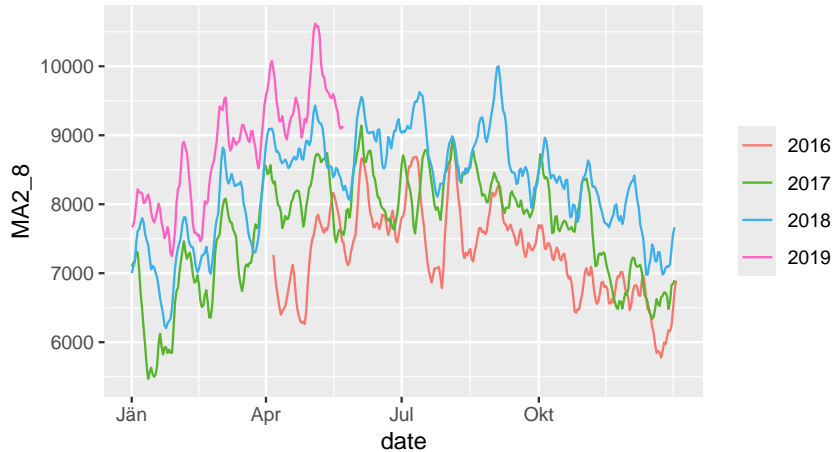
Auch wenn in den oberen beiden Plots eine jährliche und monatliche Saisonalität gut erkennbar ist, zeigt der Verlauf dennoch eine hohe Volatilität (vmtl. durch die leichte wochenweise Autokorrelation, wie weiter oben beschrieben). Ein 2x8 Moving Average würde die wöchentliche Saisonalität glätten, ohne 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")
```


Saisonalitätsplot der Notrufe

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.

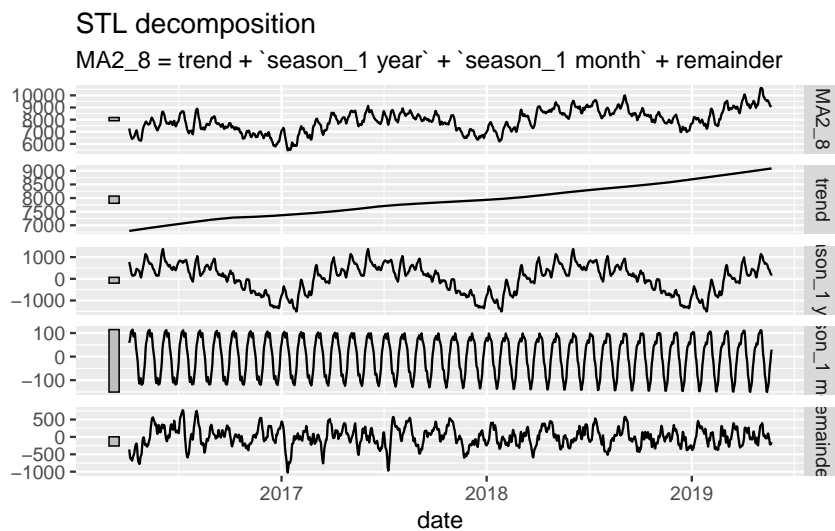
```
ts_sby %>%
  features(MA2_8, feat_spectral)
```

```
# 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 vielversprechendsten. 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 Baselinemodell 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/>.

A.3 Baseline Model

Einleitung

Dieses Dokument beschreibt die Erarbeitung eines Baseline- oder Benchmarkmodells. Dieses Modell dient der Anwendung einer simplen Lernmethode auf die gegebene Aufgabenstellung. Die Ergebnisse dieses Modells werden in weiterer Folge zur Qualitätsmessung und Evaluierung von ausgefeilteren Modellen verwendet.

([hyndman_forecasting_2021?](#)) schlägt für die Erstellung von Benchmarks im Zeitreihenbereich die Algorithmen

- **Mean:** Der Mittelwert der Zeitreihe
- **Naïve:** Der Wert der letzten Observation
- **Seasonal Naïve:** Der Wert der letzten gleichen Saison zB das gleiche Monat des Vorjahres, oder der gleiche Tag des Vormonats (abhängig von der Saisonalität)
- **Drift:** Eine Variation von Naïve, die die durchschnittliche Veränderungsrate der Daten berücksichtigt und somit auch steigen oder fallen kann

Im R-package *fpp3* ist außerdem auch ein **TSLM** Algorithmus enthalten, der ein Lineares Modell mit Trend- und saisonalen Komponenten auf die Daten anwendet. Dieser Algorithmus könnte sich in diesem Anwendungsfall auch gut für ein Baselinemodell eignen.

Nachdem die Modelle wenig Rechenkapazität benötigen, bietet es sich an, einfach alle Modelle zu trainieren und anschließend die, für diese Aufgabe gültige Benchmark zu wählen. Außerdem gibt die aktuelle Vorgehensweise, nämlich ein konstantes Vorhalten von 90 Bereitschaftsfahrenden ebenso bereits eine zu berücksichtigende Messvariable vor, schließlich sollen ja langfristig Bereitschaftskosten durch die Anwendung eines Modells gespart werden können.

Ergebnisse der explorativen Datenanalyse

Die Erkenntnisse der explorativen Datenanalyse liefern eine Grundlage für die Erstellung eines Baseline Modells und lassen sich in Bezug auf die Zielvariable wie folgt zusammenfassen:

1. Die vermutete Saisonalität ist augenscheinlich gegeben, der rasche Wechsel zwischen sehr hohem und fast keinem Bedarf an Bereitschaftspersonal könnten eine Vorhersage erschweren. Ab 2019 ändert sich das saisonale Verhalten der Daten und auch ein durchschnittlicher Anstieg in den Werten der Zielvariablen ist zu beobachten.
2. Eine lineare Annäherung in der Ansicht, die sogenannte Smooth-Curve, zeigt, dass die Spitzenwerte auch mit einem 99% Konfidenzintervall nicht abgedeckt werden.
3. Ein Mindestmaß an etwa 35 Personen in Bereitschaft sinnvoll ist.
4. Ebenso wurde eine jährliche und (instabile) monatliche Saisonalität mit schwacher ausgeprägten wöchentlichen Perioden erkannt. Die Autokorrelation zeigt, dass eine Vorhersage über einen längeren Zeitraum nicht zielführend ist. Das wird vmtl. über die Änderungen im Muster ab Beginn 2019 erklärt werden können.
5. Es wurde außerdem festgehalten, dass eine nahezu lineare Abhängigkeit des Bedarfs an StandBy-Personal zu der Anzahl eingehender Notrufe besteht, diese jedoch auch vom diensthabenden Personal beeinflusst wird.

Am erfolversprechendsten wurde eine indirekte Vorhersage aufgrund der Saisonalität des Merkmals *calls* und der, ab einem gewissen Wert, nahezu linear anzunehmenden Korrelation mit *sby_need* beurteilt, wobei Letztere durch *n_duty* beeinflusst ist.

Für ein Baselinemodell scheint der obige Ansatz jedoch zu aufwendig. Deshalb wird dieses direkt auf die Zielvariable trainiert. Um den oben genannten Herausforderungen in der Vorhersage zu begegnen, wird jedoch noch eine weitere Vorverarbeitung der Daten nötig sein.

Daten laden

```
load(file = "../00_sample_data/02_processed/data_explorative.rda")
```

Aufsplitten der Daten in Trainings- und Testdaten

Gemäß Aufgabe soll der Vorhersagezeitraum etwa 2 Monate betragen. Der Zeitraum, der durch das Datenmaterial abgedeckt wird, umfasst etwa drei Jahre (36 Monate). Übliche Aufteilungen von Datensätzen in Trainings- und Testdaten, sehen ein Verhältnis von 70:30 oder 80:20 Prozent vor (je nach Größe des Datensatzes), der Umfang des gewünschten Vorhersagezeitraums sollte aber jedenfalls gegeben sein. Ein Verhältnis von 80:20 entspricht bei den vorliegenden Daten etwa 29 Monate:7 Monate. Der Vorhersagezeitraum wäre damit abgedeckt. Die, in der explorativen Datenanalyse festgestellte Änderung des saisonalen Verhaltens mit dem Jahreswechsel auf 2019 sowie ein deutlich schlechterer Autokorrelationswert ab lag 63 müssen in den Trainingsdaten unbedingt berücksichtigt werden. Daher wird der Testdatensatz auf 2 Monate reduziert.

Die Anzahl der Variablen wird gleichzeitig ebenfalls auf ein benötigtes Mindestmaß beschränkt.

```
ts_sby_train <- ts_sby %>%
  select(date, sby_need) %>%
  slice(0:(n() - 62))

ts_sby_test <- ts_sby %>%
  select(date, sby_need) %>%
  slice((n() - 61):n())
```

Modellerstellung

Erster Versuch

Der erste Versuch der Modellerstellung wendet die beschriebenen Benchmarkmodelle, trotz der erkannten Herausforderungen, direkt auf die Zielvariable an, um zu sehen, wie sich die Berechnungen verhalten. Ein brauchbares Ergebnis wird an dieser Stelle nicht erwartet. Um die monatlichen saisonalen Muster einzufangen, aber die Änderungen zu Beginn des Jahres 2019 hervorzuheben, wird der Trainingszeitraum weiter auf ein Jahr beschränkt.

```
# Training
t <- system.time(
  progressr::with_progress(
    sby_basemodel <- ts_sby_train %>%
      filter(date >= "2018-03-26") %>%
      model()
```

```

    mean = MEAN(sby_need),
    naive = NAIVE(sby_need),
    snaive = SNAIVE(sby_need ~ lag(63), drift = TRUE),
    drift = RW(sby_need ~ drift()),
    tslm = TSLM(sby_need ~ trend() + season()),
  )
), gcFirst = TRUE
)

print(paste0(" Die Benötigte Zeit für das Training betrug ", round(t[3],2), " Sekunden."))

```

```
[1] " Die Benötigte Zeit für das Training betrug 0.07 Sekunden."
```

```

# Vorhersage
t <- system.time(
  progressr::with_progress(
    sby_basefc <- sby_basemodel %>%
      forecast(h = "2 months", level = c(80))
  ), gcFirst = TRUE
)

print(paste0(" Die Benötigte Zeit für die Vorhersage war ", round(t[3],2), " Sekunden."))

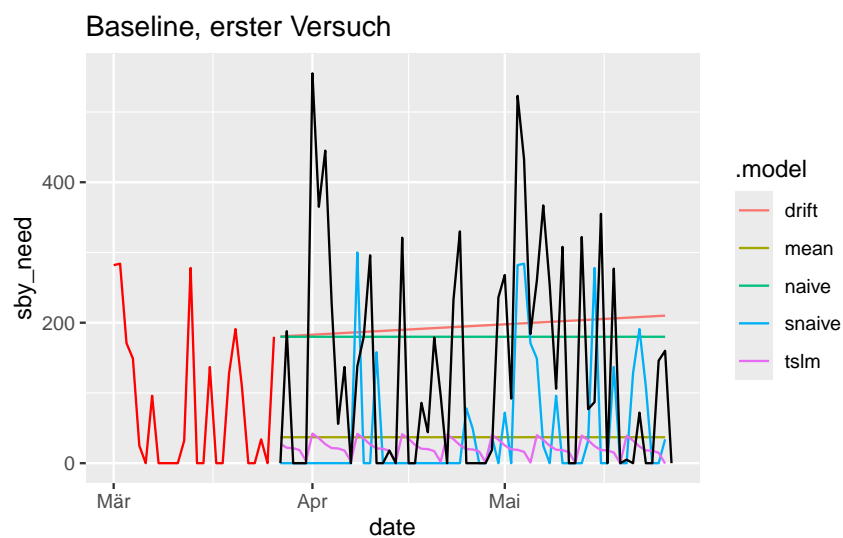
```

```
[1] " Die Benötigte Zeit für die Vorhersage war 0.13 Sekunden."
```

```

sby_basefc %>%
  autoplot(ts_sby_test,
    level = NULL) +
  autolayer(ts_sby_train %>% filter(date >= "2019-03-01"), colour = "red" ) +
  labs(y = "sby_need",
    title = "Baseline, erster Versuch")

```



In dem o.a. Plot sind die Trainingsdaten Rot und die Testdaten Schwarz dargestellt. Die Vorhersagen durch die einzelnen Algorithmen sind farblich in der Legende kodiert. Wenig überraschend ist die Vorhersage nicht brauchbar. Der *Drift* zeigt einen erwartbaren positiven Trend und *Naïve* eben nur den letzten Wert (in diesem Fall 180). Der *Mean* liegt bei ca. 37 und damit etwa dort, wo gem. explorativer Datenanalyse ein Mindestwert angesetzt werden sollte.

