Vignette ifwtrends

Contents


```
library(knitr)
library(tidyverse)
library(tsbox)
library(gtrendsR)
library(trendecon)
library(glmnet)
library(lubridate)
library(zoo)
library(stringr)
```

1 Einordnung

Es gibt eine große Literatur, welche Google Trends Daten in unterschiedlichen Modellen benutzt, um ökonomische Variablen zu prognostizieren. Einen ausführlichen Überblick gibt hierzu der Bericht *Big Data*

in der makroökonomischen Analyse (Kieler Beiträge zur Wirtschaftspolitik Nr. 32), Abschnitt 2.3.4. Zusammenfassend lässt sich sagen, dass die zusätzliche Verwendung von Google Trends Daten dort, wo schon andere Indikatoren vorliegen, zu keiner relevanten Verbesserung der Prognosen führt. Prognosen, welche nur auf Google Daten beruhen, schneiden aber meist ähnlich gut ab wie Prognosen, die lediglich klassische Indikatoren benutzen. Die Stärke der Google Trends Daten liegt vor allem in ihrer hohen regionalen und zeitlichen Verfügbarkeit. Dies führt unserer Ansicht nach zu drei zentralen Anwendungsfeldern für Google Daten in der Prognose:

1. Als Prognosevariablen, wenn klassische Variablen noch nicht vorliegen.
2. Als Prognosevariablen für Länder, für die sonst wenige Indikatoren vorliegen.
3. Als Echtzeitindikatoren für die wirtschaftliche Aktivität in Krisenzeiten, wenn schnell politische Entscheidungen gefällt werden müssen.

2 Google-Trends Daten

2.1 Suchanfragen

Google stellt Zeitreihen der relativen Häufigkeit eines Suchbegriffes in Form von Google Trends zur Verfügung. Hier folgt zunächst eine grundlegende Beschreibung der Daten, um dann deren statistische Besonderheiten näher zu beleuchten.

Die Zeitreihen reichen bis 2004-01-01 zurück und können geografisch eingeschränkt werden, z.B. nach Ländern oder subnationalen Entitäten. Hier sollte beachtet werden, dass (die Suchmaschine) Google in einigen Ländern (insbesondere in autoritär regierten Ländern wie z.B. China) nicht verfügbar ist und für diese Länder deshalb keine Daten vorhanden sind. Da die Nutzung des Internets vor allem seid 2006 stark zunahm folgen wir hier der Literatur und Google Zeitreihen ab 2006.

Bei den möglichen Suchanfragen unterschiedet man zwischen *terms*, *topics* und *categories*. Google definiert einen *term* wie folgt (Google 2021a):

"Search terms show matches for all terms in your query, in the language given.

- If you search the term ‘banana,’ results include terms like ‘banana’ or ‘banana sandwich’
- If you specify ‘banana sandwich,’ results include searches for ‘banana sandwich,’ as well as ‘banana for lunch’ and ‘peanut butter sandwich’".

Die Definition eines *topic* ist (ebd.):

"Topics are a group of terms that share the same concept in any language. Topics display below search terms.

If you search the topic ‘London,’ your search includes results for topics such as: * ‘Capital of the UK’ * ‘Londres’, which is ‘London’ in Spanish".

Die Eingabe eines Suchbegriffs als Topic führt also vor allem zu einer Invarianz des Suchbegriffs gegenüber der Landessprache. Die Differenzierung zwischen Topic und Term ist aber lediglich online in der Google Suchmaske möglich. Bei der Nutzung der R-Funktionen zum Herunterladen von Google-Daten sollte der Suchbegriff also immer in der jeweiligen Landessprache eingegeben werden. Wir unterscheiden deshalb im Folgenden nicht weiter zwischen *term* und *topic*, sondern bezeichnen beides als *Suchbegriff*.

Um weiter zu spezifizieren, welche Daten in den Index eingehen sollen, kann bei einer Abfrage zusätzlich eine *category* angegeben werden. Bei Google heißt es hierzu (Google 2021b):

“If you’re using Trends to search for a word that has multiple meanings, you can filter your results to a certain category to get data for the right version of the word. For example, if you search for “jaguar,” you can add a category to indicate if you mean the animal or the car manufacturer."

Des Weiteren kann der Index auch nur für eine Kategorie ohne Angabe eines speziellen Suchbegriffs berechnet werden; dann gehen alle Suchanfragen, welche Google dieser Kategorie zuordnet, in den Index mit ein. Dieses Vorgehen wird im Folgenden häufiger verwendet.

2.2 Statistische Besonderheiten

2.2.1 Berechnung des Index

Der Google Trends Index ist ein relativer Index. Der Index SVI_{ct} gibt den Anteil der Suchanfragen eines Suchbegriffs/Kategorie c zum Zeitpunkt t an der Gesamtzahl der Suchanfragen zu diesem Zeitpunkt t an, normiert mit einer multiplikativen Konstanten C_c welche vom betrachteten Zeiraum abhängt, sodass das Maximum des Index im betrachteten Zeitraum bei 100 liegt (Woloszko 2020):

$$SVI_{ct} = \frac{SV_{ct}}{SVT_t} \cdot C_c$$

Der Wert des Indexes zum Zeitpunk t kann also unterschiedlich sein, je nachdem welches Zeitfenster man herunterlädt. Dies sollte bei der Arbeit mit den Daten immer beachtet werden. Insbesondere kann für frühere Zeiträume nicht der Index bis zu diesem Zeitpunkt abgeschnitten werden, sondern muss immer neu heruntergeladen werden.

###Sample Der Index wird auf Basis eines zufällig gezogenen Samples an Suchanfragen zu einem Zeitpunkt berechnet. Dieses Sample kann sich zwischen zwei Abfragen, insbesondere an zwei verschiedenen Tagen ändern, sodass sich auch die Zeitreihe geringfügig ändern kann. Dies ist vor allem für Suchbegriffe mit einem geringen Anfragevolumen, z.B. in kleinen regionalen Einheiten relevant. Hier sollte dann eine Mehrfachziehung vorgenommen werden um einen selectin bias zu vermeiden. Da wir im folgenden Kategorienen welche durch die Vielzahl der Suchbegriffe die darunter fallen große Volumina haben und als Region Deutschland betrachten kann dieser Fehler hier vernachlässigt werden. Eine tiefergehende Analyse findet sich in Eichenauer et. al (2020).

