Advanced Model

Georg Grunsky

2025-06-04

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 Datenaanalyse 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 Dienshabenden 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 Trianings- 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)) %>%
```

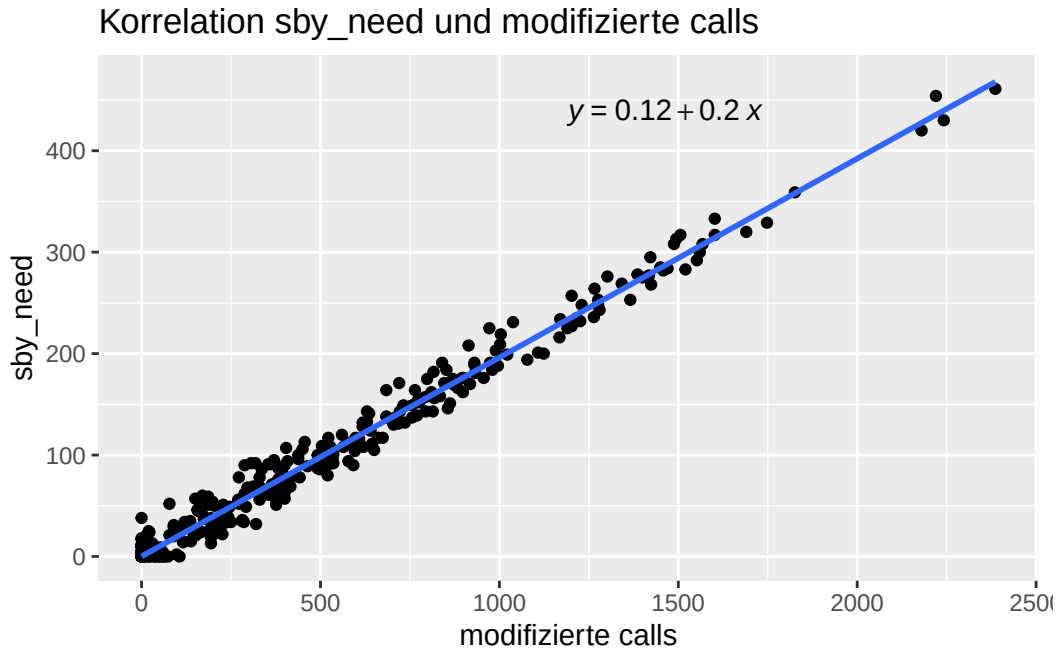
```
  arrange(date) %>%
  slice((n() - 61):n())
```

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")
```



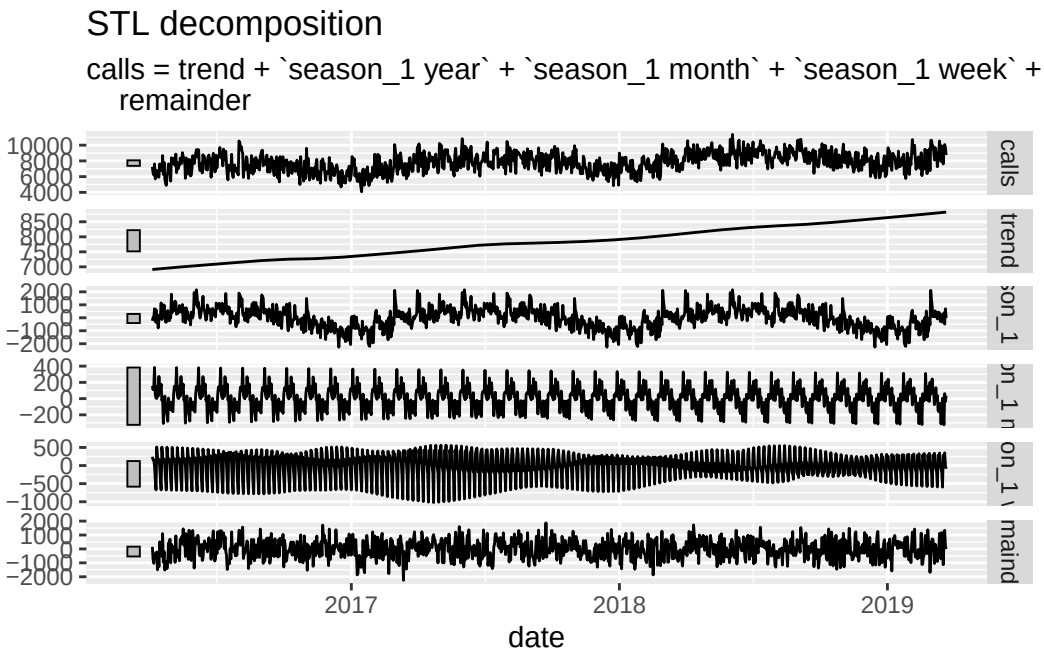

```
# Lineares Modell erzeugen
lm_sby = lm(sby_need ~ (pmax(0,(calls - (n_duty - 1700) * 5) - 8150)),
            data = ts_sby_train)
```