TSLM erkennt wochenweise Schwankungen (wie auch im Autokorrelationsplot beobachtet), setzt diese allerdings viel zu gleichmäßig und vor allem zu niedrig an. *Snaïve* kann offenbar den rapiden Wechsel zwischen 0 und auftretenden Spitzen modellieren. Setzt diese Spitzen allerdings zu niedrig an und, viel fataler, nicht dort wo sie tatsächlich auftreten.

Positiv hervorzuheben ist die benötigte Zeit für Training und Vorhersage. Diese betrug hier nur wenige hunderstel Sekunden. Der dazu verwendete Rechner war ein älteres Notebook vom Typ: Dell Latitude 5490, Intel i7 vPro (8th Generation) mit 16 Gb RAM.

Die Berechnung der Modelle auf neueren Systemen sollte die Hardware demnach nicht vor große Herausforderungen stellen.

Zweiter Versuch mit modifizierten Daten

Der Vorhersageplot des ersten Versuches zeigt gut dessen Scheitern. Die hohen Ausschläge zu Beginn jedes Monats werden nicht durch die Modelle abgedeckt, die Peaks sind aber gleichzeitig auch zu unregelmäßig um zB von einem *Snaïve*-Model "getroffen" zu werden. *TSLM* lässt schön die wöchentliche Komponente in den Daten erkennen, die aber eigentlich nur schwach ausgeprägt ist.

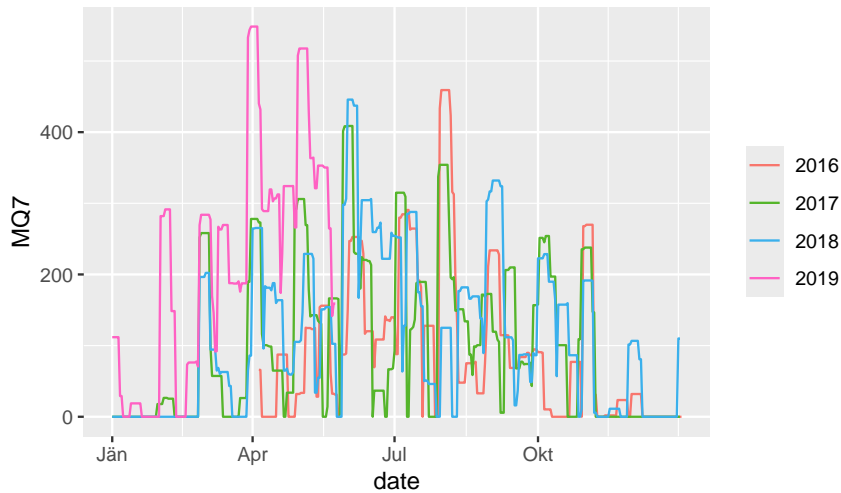
Die Idee für den zweiten Versuch eines Benchmarkmodells ist es, die rapiden Wechsel zwischen Spitzen und 0-Linie zu entfernen, die entsprechende Höhe des möglichen Bedarfes abzudecken und durch Aggregation die "Treffsicherheit" für auftretenden Bedarf zu erhöhen. Auf die intra-wöchentlichen Schwankungen wird keine Rücksicht mehr genommen.

Dazu bietet sich, wie auch in der explorativen Datenanalyse, die Verwendung eines gleitenden Durchschnitts an, um den Wochenbedarf zu glätten. Das Problem der Durchschnittsbildung ist jedoch, dass die Spitzen "per design" nie richtig vorhergesagt werden können. Als gleitendes Maß wird in diesem Fall daher die 99% Quantile mit einem Fenster von sieben Tagen getestet.

```
ts_sby <- ts_sby %>%
  mutate(
    MQ7 = slider::slide_dbl(sby_need,
                           ~ quantile(x = .x,
                                       probs = 0.99),
                           .before = 3, .after = 3,
                           .complete = TRUE))

ts_sby %>%
  gg_season(MQ7, period = "1y") +
  labs(title = "Gleitende Quantille")
```

Gleitende Quantile



Im saisonalen Plot sieht man, dass die Spitzenwerte, durch die Glättung, deutlich besser übereinander liegen und auch die monatliche Komponente der Saisonalität ist gut erkennbar. Der u.a. Wert der spektralen Entropy hat sich gegenüber jenem der Zielvariablen, in der explorativen Datenanalyse merkbar verbessert.

```
ts_sby %>%
  features(MQ7, feat_spectral)
```

```
# A tibble: 1 x 1
  spectral_entropy
  <dbl>
1         0.695
```

Die Treffsicherheit des Baselinemodells soll weiters durch eine wochenweise Aggregation erhöht werden. Das bedeutet in der Praxis, dass jeweils für eine ganze Woche die selbe Anzahl an Bereitschaftspersonal vorzuhalten wäre, da nicht genau bestimmt werden kann, wann genau der Spitzenbedarf auftreten wird. Nach der Aggregation werden ein neuer Trainings und Testdatensatz gebildet und die Modelle trainiert.

```
```{r}
#| message: false
#| warning: false

Aggregation der täglichen Daten
ts_sby_week <- ts_sby %>%
 filter(!is.na(MQ7)) %>%
 index_by(week = ~ yearweek(.)) %>%
 summarise(w_need = max(MQ7)) %>%
 filter(!is.na(w_need))

Datenaufteilung (61 Tage werden zu 9 Wochen)
ts_sby_train <- ts_sby_week %>%
 select(week, w_need) %>%
```

```

slice(0:(n() - 9))

ts_sby_test <- ts_sby_week %>%
 select(week, w_need) %>%
 slice((n() - 9):n())
...

```

```

Training
t <- system.time(
 progressr::with_progress(
 sby_basemodel <- ts_sby_train %>%
 filter(year(week) >= 2017) %>%
 model(
 mean = MEAN(w_need),
 naive = NAIVE(w_need),
 snaive = SNAIVE(w_need ~ lag(52), drift = TRUE),
 drift = RW(w_need ~ drift()),
 tslm = TSLM(w_need ~ trend() + season()),
)
), gcFirst = TRUE
)

print(paste0(" Die Benötigte Zeit für das Training betrug ", round(t[3],2), " Sekunden."))

```

```
[1] " Die Benötigte Zeit für das Training betrug 0.06 Sekunden."
```

```

Vorhersage
t <- system.time(
 progressr::with_progress(
 sby_basefc <- sby_basemodel %>%
 forecast(h = "12 months", level = c(99))
), gcFirst = TRUE
)

print(paste0(" Die Benötigte Zeit für die Vorhersage war ", round(t[3],2), " Sekunden."))

```

```
[1] " Die Benötigte Zeit für die Vorhersage war 0.22 Sekunden."
```

```

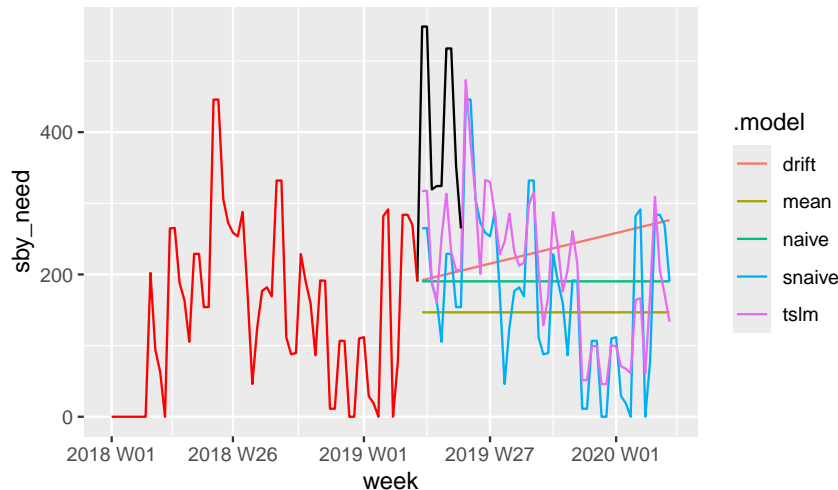
Visualisierung der Vorhersage

sby_basefc %>%
 autoplot(ts_sby_test,
 level = NULL) +
 autolayer(ts_sby_train %>% filter(year(week) >= 2018), colour = "red") +
 labs(y = "sby_need",
 title = "Baseline, zweiter Versuch")

```



### Baseline, zweiter Versuch



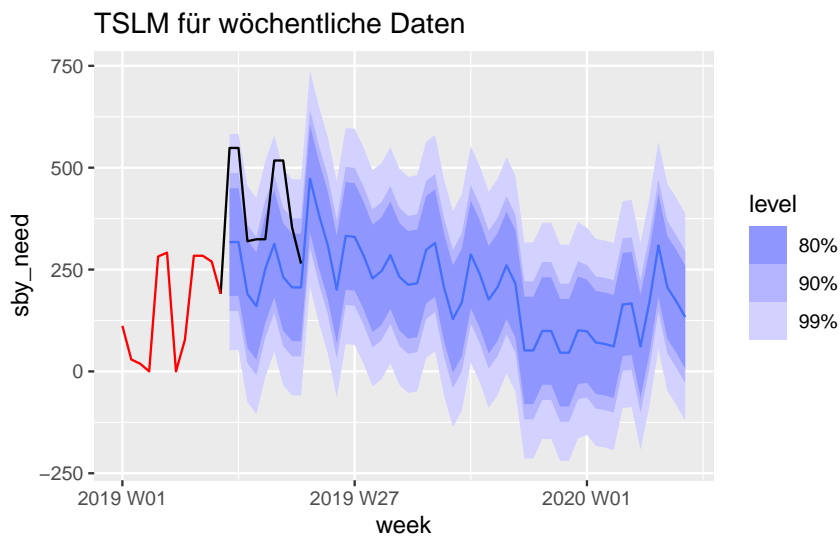
Nachdem im Saisonalitätsplot ein Aufwärtstrend klar erkennbar ist, wurde der Trainingszeitraum auf zwei etwa zwei Jahre und der Vorhersagezeitraum auf ein Jahr erweitert. Die Berechnungszeit für das Training und die Vorhersage kann auch bei diesem Ansatz vernachlässigt werden.

Der o.a. Plot folgt dem Farbschema des vorigen Vorhersageplots. Die unterschiedlichen Qualitäten der Vorhersagen sind gut erkennbar. *Drift* und *Naïve* können auch hier wieder verworfen werden. *Mean* gibt einen sehr hohen Durchschnitt aus. *Snaïve* bildet zwar die Saisonalität ab, lässt aber den Trend unbeachtet.

*TSLM* scheint, im aggregierten Ansatz, für ein Baselinemodell zu funktionieren. Die Vorhersage über den Zeitraum von einem Jahr zeigt, dass das Modell sowohl den Trend als auch die Saisonalität aufnimmt. Der erhöhte Bedarf zu Beginn jedes Monats ist deutlich herausgearbeitet. Die, sich mit den Trainingsdaten überschneidenden, Vorhersagewerte sind zu niedrig, was durch den plötzlich stärkeren Aufwärtstrend erklärbar ist, das Modell scheint sich aber im weiteren Verlauf einer möglichen Realität anzunähern. Was bei dieser Variante nicht wiederzufinden ist, sind die Ruhephasen. Im Gegensatz zu aktuellen Vorgehensweise wird bei diesem Modell, außer in den Wintermonaten, sonst nie mit 90 Personen im Bereitschaftsdienst das Auslangen gefunden werden. Das bedeutet, für das Deutsche Rote Kreuz Berlin wird der Bereitschaftsdienst damit in Summe vermutlich teurer, jedoch besser planbar und es müssen weniger Bereitschaftsfahrer zusätzlich aktiviert werden.

*TSLM* wird als Baselinemodell für diesen UseCase akzeptiert und u.a. noch einmal im Fokus mit 80%, 90% und 99% Konfidenzintervall dargestellt. Das 99% Intervall hätte, im Vergleich mit den Testdaten, auch die hohen Bedarfsspitzen ausreichend prognostiziert, allerdings würden mit diesem Konfidenzintervall in Monaten mit weniger Bedarf trotzdem über 250 Bereitschaftsfahrer:innen vorgesehen werden. Zumindest beim Baseline-Modell werden hier Abstriche in den Anforderungen gemacht werden müssen.

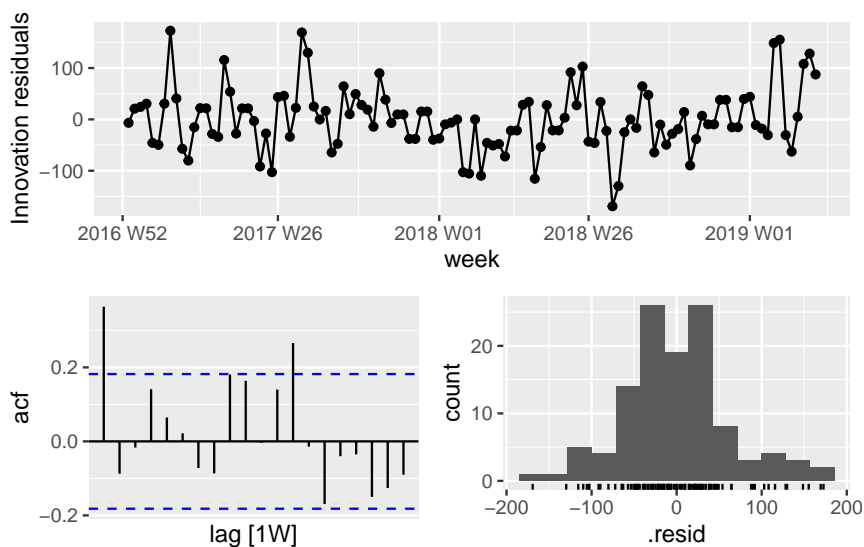
```
sby_basefc %>%
 filter(.model == "tslm") %>%
 autoplot(ts_sby_test,
 level = c(80, 90, 99)) +
 autolayer(ts_sby_train %>% filter(year(week) >= 2019), colour = "red") +
 labs(y = "sby_need",
 title = "TSLM für wöchentliche Daten")
```



## Evaluierung

Der u. a. angeführte Residualsplot zeigt ein nahezu optimales Verhalten des Algorithmus. In den Residuals ist auf den ersten Blick kein deutliches "Rest-Pattern" in den Daten zu beobachten und genauso wenig im Autokorrelationswert. Die Residuals wären im besten Fall mit einem arithmetischen Mittel von 0 normal verteilt. Das trifft nicht ganz zu, die Annäherung an das gewünschte Verhalten ist aber gut erkennbar. Dieser Plot zeigt allerdings nur, wie gut die Realität mit dem erstellten Modell (den sogenannten *fitted values*) erklärt werden kann. Es ist keine Bewertung der Vorhersagefähigkeit.