2.2.2 Allgemeiner Trend

Da die Nutzung von Google und damit auch die Anzahl verschiedener Suchbegriffe stark zugenommen hat, sinkt für jeden einzelnen Suchbegriff der relative Anteil am Gesamtsuchvolumen im Zeitverlauf (Abb. **??**).

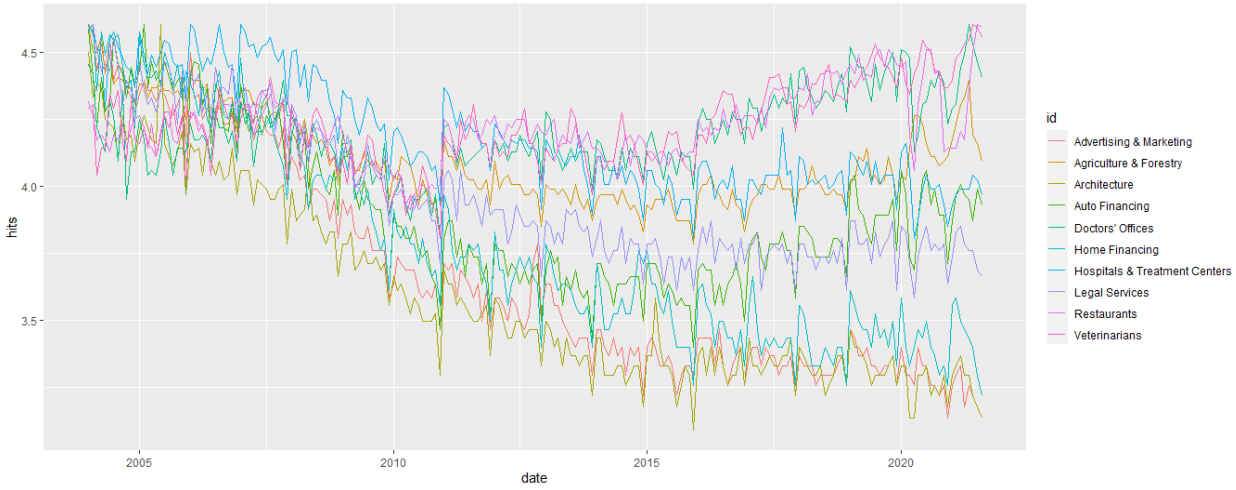


Abb. 1 zeigt die um diesen gemeinsamen Trend bereinigten Reihen.

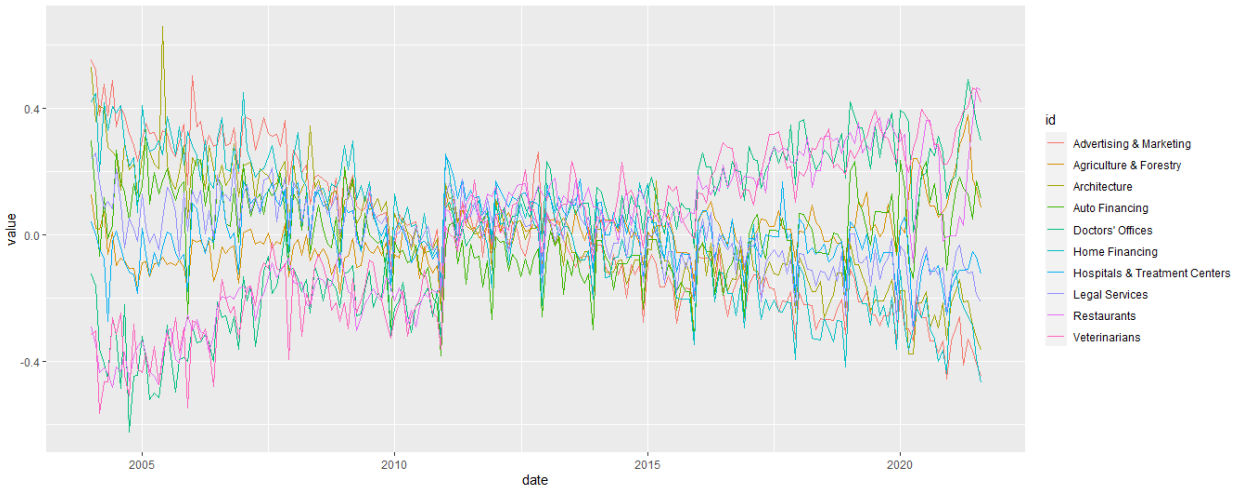


Figure 1: Trendbereinigte Google Kategorien

2.2.3 Frequenz

Die Google Trends Daten können für ein Zeitfenster von 12 Monaten auf Tagesbasis abgefragt werden. Für Zeiträume bis 5 Jahre können wöchentliche Daten heruntergeladen werden. Für alle längeren Zeiträume liegen die Daten nur monatlich vor. Da wir die 12-Monatsfenster der täglichen Reihen und die 5-Jahres-Fenster der wöchentlichen Reihen beliebig wählen können, kann mit der Chow-Lin-Methode für lange Zeiträume eine Reihe auf Tagesbasis erstellt werden, welche konsistent mit den wöchentlichen und monatlichen Reihen ist. Wir folgen dabei Eichenauer et al. (2020). Dies ist in der Funktion `daily_series` (Abschnitt 3.4) implementiert.

2.2.4 Strukturbruch

Im Januar 2011 wurde die regionale Erfassung der Suchanfragen geändert. Dadurch wird in regional eingeschränkten Reihen in 2011 ein Bruch sichtbar (vgl. Abb. **??**. Auch in 2016 wurde die Methode zur Datenerhebung nochmals verändert, was auch einen Strukturbruch in den Reihen zur Folge hat. Eine einfache Methode das Problem zu umgehen ist bei der Betrachtung von Änderungsraten die betreffenden Zeiträume auszulassen. Wir wenden im Folgenden dieses Vorgehen an. Woloszko (2020) ist nach unserem Kenntnisstand das einzige Papier, welches diese Strukturbrüche genauer adressiert.

3 Funktionen

3.1 pca

Die Funktion `pca` nimmt als Argumente mehrere Suchwörter oder Kategorien entgegen. Des Weiteren eine Region, das Start- und Enddatum (Default: 2006-01-01 und heute) sowie die Anzahl der zu berechnenden Hauptkomponenten (Default: Anzahl der Zeitreihen). Für die Zeitreihen wird hier momentan eine monatliche Frequenz angenommen.