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.

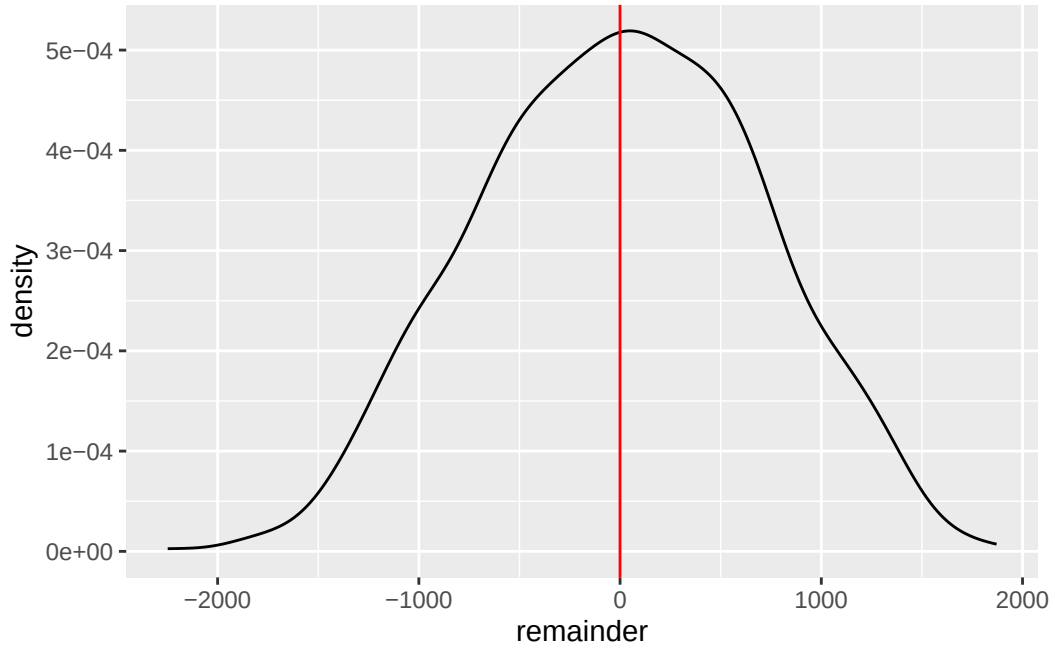
```
decomp <- ts_sby_train %>%
  model(
    STL(calls ~ trend(window = 365) +
                  season(period = "1 year", window = 540) +
                  season(period = "1 month", window = 61) +
                  season(period = "1 week", window = 28))) %>%
  components()

  decomp %>% autoplot()
```