```
mod <- select(sby_basemodel, tslm)
gg_tsresiduals(mod)
```

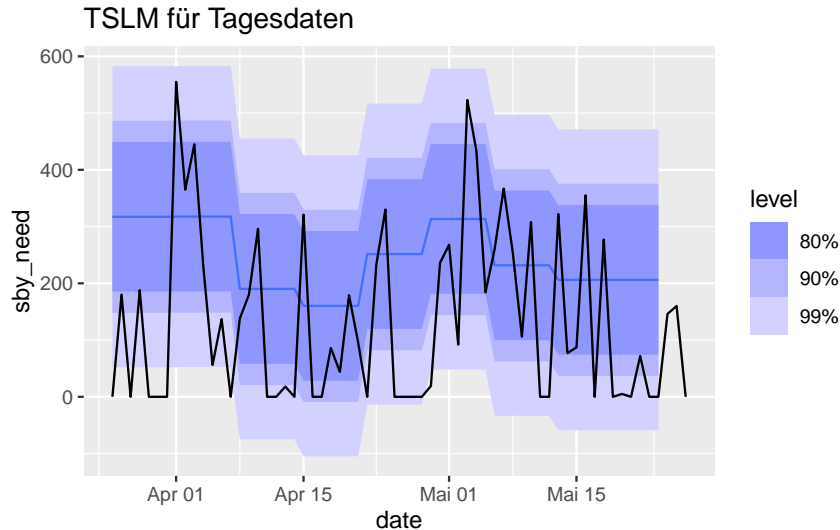


Für diesen Fall werden nachfolgend einige Bewertungsmethoden durchgeführt, die die Güte des *TSLM* belegen sollen. Die Anforderungen des Use Case sehen eine tageweise Vorhersage vor, das Modell berechnet allerdings wöchentliche Daten. Auch für die Vergleichbarkeit mit zukünftigen Modellen ist es sinnvoll die aggregierten Daten wieder auf Tagesdaten aufzuteilen. Beim *TSLM* resultiert das natürlich darin, dass innerhalb einer Woche alle Tage den gleichen Wert haben.

```
week2day <- ts_sby %>%
 filter(!is.na(MQ7)) %>%
 mutate(week = yearweek(date)) %>%
 select(week, date)

fc <- sby_basefc %>%
 as_tibble() %>%
 left_join(week2day, by = "week") %>%
 filter(!is.na(date)) %>%
 mutate(sby_need = w_need) %>%
 select(.model, date, sby_need, .mean) %>%
 as_fable(index = date, key = .model, response = "sby_need", distribution = sby_need)

fc %>%
 filter(.model == "tslm") %>%
 autoplot(filter(ts_sby, date >= "2019-03-25"),
 level = c(80, 90, 99)) +
 labs(y = "sby_need",
 title = "TSLM für Tagesdaten")
```



Wie die Grafik zeigt wird durch das Baseline Model zumeist ein zu hoher Wert vorhergesagt, die Spitzenwerte der Testdaten werden durch die Forecast aber nicht erreicht. Als einfaches Benchmark-Modell ist *TSLM* in diesem Fall jedoch geeignet.

```
fc %>%
 accuracy(ts_sby, list(RMSE = RMSE, MAE = MAE)) %>%
 select(-.type)
```

```
A tibble: 5 x 3
 .model RMSE MAE
 <chr> <dbl> <dbl>
1 drift 165. 147.
2 mean 154. 134.
3 naive 162. 144.
4 snaive 161. 142.
5 tslm 186. 164.
```

Die Messwerte RMSE und MAE zeigen unterschiedliche Bewertungen, in beiden Fällen wäre jedoch der einfache Durchschnitt (*mean*) das am wenigsten fehleranfällige Modell. Dass den Anforderungen an den Use Case dabei nicht entsprochen wird, ist aus den Visualisierungen gut erkennbar. Außerdem besteht hier die "Benchmark" in der bisherigen Vorgehensweise und einem Fixwert von 90 Bereitschaftsfahrenden. Im Allgemeinen sind die Fehlerwerte sehr hoch zu bewerten, aber für den zukünftigen Vergleich mit weiter entwickelten Modellen sind sie hilfreich.

```
fc %>%
 accuracy(ts_sby,
 list(quantile_score = quantile_score), probs = 0.90) %>%
 select(-.type)
```

```
A tibble: 5 x 2
 .model quantile_score
 <chr> <dbl>
1 drift 80.2
2 mean 66.4
3 naive 77.3
4 snaive 59.1
5 tslm 55.7
```

Ergänzend zu den irreführenden vorangegangenen Gütekriterien bietet sich für den vorliegenden Use Case zusätzlich die Bewertung der Verteilung zur Gütebestimmung an. Der **Quantile Score** vergleicht die vorhergesagten mit den aufgetretenen Werten unter der Annahme, dass (gem. gesetztem Parameter) 90% der realen Werte unter der entsprechenden Quantile der Verteilung liegen. Die Messung folgt einer *Pinball Loss Function* (Vermorel 2012) wobei, bei dem gesetzten 90% Parameter, Fehler die überhalb dieser Marke liegen gewichteter bewertet werden, als jene die unterhalb des Wertes liegen. Der Durchschnitt der einzelnen Bewertungen gibt den Quantile Score, der als absoluter Fehler zu verstehen ist. Je niedriger der Quantile Score, desto besser.

Weitere Messwerte beschreiben zusätzlich die Fähigkeit des Modells den Anforderungen des Use Case hinsichtlich zu geringer Schätzungen gerecht zu werden. Die bisherige Vorgehensweise einer fixen Einteilung von 90 Fahrer:innen im Bereitschaftsdienst wird hier für einen Vergleich ebenfalls berechnet.

```
Evaluierungsdatensatz - Abweichungen aus Forecast und realen Werten "sby_diff"
eval <- fc %>%
 filter(.model == "tslm") %>%
 as.tibble() %>%
 select(date, .mean) %>%
 mutate(sby = .mean) %>%
 filter(!is.na(sby)) %>%
 as_tsibble(index = date) %>%
```

```
select(date, sby, sby_need) %>%
 filter(!is.na(sby)) %>%
 mutate(sby_diff = sby - sby_need,
 act_diff = 90 - sby_need) # Abweichungen der bisherigen Vorgehensweise
```

Positive Werte von *sby\_diff* und *act\_diff* bedeuten, dass zuviel Bereitschaftspersonal vorhergesagt wurde und negative Werte besagen, dass der Bedarf nicht durch die Vorhersage abgedeckt wurde. Generell sind daher positive Werte gegenüber negativen vorzuziehen.

```
Messgrößen von TSLM
```

```
print(paste0("RMSE: ", sqrt(mean(eval$sby_diff^2))))
```

```
[1] "RMSE: 185.938781697687"
```

```
print(paste0("MAE: ", mean(abs(eval$sby_diff))))
```

```
[1] "MAE: 164.48169709389"
```

```
print(paste0("Durchschnittswert der Abweichungen (MeanError): ", mean(eval$sby_diff)))
```

```
[1] "Durchschnittswert der Abweichungen (MeanError): 108.639431818182"
```

```
print(paste0("In ", round(nrow(eval[eval$sby_diff < 0,]) / nrow(eval) * 100, 2), "% der Fälle w
```

```
[1] "In 26.23% der Fälle wurde zuwenig Personal vorhergesagt (% low)"
```

```
print(paste0("Summe der mangelhaften negativen Vorhersagen (sum low): ", sum(eval[eval$sby_diff
```

```
[1] "Summe der mangelhaften negativen Vorhersagen (sum low): -1703.18909090909"
```

```
print(paste0("Durchschnittswert der mangelhaften negativen Vorhersagen (mean low): ", mean(eval
```

```
[1] "Durchschnittswert der mangelhaften negativen Vorhersagen (mean low): -106.449318181818"
```

```
print(paste0("Größte negative Vorhersage (max low): ", min(eval[eval$sby_diff < 0,]$sby_diff)))
```

```
[1] "Größte negative Vorhersage (max low): -237.440568181818"
```

```
Messgrößen der bisherigen Vorgehensweise
```

```
print(paste0("RMSE: ", sqrt(mean(eval$act_diff^2))))
```

```
[1] "RMSE: 160.051477374737"
```

```
print(paste0("MAE: ", mean(abs(eval$sact_diff))))

[1] "MAE: 125.590163934426"

print(paste0("Durchschnittswert der Abweichungen (MeanError): ", mean(eval$sact_diff)))

[1] "Durchschnittswert der Abweichungen (MeanError): -46.3770491803279"

print(paste0("In ", round(nrow(eval[eval$sact_diff < 0,]) / nrow(eval) * 100, 2), "% der Fälle w

[1] "In 47.54% der Fälle wurde zuwenig Personal vorhergesagt (% low)"

print(paste0("Summe der mangelhaften negativen Vorhersagen (sum low): ", sum(eval[eval$sact_diff

[1] "Summe der mangelhaften negativen Vorhersagen (sum low): -5245"

print(paste0("Durchschnittswert der mangelhaften negativen Vorhersagen (mean low): ", mean(eval

[1] "Durchschnittswert der mangelhaften negativen Vorhersagen (mean low): -180.862068965517"

print(paste0("Größte negative Vorhersage (max low): ", min(eval[eval$sby_diff < 0,]$sact_diff)))

[1] "Größte negative Vorhersage (max low): -465"
```

Die nachfolgende Tabelle zeigt den Vergleich des Baseline Modells zum aktuellen Vorgehen.

Modell	RMSE	MAE	MeanError	% low	sum low	mean low	max low
Baseline	185,94	164,48	108,64	26,23	1703,19	106,45	237,44
Aktuelles Vorgehen	160,05	125,59	46,38	47,54	5245	180,86	465

Wie auch im obigen Vergleich zum Mean "Algorithmus" sind sowohl der RMSE als auch der MAE beim aktuellen Vorgehen besser bewertet. Die Verbesserung von TSLM wird deutlich, wenn es darum geht Tage mit hohem Personalbedarf ausreichend vorherzusagen. So wird bisher etwa bei der Hälfte der Tage zu wenig Bereitschaftspersonal vorgesehen. In Summe mussten in der aktuellen Vorgehensweise, in zwei Monaten Vorhersagezeitraum 5245 zusätzliche Bereitschaftsfahrer:innen aktiviert werden. Mit TSLM können die zwei Werte wesentlich verbessert werden.

## Speichern modelspezifischer Daten

```
Speichern der Sampledaten (hier alles)
save(ts_sby_week, ts_sby_test, ts_sby_train, eval, ts_sby,
 file = "../00_sample_data/03_for_modeling/baseline.rda")
```

Das Modell selbst wird im Ordner **03\_deployment** des Modellverzeichnis als **sby\_baseline.R** Datei abgelegt.