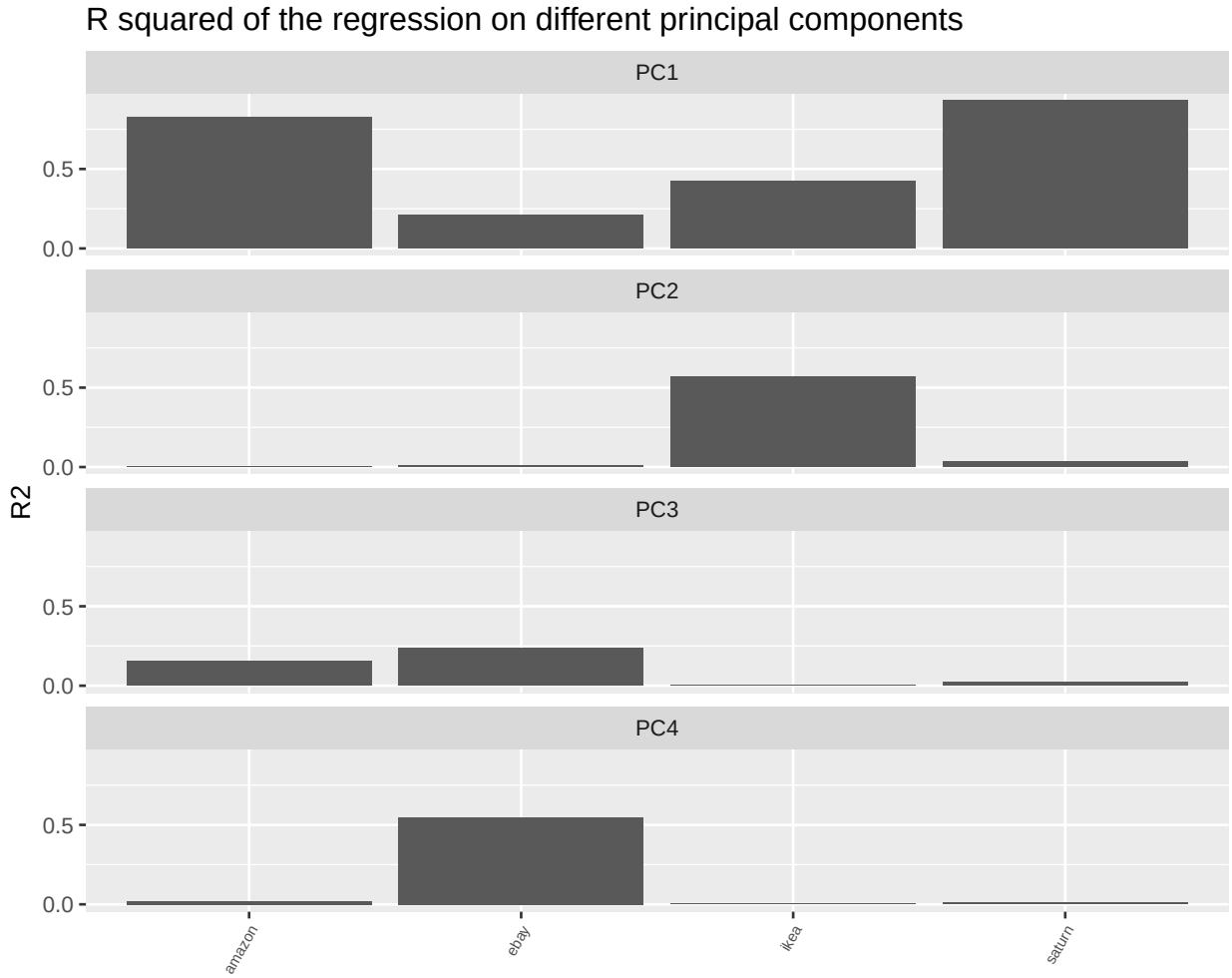
```
pca(keywords = c("ikea", "saturn"),
    categories = 0,
    geo = "DE",
    time = str_c("2018-01-01 ",Sys.Date()))
```

```
#> # A tibble: 46 x 5
#>    date           PC1      PC2  ikea saturn
#>    <date>       <dbl>    <dbl> <int>  <int>
#>  1 2018-01-01   3.96    4.51      81     56
#>  2 2018-02-01  -5.51   -0.708     72     50
#>  3 2018-03-01  -8.67   -2.45      69     48
#>  4 2018-04-01 -12.7    -3.72      66     45
#>  5 2018-05-01 -15.1   -12.6       57     47
#>  6 2018-06-01 -13.8   -12.2       58     48
#>  7 2018-07-01 -13.2    -6.84      63     46
#>  8 2018-08-01  -3.24    1.49      75     51
#>  9 2018-09-01   3.89    0.0340    77     58
#> 10 2018-10-01   0.342   1.88      77     54
#> # ... with 36 more rows
```

3.2 factorR2

Für schon berechnete Faktoren `factors` aus Zeitreihen `series` ist dies eine Methode, die Erklärungskraft der Faktoren zu bestimmen. Dabei wird für jeden Faktor eine Regression auf jede Zeitreihe vorgenommen und das jeweilige R^2 in einer Tabelle abgetragen. Mit Wahl des Parameters `plot=TRUE` wird zusätzlich ein Barplot ausgegeben.

```
dat <- pca(keywords = c("ikea", "saturn", "amazon", "ebay"),
    categories = 0,
    geo = "DE",
    time = str_c("2018-08-01 ", Sys.Date()))
#> [time]: 'date'

series <- dat %>% select(date, 6:9)
factors <- dat %>% select(date, 2:5)

factorR2(series, factors, plot = T)
#> $res
#> # A tibble: 4 x 5
#>   factors    ikea  saturn  amazon    ebay
#>   <chr>     <dbl>   <dbl>   <dbl>   <dbl>
#> 1 PC1     0.424   0.930   0.826   0.208
#> 2 PC2     0.567   0.0357  0.00157 0.00748
#> 3 PC3     0.00271 0.0246  0.152   0.236
#> 4 PC4     0.00615 0.00945 0.0199  0.548
#>
#> $plot
```