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 = "2 months", level = c(80)) %>%
```

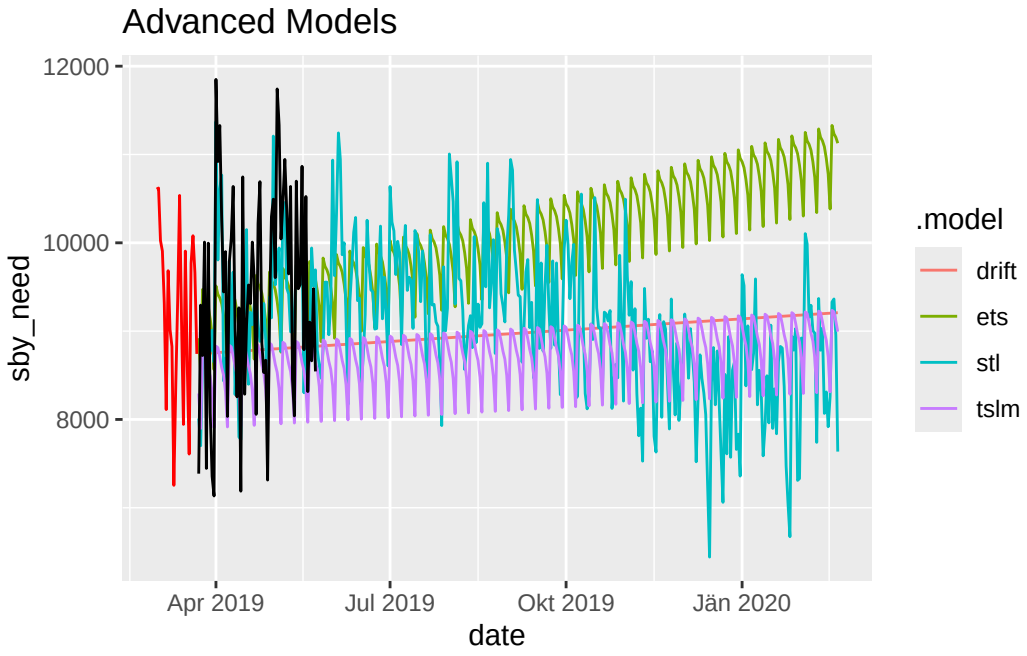
```
      forecast(h = "11 months", level = c(80))
  ), gcFirst = TRUE
)

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."
```

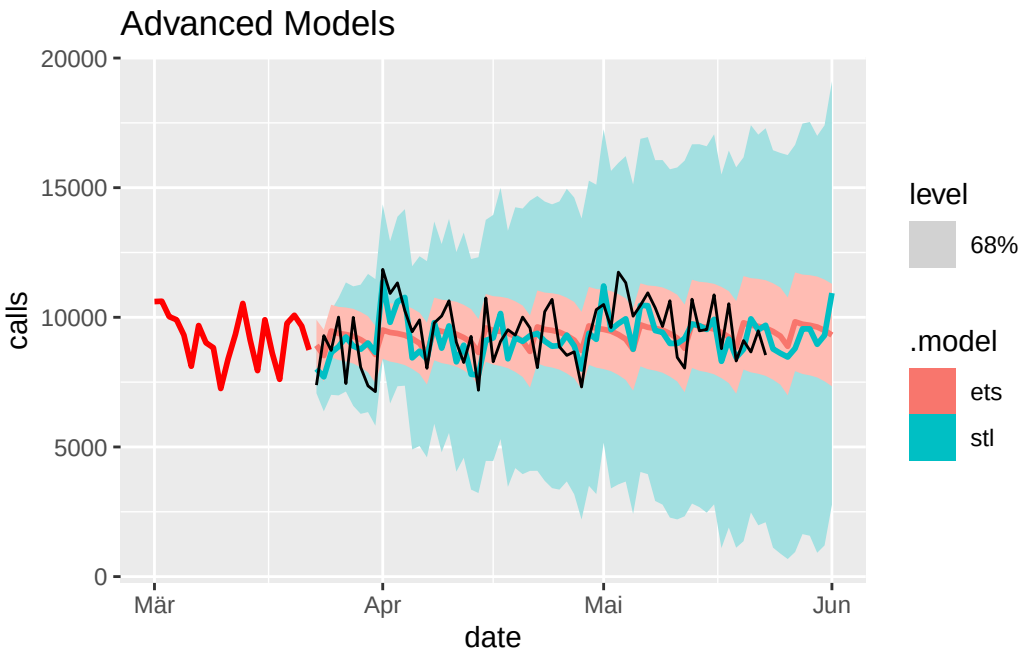
Die benötigte Zeit für die Berechnung der Modelle sowie für das Forecasting kann durchaus als "verschwindend" bezeichnet werden. Von dieser Seite ist keine weitere Hürde für die Umsetzung der Modelle zu erwarten. In der späteren Anwendung wird natürlich nur ein Forecastzeitraum von 2 Monaten benötigt, 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. Selbsverständlich 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 Peaks "einzusammeln".