```
rm(list = ls()) # nach Durchlauf alles löschen
```

## Literaturverzeichnis

Vermorel, Joannès. 2012. „Pinball Loss Function Definition“. <https://www.lokad.com/pinball-loss-function-definition/>.

## A.4 Advanced Model

### Einleitung

Dieses Dokument beschreibt die Erarbeitung eines ersten fortschrittlicheren Modells. Hierbei wird auf Erkenntnissen einer vorangegangenen Use Case Analyse (Grunsky 2024), sowie auf den Ergebnissen der explorativen Datenanalyse aufgebaut.

Der Fokus liegt hierbei auf einer Modellierung der in der explorativen Datenanalyse festgehaltenen Saisonalität der eingegangenen Notrufe (*calls*) und der, ab einem festgelegten Schwellwert, linearen Korrelation dieser zum benötigten Bereitschaftspersonal (*sby\_need*) - unter Berücksichtigung der Anzahl des Dienstabenden Personals (*n\_duty*).

Die in der Use Case Analyse vorgeschlagene, fallspezifische Kostenfunktion wird in der u.a. Modellentwicklung jedoch noch nicht implementiert und die Evaluierung der Einfachheit halber mit herkömmlichen Methoden durchgeführt. Bekannte Messgrößen erleichtern hierbei auch die Interpretierbarkeit der Modellgüte und müssen nicht erst im Voraus erklärt werden.

Im R-package *fpp3* (**hyndman\_forecasting\_2021?**) ist unter anderem der Algorithmus **STL** (Seasonal and Trend decomposition using Loess) enthalten, auf den sich die gegebene Modellentwicklung konzentriert. Weitere saisonale Algorithmen wie **TSLM** und **ETS**, sowie die einfache Darstellung des **Drift** dienen in der optischen Performancemessung als Vergleich zu STL.

Wie auch in der Entwicklung des Baselinemodells liegt die Herausforderung vor allem im raschen Wechsel zwischen sehr hohem und fast keinem Bedarf an Bereitschaftspersonal. Das empfohlene Mindestmaß an etwa 35 Personen in Bereitschaft wird in der Entwicklung des Modells noch nicht berücksichtigt, aber in einem finalen Deployment als Mindestwert implementiert.

Das gewünschte Ergebnis ist eine möglichst akkurate Vorhersage der Bedarfsspitzen, eine ausreichende Generalisierung des Modells und ein Erhalt der Volatilität um, im Vergleich zu den bisher fix eingesetzten 90 Fahrer:innen, Bereitschaftskosten einsparen zu können.

### Daten laden

```
load(file = "../00_sample_data/02_processed/data_explorative.rda")
```

### Aufsplitten der Daten in Trainings- und Testdaten

Die Autokorrelation des Merkmals *calls* ist, entsprechend der explorativen Datenanalyse, deutlich besser zu bewerten als jene der Zielvariablen. Dennoch soll auch in diesem Modell die Änderung des saisonalen Verhaltens mit dem Jahreswechsel auf 2019 mit berücksichtigt werden. Der Testdatensatz wird daher auch in diesem Modell mit 2 Monaten angesetzt.

```
ts_sby_train <- ts_sby %>%
 select(date, calls, MA2_8, n_duty, sby_need) %>%
 filter(!is.na(MA2_8)) %>%
 arrange(date) %>%
 slice(0:(n() - 62))

ts_sby_test <- ts_sby %>%
 select(date, calls, MA2_8, n_duty, sby_need) %>%
 filter(!is.na(MA2_8)) %>%
```



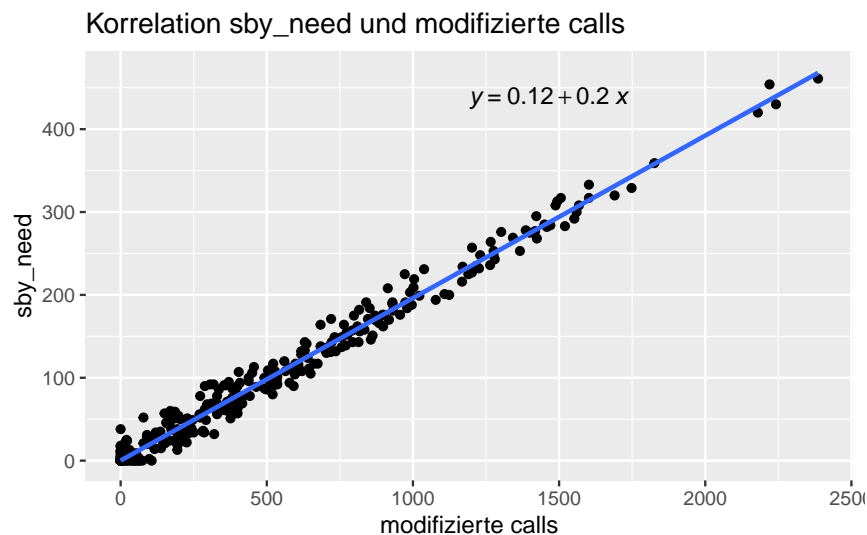
```
arrange(date) %>%
slice((n() - 61):n())
```

## Modelerstellung

### Lineares Modell

Im ersten Schritt wird das lineare Modell umgesetzt, das die Korrelation von *calls* und *sby\_need* unter Einfluss von *n\_duty* abbildet. Dabei wurde festgehalten, dass sich eine Änderung des diensthabenden Personals um etwa das fünffache auf den Schwellwert auswirkt, ab dem Bereitschaftspersonal benötigt wird. Wie auch in der explorativen Datenanalyse beschrieben, wird dieser Umstand im linearen Modell berücksichtigt. Die angeführte Grafik zeigt die lineare Abhängigkeit, nach Anpassung der Notrufe aufgrund von *n\_duty* und nach Abzug eines einheitlichen Schwellwertes von 8150 Anrufen im Trainingsdatensatz.

```
ggplot(ts_sby_train,
 aes(x = (pmax(0, (calls - (n_duty - 1700) * 5) - 8150)),
 y = sby_need
)) +
 geom_point() +
 stat_smooth(method = "lm", se = TRUE) +
 stat_regline_equation(label.x.npc = "center") +
 labs(x = "modifizierte calls",
 title = "Korrelation sby_need und modifizierte calls")
```



```
Lineares Modell erzeugen
lm_sby = lm(sby_need ~ (pmax(0, (calls - (n_duty - 1700) * 5) - 8150)),
 data = ts_sby_train)
```

## Advanced Forecast

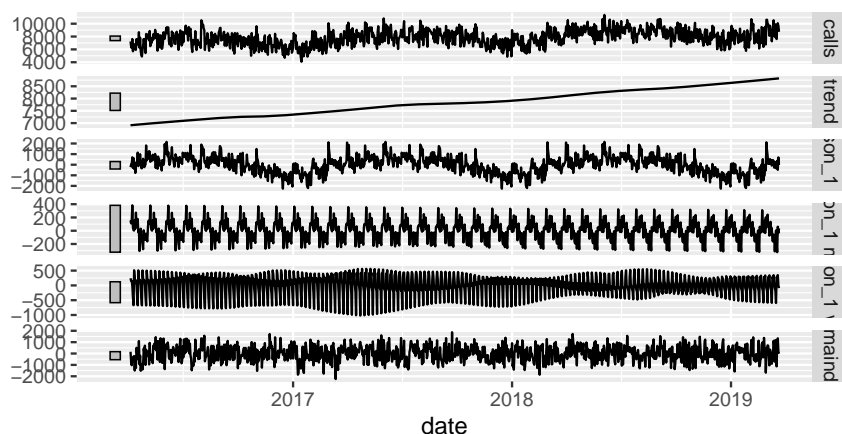
In der explorativen Datenanalyse wurde das Merkmal der eingegangenen Notrufe *calls* in seiner Saisonalität als das am besten geeignete Merkmal für eine Vorhersage identifiziert. Eine erste Dekomposition der saisonalen Komponenten mit STL zeigte bereits gute Ergebnisse jedoch mit einer vermuteten “Restsaisonalität” im Remainder. Dieser wurde mit Hinzufügen der Wochenperioden begegnet, die das Ergebnis noch weiter verbessert.

```
decomp <- ts_sby_train %>%
 model(
 STL(calls ~ trend(window = 365) +
 season(period = "1 year", window = 540) +
 season(period = "1 month", window = 61) +
 season(period = "1 week", window = 28))) %>%
 components()

decomp %>% autoplot()
```

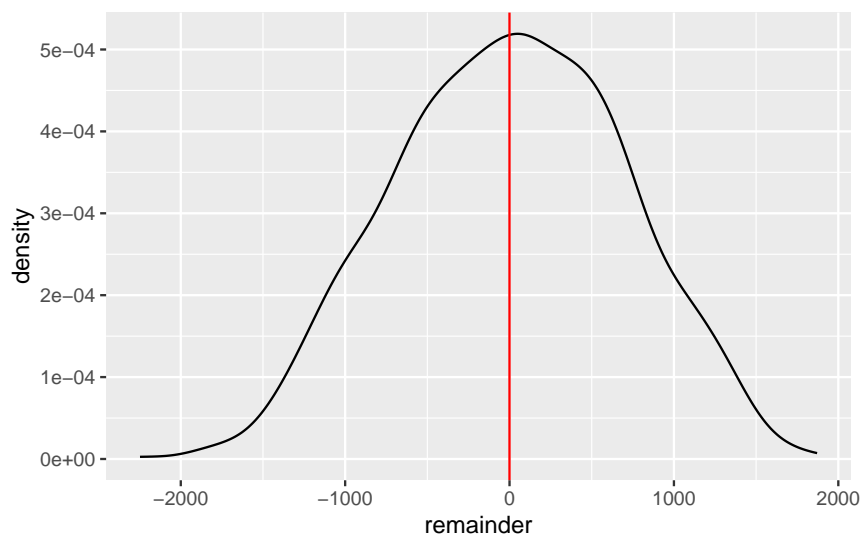
### STL decomposition

calls = trend + `season\_1 year` + `season\_1 month` + `season\_1 week` + remainder



Wie die u.a. Darstellung verdeutlicht sind die “Remainder” nahezu um den Wert 0 (*mean* in Rot) Normverteilt und entsprechen damit einer Annäherung an ein optimales Ergebnis.

```
ggplot(data = decomp, aes(x = remainder)) +
 geom_density() +
 geom_vline(xintercept = mean(decomp$remainder), color = "red")
```



Der nachfolgende Code erstellt die zu vergleichenden Modells, wobei der Fokus auf STL als *decomposition model* liegt. Das Modell ermittelt die oben dargestellte Saisonalität und wendet einen einfachen *Drift Algorithmus* auf die saisonal bereinigten Daten an.