3.3 roll

Die Funktion `roll` dient der Erstellung von Vintages für Prognoseevaluationen. Dabei wird für jedes Datum in `start_period` bis `end` die Reihe von `start_series` als Anfang bis zu diesem Datum als Endpunkt neu heruntergeladen. Dies ist notwendig, da zu zwei Download-Zeitpunkten die ganze Google Trends Reihe verschieden sein kann. Als Funktion können Download-Funktion mit den Argumenten `keyword`, `category`, `geo`, `time` und weiteren Argumenten `...` und einem Tibble mit den Spalten `time`, `id` und `value` (Spaltennamen können auch abweichen) benutzt werden. Zurückgegeben wird eine Liste mit einer jeweils um eine Zeiteinheit längeren Zeitreihe. Diese kann dann z.B. mit `lapply(roll(...), f)` zur Prognoseevaluation benutzt werden, wobei `f` eine Prognose-Funktion ist.

```
roll(keyword = c("ikea", "saturn"),
     geo = "DE",
     start_series = "2011-01-01",
     start_period = "2018-05-01",
     end = "2018-12-01",
     fun = ts_gtrends)
#> [[1]]
#> # A tibble: 178 x 3
#>    id    time       value
#>    <chr> <date>     <int>
```

```
#>  1 ikea  2011-01-01    53
#>  2 ikea  2011-02-01    50
#>  3 ikea  2011-03-01    45
#>  4 ikea  2011-04-01    44
#>  5 ikea  2011-05-01    44
#>  6 ikea  2011-06-01    55
#>  7 ikea  2011-07-01    56
#>  8 ikea  2011-08-01    64
#>  9 ikea  2011-09-01    58
#> 10 ikea  2011-10-01    62
#> # ... with 168 more rows
#>
#> [[2]]
#> # A tibble: 180 x 3
#>    id    time       value
#>    <chr> <date>     <int>
#>  1 ikea  2011-01-01    53
#>  2 ikea  2011-02-01    50
#>  3 ikea  2011-03-01    45
#>  4 ikea  2011-04-01    43
#>  5 ikea  2011-05-01    44
#>  6 ikea  2011-06-01    54
#>  7 ikea  2011-07-01    56
#>  8 ikea  2011-08-01    63
#>  9 ikea  2011-09-01    58
#> 10 ikea  2011-10-01    62
#> # ... with 170 more rows
#>
#> [[3]]
#> # A tibble: 182 x 3
#>    id    time       value
#>    <chr> <date>     <int>
#>  1 ikea  2011-01-01    55
#>  2 ikea  2011-02-01    48
#>  3 ikea  2011-03-01    47
#>  4 ikea  2011-04-01    44
#>  5 ikea  2011-05-01    44
#>  6 ikea  2011-06-01    54
#>  7 ikea  2011-07-01    57
#>  8 ikea  2011-08-01    63
#>  9 ikea  2011-09-01    61
#> 10 ikea  2011-10-01    63
#> # ... with 172 more rows
#>
#> [[4]]
#> # A tibble: 184 x 3
#>    id    time       value
#>    <chr> <date>     <int>
#>  1 ikea  2011-01-01    53
#>  2 ikea  2011-02-01    47
#>  3 ikea  2011-03-01    43
#>  4 ikea  2011-04-01    43
#>  5 ikea  2011-05-01    43
```

```
#>  6 ikea  2011-06-01    53
#>  7 ikea  2011-07-01    56
#>  8 ikea  2011-08-01    63
#>  9 ikea  2011-09-01    59
#> 10 ikea  2011-10-01    62
#> # ... with 174 more rows
#>
#> [[5]]
#> # A tibble: 186 x 3
#>    id    time       value
#>    <chr> <date>     <int>
#>  1 ikea  2011-01-01    54
#>  2 ikea  2011-02-01    48
#>  3 ikea  2011-03-01    45
#>  4 ikea  2011-04-01    42
#>  5 ikea  2011-05-01    43
#>  6 ikea  2011-06-01    52
#>  7 ikea  2011-07-01    57
#>  8 ikea  2011-08-01    65
#>  9 ikea  2011-09-01    60
#> 10 ikea  2011-10-01    61
#> # ... with 176 more rows
#>
#> [[6]]
#> # A tibble: 188 x 3
#>    id    time       value
#>    <chr> <date>     <int>
#>  1 ikea  2011-01-01    54
#>  2 ikea  2011-02-01    48
#>  3 ikea  2011-03-01    44
#>  4 ikea  2011-04-01    43
#>  5 ikea  2011-05-01    43
#>  6 ikea  2011-06-01    54
#>  7 ikea  2011-07-01    56
#>  8 ikea  2011-08-01    64
#>  9 ikea  2011-09-01    59
#> 10 ikea  2011-10-01    64
#> # ... with 178 more rows
#>
#> [[7]]
#> # A tibble: 190 x 3
#>    id    time       value
#>    <chr> <date>     <int>
#>  1 ikea  2011-01-01    54
#>  2 ikea  2011-02-01    48
#>  3 ikea  2011-03-01    46
#>  4 ikea  2011-04-01    43
#>  5 ikea  2011-05-01    43
#>  6 ikea  2011-06-01    53
#>  7 ikea  2011-07-01    57
#>  8 ikea  2011-08-01    62
#>  9 ikea  2011-09-01    58
#> 10 ikea  2011-10-01    61
```

```
#> # ... with 180 more rows
#>
#> [[8]]
#> # A tibble: 192 x 3
#>    id    time       value
#>    <chr> <date>     <int>
#>  1 ikea  2011-01-01    54
#>  2 ikea  2011-02-01    47
#>  3 ikea  2011-03-01    45
#>  4 ikea  2011-04-01    43
#>  5 ikea  2011-05-01    44
#>  6 ikea  2011-06-01    53
#>  7 ikea  2011-07-01    58
#>  8 ikea  2011-08-01    64
#>  9 ikea  2011-09-01    60
#> 10 ikea  2011-10-01    62
#> # ... with 182 more rows
```

3.4 daily_series

Für lange Zeitfenster liegen keine täglichen Daten vor, sondern nur monatliche. Die Funktion `daily_series` zieht zunächst für rollierende Zeiträume mehrere Stichproben und schätzt daraus dann mit der Chow-Lin-Methode für den ganzen Zeitraum tägliche Daten. Diese sind konsistent mit den Monatsdaten. Da momentan sehr viele Samples gezogen werden, verursacht die Funktion viele Suchanfragen bei Google, was nach einigen Malen zur vorübergehenden Sperrung der IP führt. Die Anzahl der gezogenen Fenster ist momentan unter der Anzahl aus dem Originalcode, um Anfragen zu sparen. Die dadurch hervorgerufene Abweichung scheint im Moment sehr gering bis 0 zu sein. Eine genaue Evaluation konnte jedoch wegen dem noch nicht gelösten IP-Problem noch nicht durchgeführt werden; dies steht also noch aus.

```
daily_series(keyword = c("arbeitslos"),
             geo = "DE",
             from = "2021-06-01")
#> # A tibble: 133 x 2
#>    time       value
#>    <date>     <dbl>
#>  1 2021-06-01  71.0
#>  2 2021-06-02  43.0
#>  3 2021-06-03  52.0
#>  4 2021-06-04  35.0
#>  5 2021-06-05  19.0
#>  6 2021-06-06  48.0
#>  7 2021-06-07  43.0
#>  8 2021-06-08  33.0
#>  9 2021-06-09  66.0
#> 10 2021-06-10  42.0
#> # ... with 123 more rows
```