```
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")
```



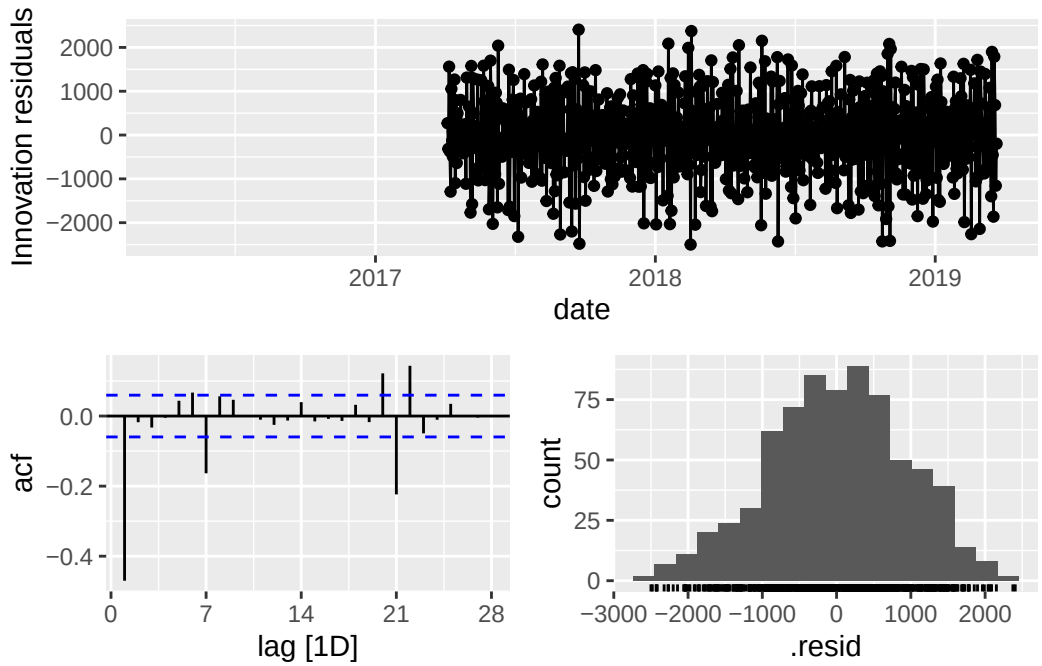
Im Vergleich von *STL* und *ETS* mit 68% Konfidenzinterwall (+/- eine Standardabweichung) für eine Vorhersage von zwei Monaten sieht man, dass *ETS* mit guter Generalisierung die Spitzen meistens abdeckt, aber die Volatilität nicht hinbekommt. Bei *STL* wird mit zunehmender vorhersagedauer die Verwendung 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 unterschieldichen 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.
2 ets              778.
```

```
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.

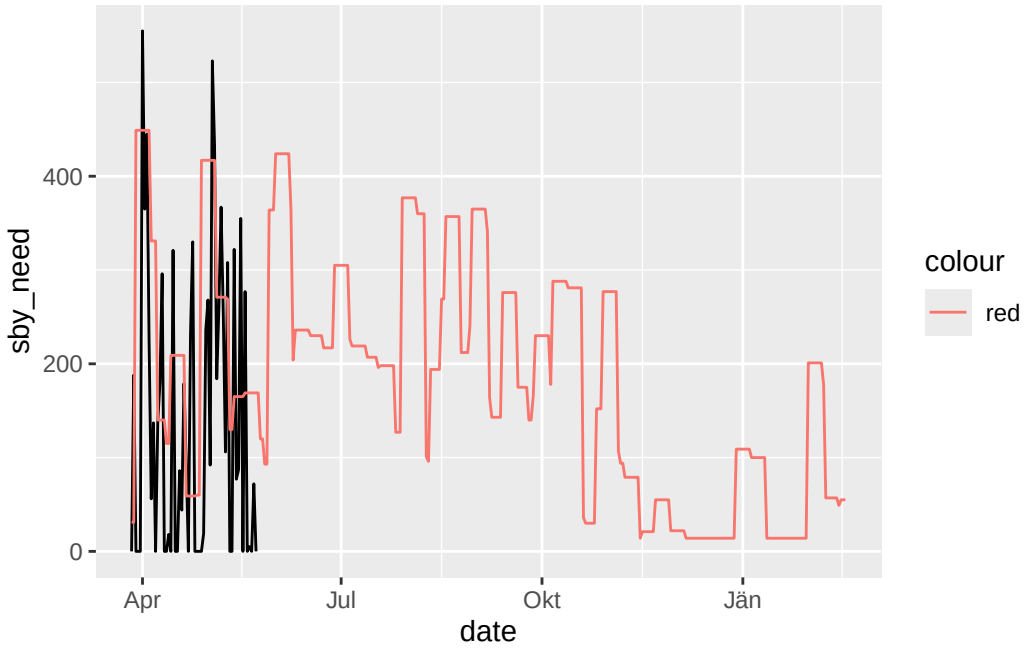
Der für das lineare Modell benötigte Wert für das diensthabende Personal, wird im u.a. Code festgelegt und dem neu gebildeten Datensatz mitgegeben. Diese Vorgehensweise ermöglicht später in der praktischen Anwendung ebenso, vorab das geplante diensthabende Personal für den Vorhersagezeitraum flexibel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