```
decomp_spec <- decomposition_model(
 STL(calls ~ trend(window = 365) +
 season(period = "1 year", window = 540) +
 season(period = "1 month", window = 61) +
 season(period = "1 week", window = 28)),
 RW(season_adjust ~ drift())
)

Training
t <- system.time(
 progressr::with_progress(
 sby_model_adv <- ts_sby_train %>%
 model(
 stl = decomp_spec,
 ets = ETS(calls ~ error("A") + trend("A") + season("A")),
 drift = RW(calls ~ drift()),
 tslm = TSLM(calls ~ trend() + season())
)
), gcFirst = TRUE
)

print(paste0(" Die Benötigte Zeit für das Training betrug ", round(t[3],2), " Sekunden."))
```

```
[1] " Die Benötigte Zeit für das Training betrug 0.37 Sekunden."
```

```
Vorhersage
t <- system.time(
 progressr::with_progress(
 sby_fc_adv <- sby_model_adv %>%
```

```

forecast(h = "11 months", level = c(80))
), gcFirst = TRUE
)

print(paste0(" Die Benötigte Zeit für die Vorhersage war ", round(t[3],2), " Sekunden."))

```

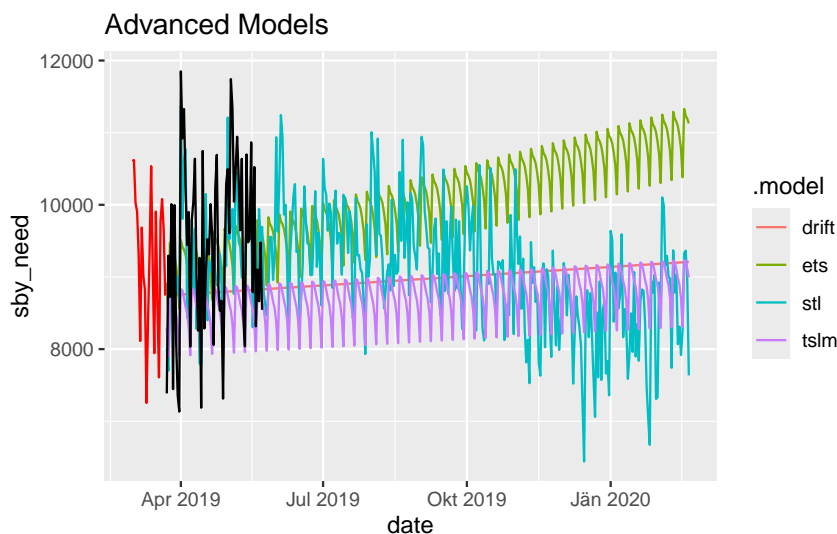
```
[1] " Die Benötigte Zeit für die Vorhersage war 0.6 Sekunden."
```

Die benötigte Zeit für die Berechnung der Modelle sowie für das Forecasting kann durchaus als “verschwindend” bezeichnet werden. Von dieser Seite ist keine weitere Hürde für die Umsetzung der Modelle zu erwarten. In der späteren Anwendung wird natürlich nur ein Forecastzeitraum von 2 Monaten benötigt, hier ermöglichen parametrisierten 11 Monate jedoch die Beurteilung, ob das Modell die jährliche Saisonalität und den Trend “verstanden” hat.

```

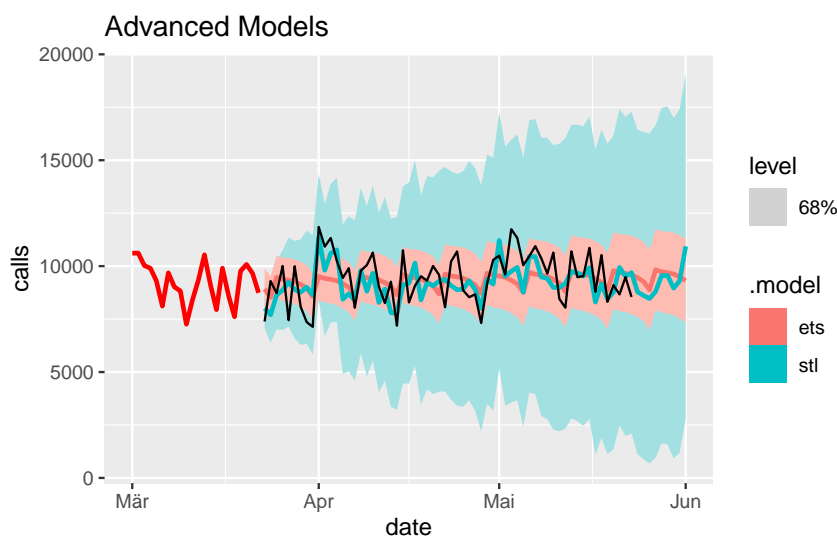
sby_fc_adv %>%
 autoplot(ts_sby_test,
 level = NULL) +
 autolayer(ts_sby_train %>% filter(date >= "2019-03-01"), colour = "red") +
 labs(y = "sby_need",
 title = "Advanced Models")

```



In dem o.a. Plot sind die Trainingsdaten Rot und die Testdaten Schwarz dargestellt. Die Vorhersagen durch die einzelnen Algorithmen sind farblich in der Legende kodiert. In der visuellen Bewertung wurde der allgemeine Trend im *Drift* gut erkannt und dargestellt. Selbstverständlich wird dieser Algorithmus aber zu keinem zufriedenstellenden Ergebnis führen. *ETS* und *TSLM* lassen zusätzlich noch eine periodische monatliche Saisonalität erkennen, setzen aber beide zu tief an. Eventuell könnte bei diesen Modellen mit einer Quantilsschätzung ein brauchbares Ergebnis erzielt werden. Die Tage mit wenig Bedarf sowie die jährliche Saisonalität sind allerdings nicht abgebildet. Überraschend gut wird die Volatilität und die Jahresperiode im *STL* modelliert. Bei näherer Betrachtung erkennt man, dass die Monatspeaks nahezu am Punkt sind (wenn auch meistens einen Tick zu niedrig), aber oft eben auch um ein oder zwei Tage verschoben. Ein Overfitting auf die Trainingsdaten scheint naheliegend. Möglicherweise hilft der Einsatz des Konfidenzintervalls um die Kurve zu glätten und auch die hohen

```
sby_fc_adv %>%
 filter(.model == "stl" | .model == "ets",
 date <= "2019-06-01") %>%
 autoplot(ts_sby_test,
 level = c(68),
 linewidth = 1) +
 autolayer(ts_sby_train %>% filter(date >= "2019-03-01"), colour = "red", linewidth = 1) +
 labs(y = "calls",
 title = "Advanced Models")
```



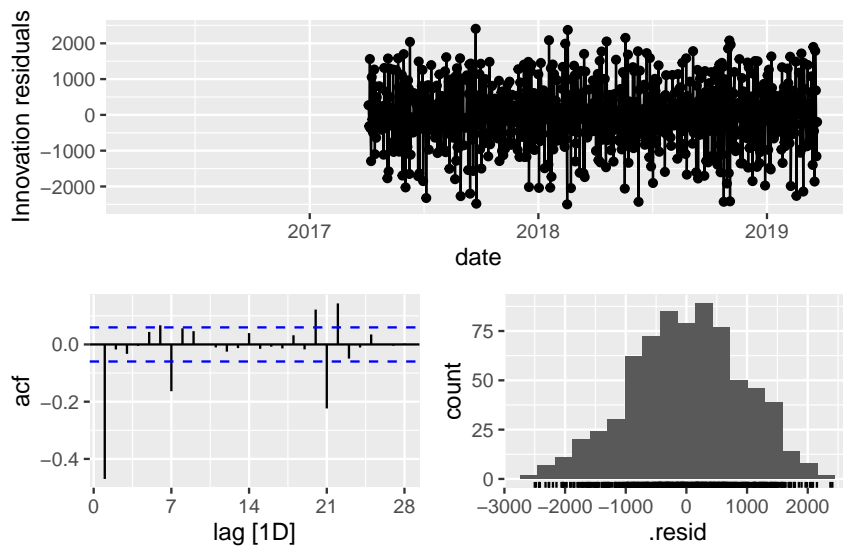
Im Vergleich von *STL* und *ETS* mit 68% Konfidenzintervall (+/- eine Standardabweichung) für eine Vorhersage von zwei Monaten sieht man, dass *ETS* mit guter Generalisierung die Spitzen meistens abdeckt, aber die Volatilität nicht hinbekommt. Bei *STL* wird mit zunehmender vorhersagedauer die Verwendung des Konfidenzintervalls unbrauchbar, weil die Streuung enorm ist. Die Entscheidung fällt auf eine Verwendung von *STL* in der Weiterentwicklung des Modells. Zu beachten ist, dass die fehlende Generalisierung nicht durch Verwendung des Konfidenzintervalls erzielt werden kann. Um die Volatilität des Modells zu erhalten ist der Punktschätzer zu verwenden und die Generalisierung anderweitig zu ermöglichen.

## Evaluierung

Die nachstehenden Werte berechnen die Güte des Modells analog zum Baselinemodell. Vergleichbar sind diese, aufgrund der unterschiedlichen Prädiktoren aber nicht. Sollten noch weitere Modelle auf Basis von *calls* erstellt werden, können diese Werte herangesogen werden.

Der u. a. angeführte Residualsplot zeigt ein nahezu optimales Verhalten des Algorithmus. In den Residuals ist auf den ersten Blick kein "Rest-Pattern" in den Daten zu beobachten und genauso wenig im Autokorrelationswert. Die Residuals wären im besten Fall mit einem arithmetischen Mittel von 0 normal verteilt. Das trifft nicht ganz zu, die Annäherung an das gewünschte Verhalten ist aber gut erkennbar. Dieser Plot zeigt allerdings nur, wie gut die Realität mit dem erstellten Modell (den sogenannten *fitted values*) erklärt werden kann. Es ist keine Bewertung der Vorhersagefähigkeit.

```
sby_model_adv %>%
 select(stl) %>%
 gg_tsresiduals()
```



```
sby_fc_adv %>%
 accuracy(ts_sby, list(RMSE = RMSE, MAE = MAE)) %>%
 select(-.type)
```

```
A tibble: 4 x 3
 .model RMSE MAE
 <chr> <dbl> <dbl>
1 drift 1314. 1086.
2 ets 1063. 915.
3 stl 990. 828.
4 tslm 1388. 1131.
```

Auch die verwendeten Messwerte RMSE und MAE für die Messung der Güte zeigen, dass *STL* am Besten bewertet wird. Wie auch in der Bewertung des Baselinemodells ist zu berücksichtigen, dass diese Messwerte nicht immer ausreichend aussagekräftig den Bedarf des Use Case abbilden. Deswegen gilt es mehrere Faktoren anzusehen und sich die Vorhersagen auch zu veranschaulichen, was hier auch getan wird.

```
sby_fc_adv %>%
 accuracy(ts_sby,
 list(quantile_score = quantile_score), probs = 0.68) %>%
 select(-.type)
```

```
A tibble: 4 x 2
 .model quantile_score
 <chr> <dbl>
1 drift 1783.
```

```
3 stl 1448.
4 tslm 1071.
```

In der Bewertung der Verteilung durch den Quantile Score, sieht man die Auswirkung der breiten Streuung von *STL* und die bereits festgestellte Unbrauchbarkeit einer Schätzung durch das Konfidenzintervall. Wie ebenso bereits anhand der Visualisierung beurteilt schneidet *ETS* hier deutlich besser ab.

## Vorhersage sby\_need

In diesem Abschnitt wird nun das zuvor erstellte lineare Modell auf die Vorhersage von *calls* angewandt. In der Einleitung wurden die Anforderungen an das Ergebnis bereits festgehalten

- akkurate Vorhersage der Bedarfsspitzen
- ausreichende Generalisierung des Modells
- Erhalt der Volatilität um Bereitschaftskosten einzusparen

Die fehlende Generalisierung wird erreicht indem ein gleitendes Maximum über die vorhergesagten *calls* laufen gelassen wird. "Verschobene Spitzen" sollten dadurch ebenfalls besser abgedeckt werden.

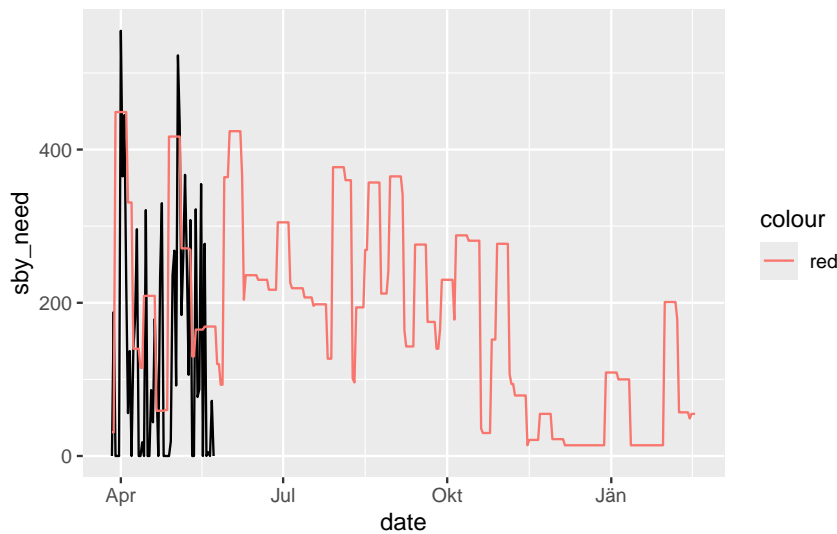
Der für das lineare Modell benötigte Wert für das diensthabende Personal, wird im u.a. Code festgelegt und dem neu gebildeten Datensatz mitgegeben. Diese Vorgehensweise ermöglicht später in der praktischen Anwendung ebenso, vorab das geplante diensthabende Personal für den Vorhersagezeitraum flexibel einzugeben, und erst dann die Vorhersage arbeiten zu lassen.