3.5 forecast_m und forecast_q

Mit der Funktion `forecast_m(r, dat, fd)` (bzw. `forecast_q(r, dat, fd)`) kann eine Prognoseevaluation mit einer montlichen (bzw. quartalsweisen) Zielvariable durchgeführt werden. `r` ist eine Liste mit Vintages, welche mit `roll` erstellt wurde und die Google Trends Daten enthält, `dat` ist die Zielvariable. Das

logische Argument fd gibt an ob von den Google Daten erste Differenzen betrachtet werden sollen. Werden als `r` Google Daten bis zu einem Zeitpunkt für welchen die Zielvariable `dat` noch nicht vorliegt eingegeben, wird für diesen Zeitpunkt eine Prognose erstellt der Zielvariable erstellt. Das benutzte Modell `last_model` wird auch zurückgegeben. Im Folgenden wird die quartalsweise prozentuale Veränderung der Dienstleistungsimporte mit Google Trends Daten vorhergesagt. Es werden dazu Kategorien welche mit Reisen zu tun haben verwendet. Zusätzlich zu den kontemporären Zeitreihen werden auch die bis zu zwei gelaggten Reihen als Kovariaten verwendet. Die Daten wurden schon vorher mit

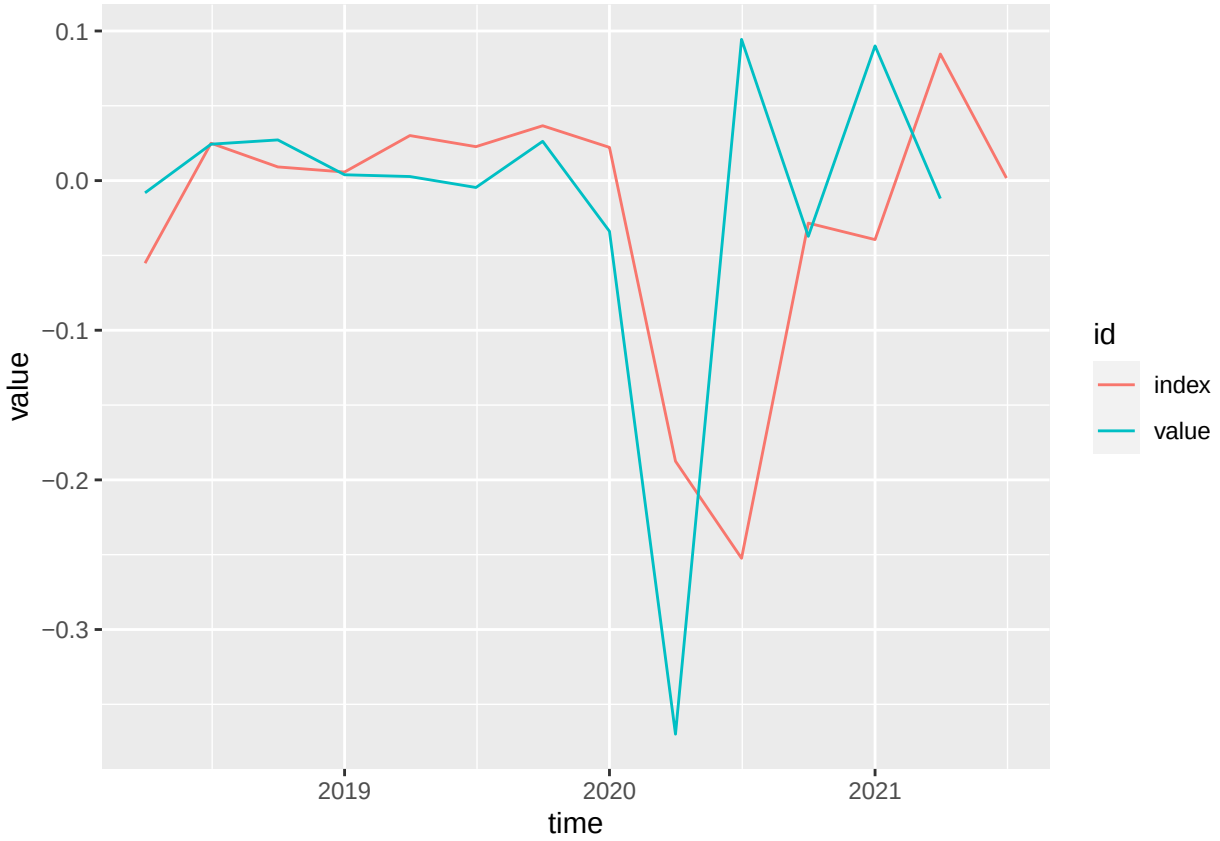
```
r_list = roll(keyword = NA,
              category = c(203, 206, 179, 1003, 1004, 208, 1010, 1011),
              start_series = "2006-01-01",
              start_period = "2018-01-01",
              end = Sys.Date(),
              fun = g_index,
              lags = 2)`
```

heruntergeladen. Bei vielen Suchanfragen empfiehlt es sich aufgrund der IP-Beschränkung in disem SChritt den Evaluationszeitraum aufzuteilen und an zwei Tagen herunterzuladen und dann zu einer Liste zusammenzufügen.