```
# Evaluierungsdatensat - 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 wurde zuw
```

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

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

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

```
print(paste0("Durchschnittswert der mangelhaften negativen Vorhersagen: ", mean(eval[eval$sby_diff < 0,
```

```
[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"
```

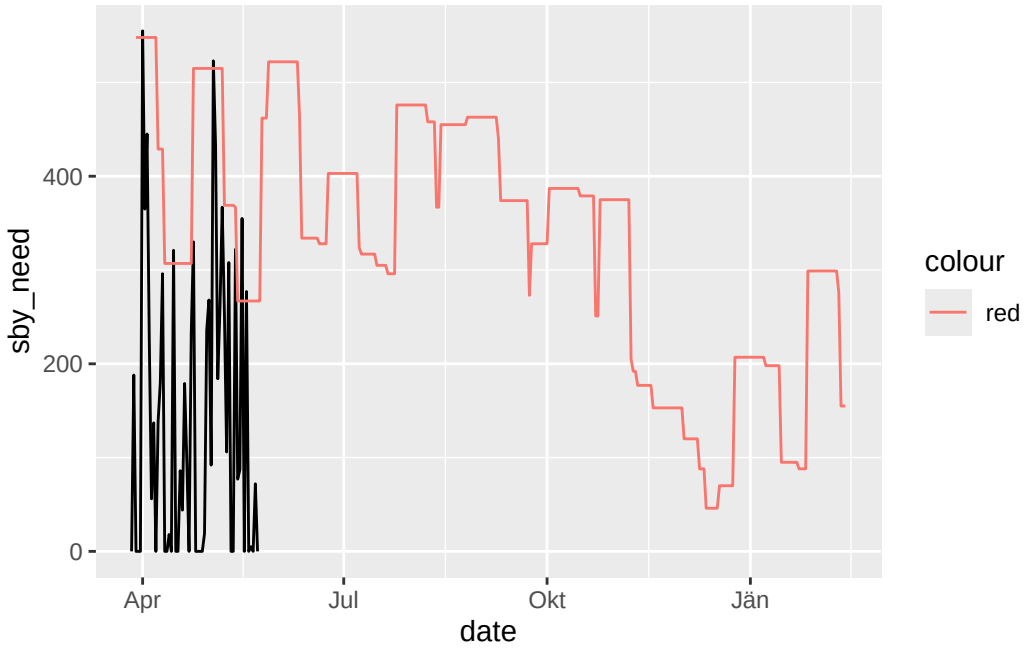
Testhalber wird in der nachgestellten Grafik die Anzahl des diensthabenden Personals um 100 gesenkt, um die Auswirkungen zu beobachten. Gleichzeitig wird der Zeitraum für das gleitende Maximum auf zwei Wochen erhöht um noch mehr Generalisierung zu erreichen.

```
# vorgesehenes 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 == "stl") %>%
  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"),
            aes(x = date, y = sby, colour = "red"))
```



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

```
# Evaluierungsdatensat - 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 wurde zuw
```

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

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

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

```
print(paste0("Durchschnittswert der mangelhaften negativen Vorhersagen: ", mean(eval[eval$sby_diff < 0,
```

```
[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 allgmeine Modellgüte ausdrücken wurden durch die Anpassung verschlechtert. Profitiert hat hingegen die Anforderung ausreichend Bereitschaftspersonal vorherzusagen. Für eine entgültige 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 ensprechen 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 Modellverzeichnisses als **sby_model_advanced.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.