```
vorgesehenes diensthabendes Personal
duty_pers <- 1900

neuer Datensatz mit gleitenden Maxima zu Generalisierung (Prediction-Dataset)
call_pred_adv <- sby_fc_adv %>%
 filter(.model == "stl") %>%
 as.tibble() %>%
 select(date, .mean) %>%
 mutate(n_duty = duty_pers,
 calls = slider::slide_dbl(.mean,
 max,
 .before = 3, .after = 3,
 .complete = TRUE)) %>%
 filter(!is.na(calls)) %>%
 as_tsibble(index = date)

Anwendung lineares Modell und hinzufügen von sby zu Prediction-Dataset
sby_pred <- predict(lm_sby, call_pred_adv, interval = "prediction")
call_pred_adv$sby <- pmax(0, round(sby_pred[,3], 0))

Anzeige im Vergleich zu tatsächlichem sby_need
ggplot() +
 geom_line(data = filter(ts_sby_test, date > "2019-03-26"),
 aes(x = date, y = sby_need)) +
 geom_line(data = filter(call_pred_adv, date > "2019-03-26"),
 aes(x = date, y = sby, colour = "red"))
```



Die Generalisierung wurde erreicht, die Volatilität erhalten. Die Spitzenwerte werden oft nicht erreicht. Das Modell ist dennoch in dieser Form anwendbar. Eine korrekte Vorhersage der Spitzenwerte im Personalbedarf ist, auch wie im Basismodell, immer mit einer Gesamtkostenerhöhung für den Bereitschaftsdienst verbunden, da nahezu immer mehr als 90 Personen als Bedarf vorhergesagt werden.

## Evaluierung sby\_need

Nach dem es sich bei dem Prediction-Dataset nicht mehr um eine Forecast gem. *fpp3*-Package handelt, können die dort bereitgestellten Funktionen zur Bewertung nicht mehr angewandt werden. Für die Vergleichbarkeit mit dem Baselinemodell und ggf. weiteren fortgeschrittenen Modellen werden einige Bewertungskriterien daher "manuell" bereitgestellt.

```
Evaluierungsdatensatz - Abweichungen aus Forecast und realen Werten "sby_diff"
eval <- call_pred_adv %>%
 right_join(ts_sby_test, by = "date") %>%
 select(date, sby, sby_need) %>%
 filter(!is.na(sby)) %>%
 mutate(sby_diff = sby - sby_need)
```

Positive Werte von *sby\_diff* bedeuten, dass zuviel Bereitschaftspersonal vorhergesagt wurde und negative Werte besagen, dass der Bedarf nicht durch die Vorhersage abgedeckt wurde. Generell sind daher positive Werte gegenüber negativen vorzuziehen.

```
print(paste0("RMSE: ", sqrt(mean(eval$sby_diff^2))))
```

```
[1] "RMSE: 190.310094854135"
```

```
print(paste0("MAE: ", mean(abs(eval$sby_diff))))
```

```
[1] "MAE: 153.35593220339"
```



```
print(paste0("Durchschnittswert der Abweichungen: ", mean(eval$sby_diff)))

[1] "Durchschnittswert der Abweichungen: 88.4745762711864"

print(paste0("In ", round(nrow(eval[eval$sby_diff < 0,]) / nrow(eval) * 100, 2), "% der Fälle w

[1] "In 27.12% der Fälle wurde zuwenig Personal vorhergesagt"

print(paste0("Summe der mangelhaften negativen Vorhersagen: ", sum(eval[eval$sby_diff < 0,]$sby

[1] "Summe der mangelhaften negativen Vorhersagen: -1914"

print(paste0("Durchschnittswert der mangelhaften negativen Vorhersagen: ", mean(eval[eval$sby_c

[1] "Durchschnittswert der mangelhaften negativen Vorhersagen: -119.625"

print(paste0("Größte negative Vorhersage: ", min(eval[eval$sby_diff < 0,]$sby_diff)))

[1] "Größte negative Vorhersage: -271"
```

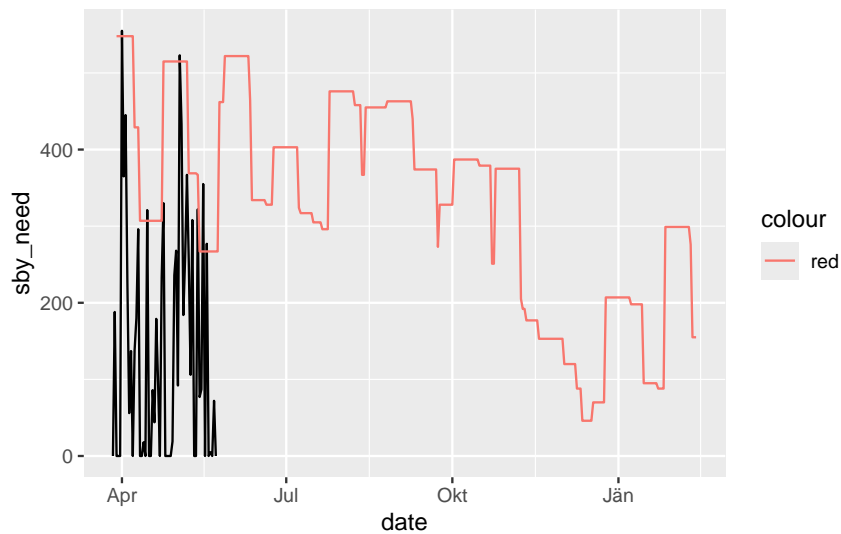
Testhalber wird in der nachgestellten Grafik die Anzahl des diensthabenden Personals um 100 gesenkt, um die Auswirkungen zu beobachten. Gleichzeitig wird der Zeitraum für das gleitende Maximum auf zwei Wochen erhöht um noch mehr Generalisierung zu erreichen.

```
vorgesehene diensthabendes Personal (Wert um 100 gesenkt!)
duty_pers <- 1800

neuer Datensatz mit gleitenden Maxima zu Generalisierung (Prediction-Dataset)
call_pred_adv <- sby_fc_adv %>%
 filter(.model == "st1") %>%
 as.tibble() %>%
 select(date, .mean) %>%
 mutate(n_duty = duty_pers,
 calls = slider::slide_dbl(.mean,
 max,
 .before = 6, .after = 7,
 .complete = TRUE)) %>%
 filter(!is.na(calls)) %>%
 as_tsibble(index = date)

Anwendung lineares Modell und hinzufügen von sby zu Prediction-Dataset
sby_pred <- predict(lm_sby, call_pred_adv, interval = "prediction")
call_pred_adv$sby <- pmax(0, round(sby_pred[,3], 0))

Anzeige im Vergleich zu tatsächlichem sby_need
ggplot() +
 geom_line(data = filter(ts_sby_test, date > "2019-03-26"),
 aes(x = date, y = sby_need)) +
 geom_line(data = filter(call_pred_adv, date > "2019-03-26"),
```



Die zweite Version sieht deutlich besser aus. Die Modifikation des diensthabenden Personals wird im Code für das Modell implementiert werden müssen, dadurch wird der vorhergesagte Bedarf höher angesetzt und ein Puffer eingebaut.

```
Evaluierungsdatensatz - Abweichungen aus Forecast und realen Werten "sby_diff"
eval <- call_pred_adv %>%
 right_join(ts_sby_test, by = "date") %>%
 select(date, sby, sby_need) %>%
 filter(!is.na(sby)) %>%
 mutate(sby_diff = sby - sby_need)

print(paste0("RMSE: ", sqrt(mean(eval$sby_diff^2))))

[1] "RMSE: 311.912046897839"

print(paste0("MAE: ", mean(abs(eval$sby_diff))))

[1] "MAE: 270.589285714286"

print(paste0("Durchschnittswert der Abweichungen: ", mean(eval$sby_diff)))

[1] "Durchschnittswert der Abweichungen: 266.053571428571"

print(paste0("In ", round(nrow(eval[eval$sby_diff < 0,]) / nrow(eval) * 100, 2), "% der Fälle w

[1] "In 8.93% der Fälle wurde zuwenig Personal vorhergesagt"

print(paste0("Summe der mangelhaften negativen Vorhersagen: ", sum(eval[eval$sby_diff < 0,]$sby

[1] "Summe der mangelhaften negativen Vorhersagen: -127"
```

```
print(paste0("Durchschnittswert der mangelhaften negativen Vorhersagen: ", mean(eval[eval$sby_c
```

```
[1] "Durchschnittswert der mangelhaften negativen Vorhersagen: -25.4"
```

```
print(paste0("Größte negative Vorhersage: ", min(eval[eval$sby_diff < 0,]$sby_diff)))
```

```
[1] "Größte negative Vorhersage: -88"
```

Jene Werte, die die allgemeine Modellgüte ausdrücken wurden durch die Anpassung verschlechtert. Profitiert hat hingegen die Anforderung ausreichend Bereitschaftspersonal vorherzusagen. Für eine entgeltliche Entscheidung wird ggf, das Management mit einbezogen, bzw. mit dem Management gemeinsam eine gültige Kostenfunktion zur Bewertung erstellt werden müssen. Ein Beispiel für so eine Kostenfunktion wurde bereits in der Use Case Analyse vorgeschlagen.

## Vergleich der Modelle

Modell	RMSE	MAE	MeanError	% low	sum low	mean low	max low
Baseline	185,94	164,48	108,64	26,23	1703,19	106,45	237,44
Aktuelles Vorgehen	160,05	125,59	46,38	47,54	5245	180,86	465
Advanced Version 1	190,31	153,36	88,47	27,12	1914	119,62	271
Advanced Version 2	311,91	270,59	266,05	8,93	127	25,4	88

Im Vergleich entsprechen der RMSE und MAE der ersten Version dieses Modells dem Baseline-Modell, ansonsten ist das Baseline-Modell jedoch besser zu bewerten als Version 1 des Advanced Modells. Die Version 2 schneidet beim RMSE und MAE wesentlich schlechter ab. Dennoch werden in der zweiten Version deutlich weniger Fälle, zu niedriger Schätzungen verzeichnet. Hier liegt die wesentliche Verbesserung im Advanced-Modell (V2). Es gilt zu klären, wo der Schwerpunkt der Vorhersage liegen soll. Vorerst wird mit der Version 2 des Advanced-Modells weitergearbeitet.

## Speichern modelspezifischer Daten

```
Speichern der Sampledaten (hier alles)
save(ts_sby_test, ts_sby_train, eval, call_pred_adv, ts_sby,
 file = "../00_sample_data/03_for_modeling/advanced.rda")
```

Das Modell selbst wird im Ordner **03\_deployment** des Modellverzeichnis als **sby\_model\_advanced**. Datei abgelegt.