Hier muss darauf geachtet werden, dass das Anfangsdatum der Zeilvariable und der Vintages jeweils übereinstimmt. Auch darf die Zeitreihe keine führenden `NA`s enhalten und am Ende darf höchstens ein `NA` sein. Dies ist häufig der Grund für Fehlermeldungen.

```
filter(gtrendsR::categories, id %in% c(203, 206, 179, 1003, 1004, 208, 1010,
                                       1011))
#>                             name   id
#> 1                    Air Travel  203
#> 2           Cruises & Charters  206
#> 3      Hotels & Accommodations  179
#> 4 Luggage & Travel Accessories 1003
#> 5             Specialty Travel 1004
#> 6          Tourist Destinations  208
#> 7   Travel Agencies & Services 1010
#> 8  Travel Guides & Travelogues 1011

r_list <- readRDS("travel.rds")
r_list <- r_list[1:45]

imports <- readxl::read_xlsx("service_imports.xlsx") %>%
  transmute(time = floor_date(as.Date(Name), "quarter"),
            value = as.numeric(`BD IMPORTS - SERVICES CONA`))
#> Warning in mask$eval_all_mutate(quo): NAs durch Umwandlung
#> erzeugt


dat <- imports %>%
  mutate(value = c(0, diff(log(value),1)) )

rmse <- function(x) sqrt(sum((x[[2]] - x[[3]])^2)/length(x[[2]])) #Root mean squared Error

forec <- forecast_q(r_list, dat, fd = T)$forec%>%
  left_join(dat, by = "time")
```

```
print(forec, n = 13)
#> # A tibble: 14 x 3
#>    time          index    value
#>    <date>        <dbl>    <dbl>
#>  1 2018-04-01 -0.0552  -0.00815
#>  2 2018-07-01  0.0250   0.0244
#>  3 2018-10-01  0.00915  0.0272
#>  4 2019-01-01  0.00575  0.00390
#>  5 2019-04-01  0.0301   0.00272
#>  6 2019-07-01  0.0227  -0.00459
#>  7 2019-10-01  0.0367   0.0263
#>  8 2020-01-01  0.0222  -0.0339
#>  9 2020-04-01 -0.188   -0.370
#> 10 2020-07-01 -0.252    0.0944
#> 11 2020-10-01 -0.0284  -0.0372
#> 12 2021-01-01 -0.0394   0.0900
#> 13 2021-04-01  0.0846  -0.0120
#> # ... with 1 more row
print(rmse(drop_na(forec)))
#> [1] 0.1199347
forec %>%
  pivot_longer(cols = -time, names_to = "id", values_to = "value") %>%
  ggplot(aes(x=  time, y = value, color = id)) +
  geom_line()
#> Warning: Removed 1 row(s) containing missing values
#> (geom_path).
```

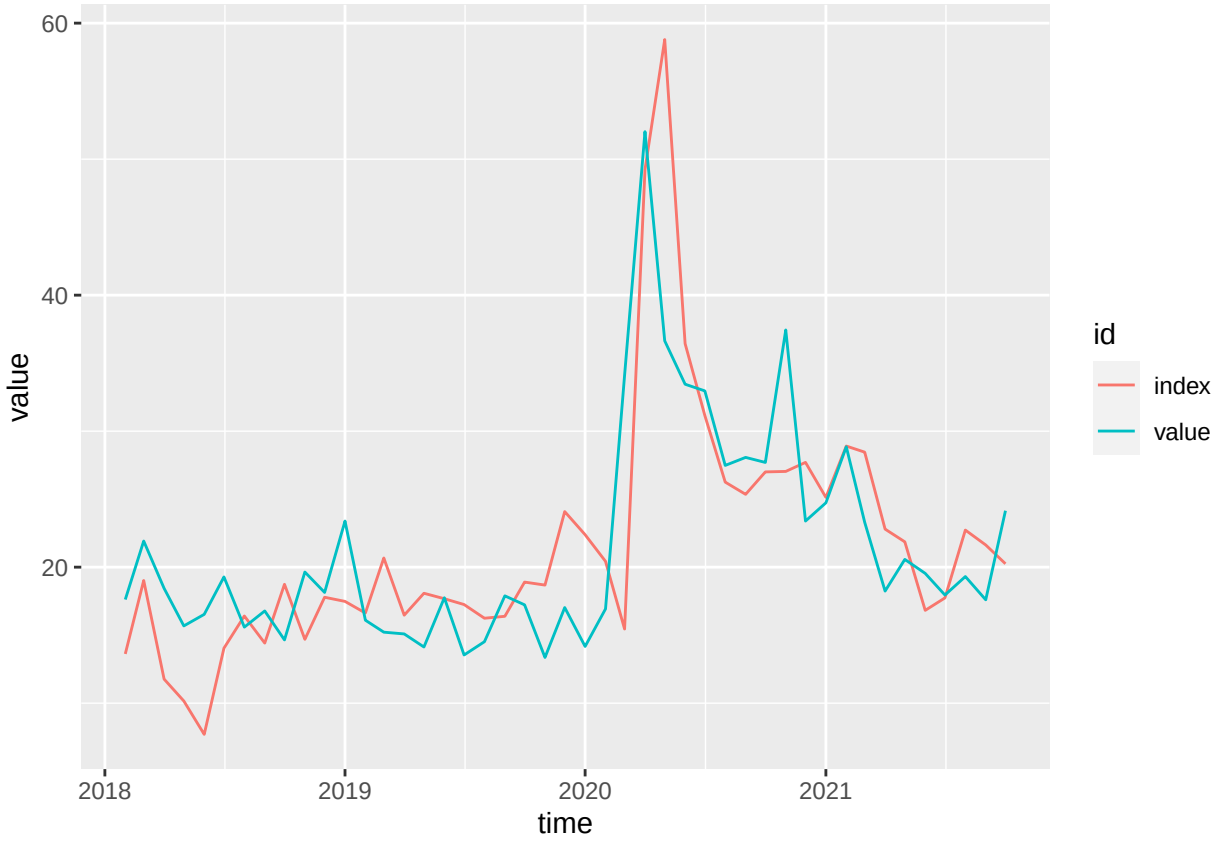


Die blaue Linie ist die Zielvariable. Die rote Linie ist jeweils die Ein-Schritt-Prognose, gegeben das Modell aus dem vorherigen Zeitraum und die Google Daten des aktuellen Zeitraums.

Hier wird nun eine monatliche Prognose des VDAX erstellt. Als Kovariaten dienen Google Reihen für die Suchbegriffe `"arbeitslos","angst","crash","hartz 4","krise","grundsicherung","kündigung","entlassung"`. Da der VDAX monatlich vorliegt, verwenden wir `forecast_m`. Auch hier werden wieder zwei lags betrachtet.

```
r_list <- readRDS("anxiety.rds")

vdax<- readxl::read_xlsx("vdax.xlsx") %>%
  transmute(time = floor_date(as.Date(Name), "month"), value = as.numeric(`VDAX-NEW VOLATILITY INDEX - I


dat <- vdax %>%
  filter(time >= min(first(r_list)$time))



forec <- forecast_m(r_list, dat, fd = F)$forec %>%
  left_join(dat, by = "time")

print(forec, n = 13)
#> # A tibble: 45 x 3
#>    time       index value
#>    <date>     <dbl> <dbl>
#>  1 2018-02-01 13.6   17.6
#>  2 2018-03-01 19.0   21.9
```

```
#>  3 2018-04-01 11.8   18.4
#>  4 2018-05-01 10.2   15.7
#>  5 2018-06-01  7.71  16.5
#>  6 2018-07-01 14.0   19.3
#>  7 2018-08-01 16.4   15.6
#>  8 2018-09-01 14.4   16.8
#>  9 2018-10-01 18.7   14.6
#> 10 2018-11-01 14.7   19.6
#> 11 2018-12-01 17.8   18.1
#> 12 2019-01-01 17.5   23.4
#> 13 2019-02-01 16.6   16.1
#> # ... with 32 more rows
print(rmse(drop_na(forec)))
#> [1] 5.980008
forec %>%
  pivot_longer(cols = -time, names_to = "id", values_to = "value") %>%
  ggplot(aes(x=  time, y = value, color = id)) +
  geom_line()
```



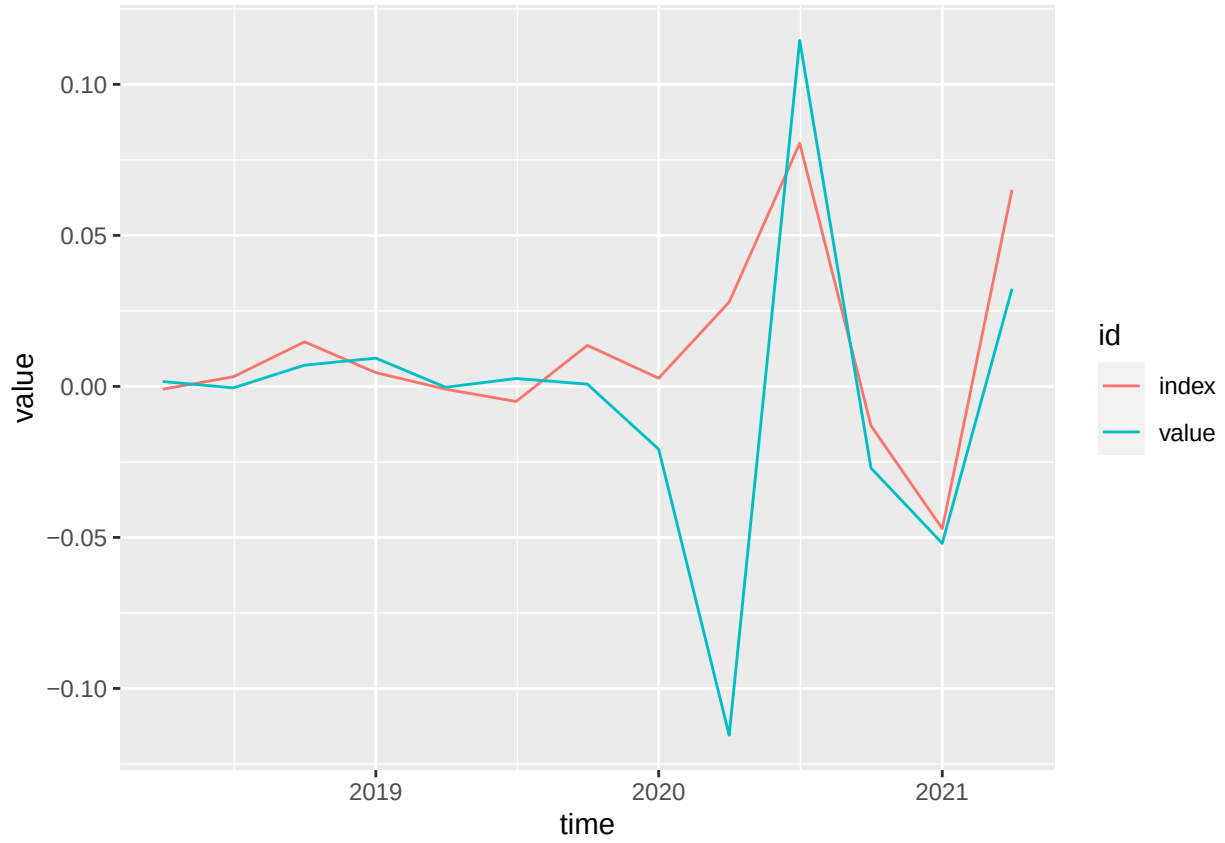
4 Studie: Prognose der privaten Konsumausgaben mit Google Trends