```
rm(list = ls()) # nach Durchlauf alles löschen
```

## Literaturverzeichnis

Grunsky, Georg. 2024. „Rettungsdienst Berliner Rotes Kreuz MachineLearning-Einsatz in der Bereit-

## A.5 Prophet Model

### Einleitung

Dieses Dokument beschreibt die Erarbeitung eines weiteren fortschrittlichen Modells. Hierbei wird ebenso auf Erkenntnissen der vorangegangenen Use Case Analyse (Grunsky 2024), sowie auf den Ergebnissen der explorativen Datenaanalyse aufgebaut.

Der Fokus liegt auch hierbei auf einer Modellierung der in der explorativen Datenanalyse festgehaltenen Saisonalität der eingegangenen Notrufe (*calls*) und der, ab einem festgelegten Schwellwert, linearen Korrelation dieser zum benötigten Bereitschaftspersonal (*sby\_need*) - unter Berücksichtigung der Anzahl des Dienshabenden Personals (*n\_duty*).

In der Use Case Analyse wurden die überblicksmäßig die Stärken und Schwächen verschiedener Machine Learning Algorithmen verglichen. Aufgrund der Einfachheit, Interpretierbarkeit und Tauglichkeit für saisonale Zeitreihen mit wenig Features schien sich **Facebook Prophet** als möglicher Algorithmus für den gegenständlichen Anwendungsfall zu eignen. Dieser Algorithmus wurde für Zeitreihenvorhersagen entwickelt und verwendet dabei ein zusammengesetztes Modell aus Trend, Saisonalität und speziellen Ereignissen in Zeitreihen. (Taylor und Letham, o. J.).

Der Vorteil von **Facebook Prophet** liegt darin, dass in einer möglichen, zukünftigen Weiterentwicklung des Modells auch Tage mit besonderen Ereignissen, wie zB Veranstaltungen und speziellen Wetterverhältnissen, mit modelliert werden und a priori in eine Vorhersage mit einfließen könnten.

Im R-package *fpp3* (**hyndman\_forecasting\_2021?**) ist zwar *Prophet* als nicht enthalten, kann aber mit laden des Packages *fable.prophet* hinzugefügt werden.

Wie auch in der Entwicklung des Baselinemodells liegt die Herausforderung vor allem im raschen Wechsel zwischen sehr hohem und fast keinem Bedarf an Bereitschaftspersonal. Das empfohlene Mindestmaß an etwa 35 Personen in Bereitschaft wird in der Entwicklung des Modells noch nicht berücksichtigt, aber in einem finalen Deployment als Mindestwert implementiert.

Das gewünschte Ergebnis ist eine möglichst akkurate Vorhersage der Bedarfsspitzen, eine ausreichende Generalisierung des Modells und ein Erhalt der Volatilität um, im Vergleich zu den bisher fix eingesetzten 90 Fahrer:innen, Bereitschaftskosten einsparen zu können.

### Daten laden

```
load(file = "../00_sample_data/02_processed/data_explorative.rda")
```

### Aufsplitten der Daten in Trainings- und Testdaten

Die Autokorrelation des Merkmals *calls* ist, entsprechend der explorativen Datenanalyse, deutlich besser zu bewerten als jene der Zielvariablen. Dennoch soll auch in diesem Modell die Änderung des saisonalen Verhaltens mit dem Jahreswechsel auf 2019 mit berücksichtigt werden. Der Testdatensatz wird daher auch in diesem Modell mit 2 Monaten angesetzt.

```
ts_sby_train <- ts_sby %>%
 select(date, calls, MA2_8, n_duty, sby_need) %>%
 filter(!is.na(MA2_8)) %>%
 arrange(date) %>%
 slice(0:(n() - 62))
```

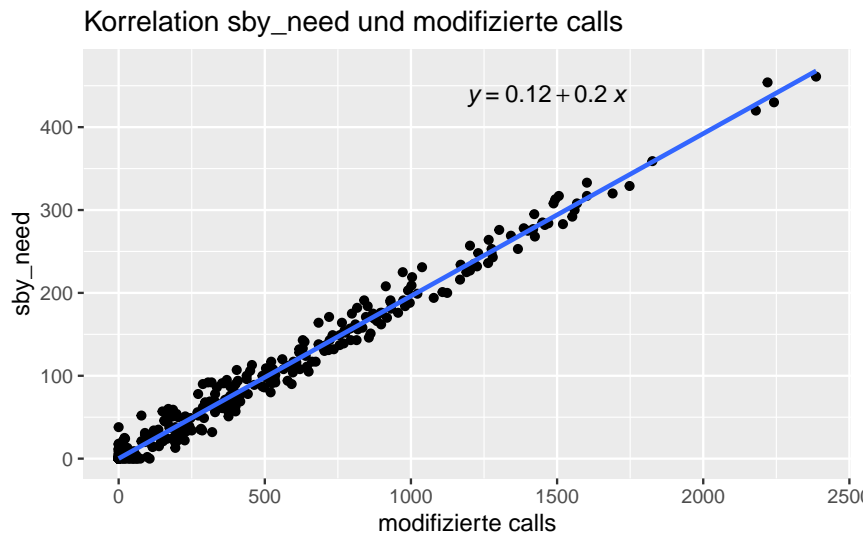
```
ts_sby_test <- ts_sby %>%
 select(date, calls, MA2_8, n_duty, sby_need) %>%
 filter(!is.na(MA2_8)) %>%
 arrange(date) %>%
 slice((n() - 61):n())
```

## Modelerstellung

### Lineares Modell

Im ersten Schritt wird erneut das lineare Modell umgesetzt, das die Korrelation von *calls* und *sby\_need* unter Einfluss von *n\_duty* abbildet. Die angeführte Grafik zeigt, wie auch im Advanced-Modell die lineare Abhängigkeit, nach Anpassung der Notrufe aufgrund von *n\_duty* und nach Abzug eines einheitlichen Schwellwertes von 8150 Anrufen im Trainingsdatensatz.

```
ggplot(ts_sby_train,
 aes(x = (pmax(0, (calls - (n_duty - 1700) * 5) - 8150)),
 y = sby_need
)) +
 geom_point() +
 stat_smooth(method = "lm", se = TRUE) +
 stat_regline_equation(label.x.npc = "center") +
 labs(x = "modifizierte calls",
 title = "Korrelation sby_need und modifizierte calls")
```



```
lm_sby = lm(sby_need ~ (pmax(0, (calls - (n_duty - 1700) * 5) - 8150)),
 data = ts_sby_train)
```

## Prophet Forecast

Der nachfolgende Code erstellt das Prophet-Modell mit einem linearen Trend, sowie jährlichen, monatlichen und wöchentlichen Saisonalitäten.

```
Training
t <- system.time(
 progressr::with_progress(
 sby_model_prophet <- ts_sby_train %>%
 model(
 prophet = prophet(calls ~ growth("linear",
 n_changepoints = 15) +
 season(period = 7,
 order = 20,
 type = "additive",
 name = "week") +
 season(period = 28,
 order = 10,
 type = "multiplicative",
 name = "month") +
 season(period = "year",
 order = 15,
 type = "multiplicative",
 name = "year")
)
)
)
)

print(paste0(" Die Benötigte Zeit für das Training betrug ", round(t[3],2), " Sekunden."))
```

```
[1] " Die Benötigte Zeit für das Training betrug 3.32 Sekunden."
```

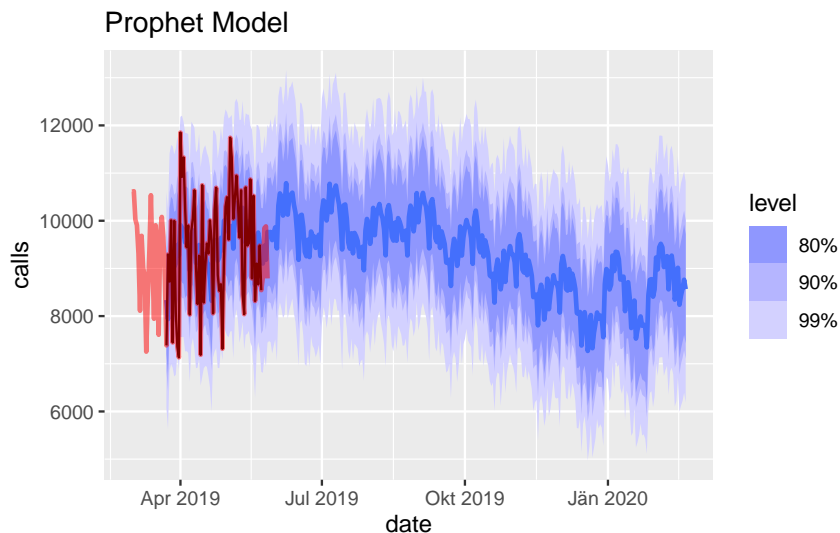
```
Vorhersage
t <- system.time(
 progressr::with_progress(
 sby_fc_prophet <- sby_model_prophet %>%
 #forecast(h = "2 months") %>%
 forecast(h = "11 months")
), gcFirst = TRUE
)

print(paste0(" Die Benötigte Zeit für die Vorhersage war ", round(t[3],2), " Sekunden."))
```

```
[1] " Die Benötigte Zeit für die Vorhersage war 10.28 Sekunden."
```

Die benötigte Rechenzeit ist bei *prophet* zwar deutlich höher als bei den vorangegangenen Modellen, liegt aber immer noch im Bereich weniger Sekunden und ist damit gut einsetzbar.

```
sby_fc_prophet %>%
 autoplot(ts_sby_test,
 level = c(80,90,99), linewidth = 1) +
 autolayer(ts_sby %>% filter(date >= "2019-03-01"),
 colour = "red", linewidth = 1, alpha = 0.5) +
 labs(y = "calls",
 title = "Prophet Model")
```



Die oben angeführte Grafik zeigt die tatsächlich eingegangenen Anrufe in Rot, die gleitende 99% Quantile in Schwarz und die Vorhersage für elf Monate in blauer Farbe.

Die Glättung durch die gleitenden Quantile ist gut erkennbar und hüllt die tatsächlichen Notrufe ein. Die Spitzenwerte werden dadurch erfolgreich abgedeckt, was jedoch auf Kosten der Tage mit niedrigem Bedarf geschieht.

Das Modell schafft es daher mit Masse, die Spitzenwerte abzudecken. In einigen wenigen Fällen gelingt dies trotzdem nicht. Die Saisonalitäten sind deutlich erkennbar, auch wenn der zu erwartende Abfall in den Wintermonaten augenscheinlich zu wenig ausgeprägt ist.

In diesem Plot wurden testhalber auch die Konfidenzintervalle für die 80%, 90% und 99% Quantile dargestellt. Im Gegensatz zum *STL*-Algorithmus des *Advanced-Modells*, bleiben diese hier auch über einen Vorhersagezeitraum von 11 Monaten relativ konstant und weisen daher ebenso eine Volatilität auf. Diese Volatilität könnte für die Vorhersage von Tagen mit niedrigem Bedarf von Vorteil sein.

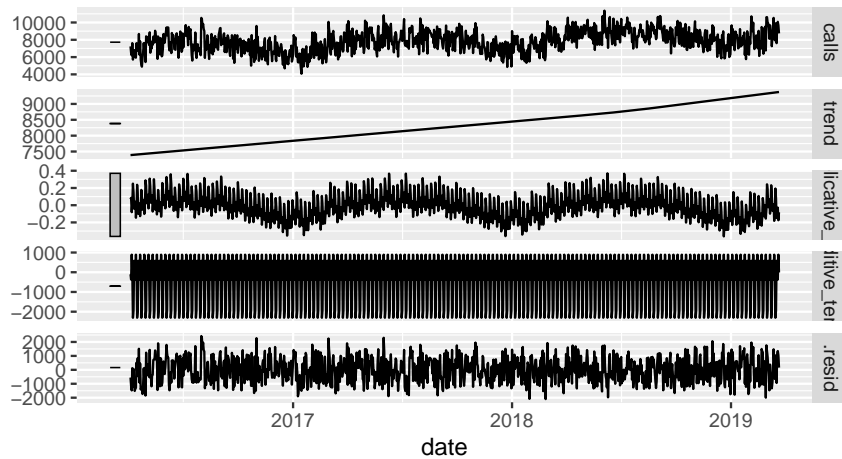
## Evaluierung

Gemäß den gesetzten Trainingsparametern unterscheidet der *prophet* Algorithmus additive und multiplikative Saisonalitäten. Der Unterschied liegt in der Variabilität der Saisonalitäten (multiplikativ ist hierbei variabler als additiv). Diese werden in der unten angeführten Decomposition des Modells gesondert angeführt. Die Residuals weisen augenscheinlich kein Auffälliges "Rest-Pattern" auf, die Modellierung scheint daher die Trainingsdaten gut aufzunehmen.

```
sby_model_prophet %>%
 components() %>%
```

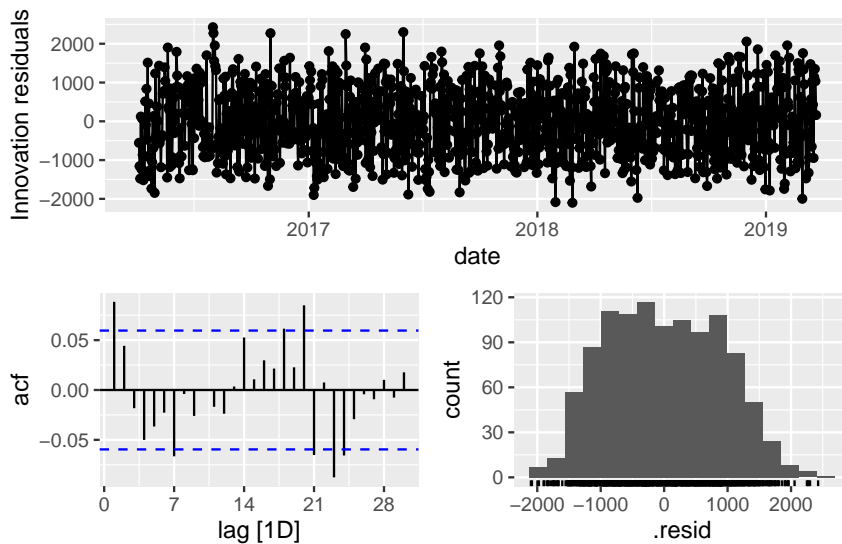
## Prophet decomposition

$\text{calls} = \text{trend} * (1 + \text{multiplicative\_terms}) + \text{additive\_terms} + \text{.resid}$



Der u. a. angeführte Residualsplot zeigt auch bei *Prophet* ein nahezu optimales Verhalten des Algorithmus. In den Residuals ist auf den ersten Blick kein deutliches "Rest-Pattern" in den Daten zu beobachten und genauso wenig im Autokorrelationswert. Die Verteilung der Residuals ist im Vergleich zum Advanced-Modell etwas breiter.

```
gg_tsresiduals(sby_model_prophet)
```



```
sby_fc_prophet %>%
 accuracy(ts_sby, list(RMSE = RMSE,
 MAE = MAE,
 quantile_score = quantile_score),
 probs = 0.68) %>%
 select(-.type)
```



```
A tibble: 1 x 4
 .model RMSE MAE quantile_score
 <chr> <dbl> <dbl> <dbl>
1 prophet 1026. 852. 746.
```

Der RMSE von Prophet-Modellen ist etwas schlechter als der des Advanced-Modells. Der MAE ist wiederum recht ähnlich. Hier wird allerdings nur die Vorhersage von *calls* in einem ersten Schuss bewertet, um zu sehen, ob eine weitere Arbeit mit diesem Modell sinnvoll erscheint.

Beim Quantile Score ist der Prophet-Algorithmus vergleichbar mit dem ETS-Algorithmus, der in der Modellierung des Advanced-Modells untersucht wurde, und damit deutlich besser als der des, schlussendlich verwendeten, STL-Algorithmus. Wie auch die o.a. Visualisierung zeigt, bleiben die Quantilen auch in einer weitreichenderen Forecast relativ konstant, wohingegen bei *STL* die unschärfe über die Zeit wesentlich zunahm und eine Verwendung der Quantile unbrauchbar machte.

## Vorhersage sby\_need

Im Advanced-Modell wurde versucht einen Puffer für die Spitzenwerte anhand eines gleitenden Maximums über die vorhergesagten Notrufe zu erreichen. Da bei *Prophet* die Verwendung von Konfidenzintervallen möglich erscheint, wird in diesem Ansatz der Puffer durch Verwendung der 90% Quantile als Vorhersagewert erreicht. Das Schöne hierbei ist, dass der zu verwendende Prozentsatz für die Berechnung einem codierten Modell auch als Hyperparameter mitgegeben werden kann.

```
sby_fc_prophet <- sby_fc_prophet %>%
 mutate(.mean = hilo(calls, 90)$upper)
```

Wie auch im Advance-Modell wird, zum Vergleich der Ergebnisse, die Höhe des diensthabenden Personals um 100 reduziert. Damit wird die Anzahl des vorhergesagten Bereitschaftspersonals etwas angehoben. Auch die Höhe des vorgesehenen Bereitschaftspersonals wird für das Modell ein verpflichtender Parameter sein. Die Planungsstelle für den Bereitschaftsdienst, sollte aber im Vorhinein wissen, wie viel Personal im vorherzusagenden Zeitraum regulär im Dienst sein soll.

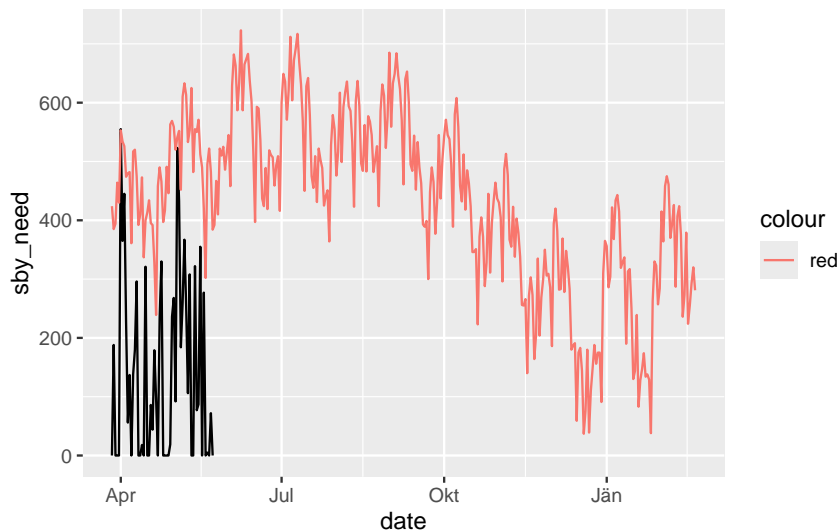
```
vorgesehenes diensthabendes Personal
duty_pers <- 1800

neuer Datensatz mit gleitenden Maxima zu Generalisierung (Prediction-Dataset)
call_pred_adv <- sby_fc_prophet %>%
 as.tibble() %>%
 select(date, .mean) %>%
 mutate(n_duty = duty_pers,
 calls = .mean) %>%
 filter(!is.na(calls)) %>%
 as_tsibble(index = date)

Anwendung lineares Modell und hinzufügen von sby zu Prediction-Dataset
sby_pred <- predict(lm_sby, call_pred_adv, interval = "prediction")
call_pred_adv$sby <- pmax(0, round(sby_pred[, 3], 0))

Anzeige im Vergleich zu tatsächlichem sby_need
ggplot() +
 geom_line(data = filter(ts_sby_test, date > "2019-03-26"),
 aes(x = date, y = sby_need)) +
```

```
geom_line(data = filter(call_pred_adv, date > "2019-03-26"),
 aes(x = date, y = sby, colour = "red"))
```



## Evaluierung sby\_need

Nachdem es sich bei dem Prediction-Dataset nicht mehr um eine Forecast gem. *fpp3*-Package handelt, können die dort bereitgestellten Funktionen zur Bewertung nicht mehr angewandt werden. Für die Vergleichbarkeit mit dem Baselinemodell und ggf. weiteren fortgeschrittenen Modellen werden einige Bewertungskriterien daher "manuell" bereitgestellt.

```
Evaluierungsdatensatz - Abweichungen aus Forecast und realen Werten "sby_diff"
eval <- call_pred_adv %>%
 right_join(ts_sby_test, by = "date") %>%
 select(date, sby, sby_need) %>%
 filter(!is.na(sby)) %>%
 mutate(sby_diff = sby - sby_need)
```

Positive Werte von *sby\_diff* bedeuten, dass zuviel Bereitschaftspersonal vorhergesagt wurde und negative Werte besagen, dass der Bedarf nicht durch die Vorhersage abgedeckt wurde. Generell sind daher positive Werte gegenüber negativen vorzuziehen.

```
print(paste0("RMSE: ", sqrt(mean(eval$sby_diff^2))))
```

```
[1] "RMSE: 356.535208413853"
```

```
print(paste0("MAE: ", mean(abs(eval$sby_diff))))
```

```
[1] "MAE: 328.516129032258"
```

```
print(paste0("Durchschnittswert der Abweichungen: ", mean(eval$sby_diff)))

[1] "Durchschnittswert der Abweichungen: 328.451612903226"

print(paste0("In ", round(nrow(eval[eval$sby_diff < 0,]) / nrow(eval) * 100, 2), "% der Fälle w

[1] "In 1.61% der Fälle wurde zuwenig Personal vorhergesagt"

print(paste0("Summe der mangelhaften negativen Vorhersagen: ", sum(eval[eval$sby_diff < 0,]$sby

[1] "Summe der mangelhaften negativen Vorhersagen: -2"

print(paste0("Durchschnittswert der mangelhaften negativen Vorhersagen: ", mean(eval[eval$sby_c

[1] "Durchschnittswert der mangelhaften negativen Vorhersagen: -2"

print(paste0("Größte negative Vorhersage: ", min(eval[eval$sby_diff < 0,]$sby_diff)))

[1] "Größte negative Vorhersage: -2"
```

## Vergleich der Modelle

Modell	RMSE	MAE	MeanError	% low	sum low	mean low	max low
Baseline	185,94	164,48	108,64	26,23	1703,19	106,45	237,44
Aktuelles Vorgehen	160,05	125,59	46,38	47,54	5245	180,86	465
Advanced Version 1	190,31	153,36	88,47	27,12	1914	119,62	271
Advanced Version 2	311,91	270,59	266,05	8,93	127	25,4	88
Prophet (Q90)	356,43	328,61	328,03	1,61	18	18	18
Prophet (Q80)	299,07	268,85	264,82	3,23	125	62,5	81

Für den Vergleich wurde der Prophet Algorithmus in zwei Versionen ausgeführt. In beiden Fällen wurde mit dem Konfidenzintervall gearbeitet, einmal mit der 80% Quantile und einmal mit der 90% Quantile. Die obige Tabelle zeigt, dass das Prophet-Modell in Verwendung des 80% Konfidenzintervalls ähnliche RMSE, MAE und MeanError Werte aufweist, wie das Advanced Model (V2), jedoch deutlich weniger oft zu niedrig schätzt. Man sieht auch gut, welchen Einfluss die Parametrisierung des Konfidenzintervalls auf das Ergebnis hat. Bei dem 90% Konfidenzintervall wird überhaupt nur einmal zu niedrig geschätzt, dafür sind eben die anderen Fehlerwerte deutlich schlechter. Die o.a. Grafik der Vorhersage lässt im Vergleich zu anderen Modellen jedoch erkennen, dass es Prophet, aufgrund der modellierten Volatilität schafft, in den Wintermonaten den Mindestbedarf wieder zu senken.

Der Quantilen-Parameters wird im deploten Modell flexibel setzbar sein. Der Defaultwert liegt bei

## Speichern modelspezifischer Daten

```
Speichern der Sampledaten (hier alles)
save(ts_sby_test, ts_sby_train, eval, call_pred_adv, ts_sby,
 file = "../00_sample_data/03_for_modeling/prophet.rda")
```

Das Modell selbst wird im Ordner **03\_deployment** des Modellverzeichnis als **sby\_model\_prophet.R** Datei abgelegt.

```
rm(list = ls()) # nach Durchlauf alles löschen
```

## Literaturverzeichnis

Grunsky, Georg. 2024. „Rettungsdienst Berliner Rotes Kreuz MachineLearning-Einsatz in der Bereitschaftsplanung - Use Case Analyse“, Dezember.

Taylor, Sean J, und Benjamin Letham. o. J. „Forecasting at scale“. <https://doi.org/10.7287/peerj.preprints.3190v2>.