Hier wollen wir die monatliche Veränderung der privaten Konsumausgaben (saisonbereinigt) für Deutschland mithilfe von Google Trends Daten vorhersagen. Als Ausgangspunkt benutzen wir die Google Kategorien, welches das RWI für eine ähnliche Prognose anhand der VGR ausgewählt hat (https://www.rwi-essen.de/konsumindikator). Diese Reihen werden saisonbereinigt, log-transformiert und auf Quartale aggregiert. Davon betrachten wir dann die erste Differenz. Es werden auch hier zusätzlich zwei lags betrachtet. Es wird dann eine RIDGE-Regression der Google-Reihen auf die Zielvariable geschätzt. Dabei gehen jeweils der kontemporäre sowie der um eins und der um zwei gelaggte Datenpunkt der Google-Reihen in das Modell ein. Mit dem so geschätzten Modell wird dann mit den Google Daten des nächsten Quartals eine Prognose für die Zielvariable berechnet. **??** zeigt für jedes Quartal die Konsumausgaben und die Prognose. Aufgrund der großen Zahl von Abfragen wurden hier für den Evaluationszeitraum zwei Teillisten mit Vintages heruntergeladen und anschließend zusammengefügt.

```
filter(gtrendsR::categories, id %in% c(560, 121,
                 277, 123,
                 988, 68,
                 660, 658, 466, 465, 659, 948,
                 270, 271, 137, 158,
                 646, 249, 256,
                 468, 898, 473, 815, 289,
                 382, 383,
                 355, 41, 439, 3, 1010, 432, 882, 614, 78, 408,
                 74,
                 179, 276,
                 7, 143, 146, 508, 38))
#>                            name   id
#> 1           Arts & Entertainment    3
#> 2                   Ticket Sales  614
#> 3    Photographic & Digital Arts  439
#> 4                  Vehicle Brands  815
#> 5                   Vauxhall-Opel  898
#> 6                Vehicle Shopping  473
#> 7                Face & Body Care  143
#> 8                       Hair Care  146
#> 9                 Book Retailers  355
#> 10                   Electricity  658
#> 11                     Oil & Gas  659
#> 12              Waste Management  660
#> 13      Grocery & Food Retailers  121
#> 14            Freight & Trucking  289
#> 15          Consumer Electronics   78
#> 16                       Finance    7
#> 17                Auto Financing  468
#> 18                Home Financing  466
#> 19                     Insurance   38
#> 20              Health Insurance  249
#> 21                Home Insurance  465
#> 22           Alcoholic Beverages  277
#> 23      Grocery & Food Retailers  121
#> 24       Non-Alcoholic Beverages  560
#> 25                   Restaurants  276
```

```
#> 26        Computer & Video Games   41
#> 27 Medical Facilities & Services  256
#> 28           Drugs & Medications  646
#> 29    Animal Products & Services  882
#> 30   Photographic & Digital Arts  439
#> 31                     Bed & Bath  948
#> 32               Home Appliances  271
#> 33              Home Furnishings  270
#> 34              Home Improvement  158
#> 35   Homemaking & Interior Decor  137
#> 36             Mobile & Wireless  382
#> 37             Service Providers  383
#> 38                     Education   74
#> 39               Social Services  508
#> 40                     Newspapers  408
#> 41    Animal Products & Services  882
#> 42                       Apparel   68
#> 43                      Costumes  988
#> 44          Consumer Electronics   78
#> 45                Book Retailers  355
#> 46                  Ticket Sales  614
#> 47              Tobacco Products  123
#> 48                          Toys  432
#> 49       Hotels & Accommodations  179
#> 50    Travel Agencies & Services 1010


r1 <- readRDS("consum_gindex_roll_1219")
r2 <- readRDS("consum_gindex_roll_0721")

r <- c(r1, r2) #Auf Gleiche Laenge wie dat Kuerzen

consexp <- readxl::read_xlsx("consumer_exp_GER.xlsx") %>%
  transmute(time = floor_date(as.Date(Name), "quarter"), value = as.numeric(`BD CONSUMER EXPENDITURE CO
#> Warning in mask$eval_all_mutate(quo): NAs durch Umwandlung
#> erzeugt


dat <- consexp %>%
  mutate(value = value/lag(value) -1 )

dat[1,2] <- 0 #Replace NA with 0 at beginning




forec <- forecast_q(r, dat, fd = T)$forec %>%
  left_join(dat, by = "time")

print(forec, n = 13)
#> # A tibble: 13 x 3
#>    time           index     value
#>    <date>         <dbl>     <dbl>
```

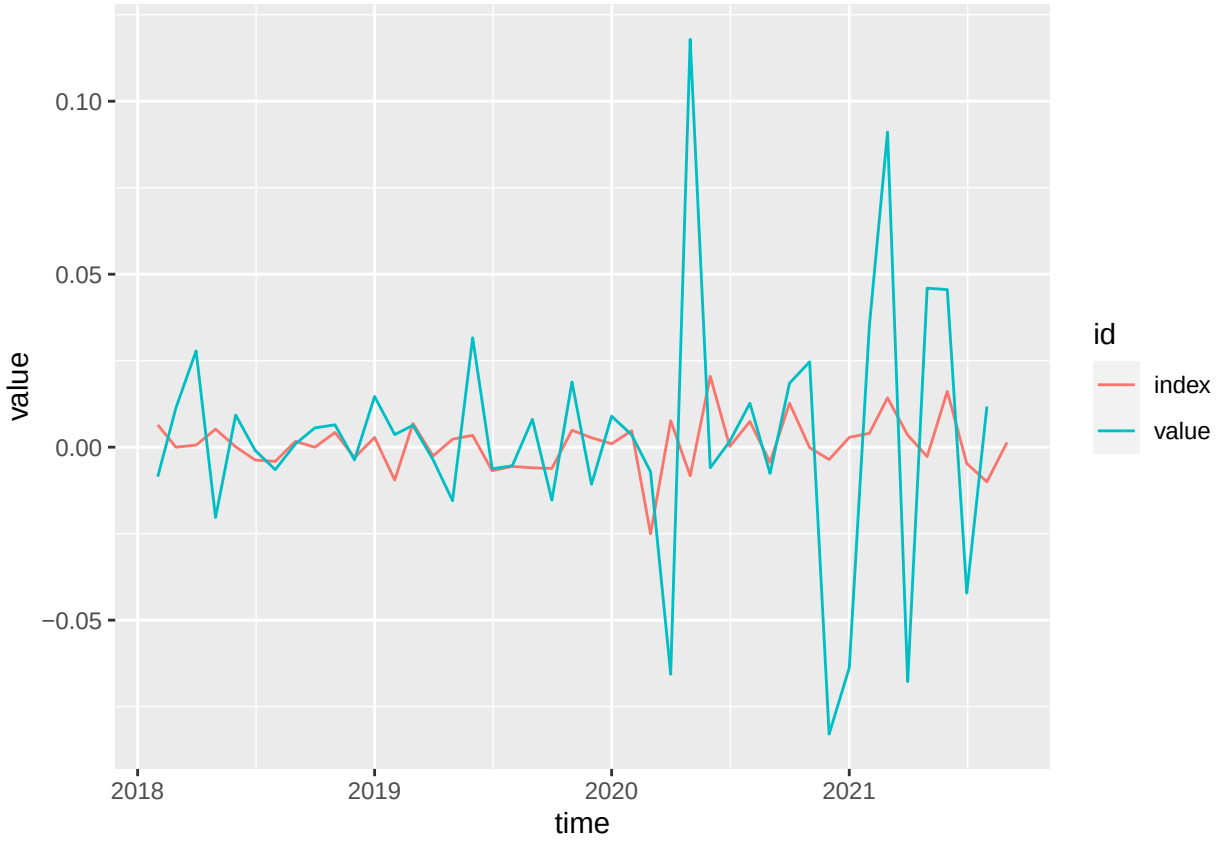
```
#>  1 2018-04-01 -0.000930  0.00162
#>  2 2018-07-01  0.00318  -0.000475
#>  3 2018-10-01  0.0148    0.00703
#>  4 2019-01-01  0.00456   0.00936
#>  5 2019-04-01 -0.000917 -0.000280
#>  6 2019-07-01 -0.00499   0.00262
#>  7 2019-10-01  0.0136    0.000746
#>  8 2020-01-01  0.00271  -0.0208
#>  9 2020-04-01  0.0279   -0.115
#> 10 2020-07-01  0.0805    0.115
#> 11 2020-10-01 -0.0130   -0.0270
#> 12 2021-01-01 -0.0471   -0.0520
#> 13 2021-04-01  0.0651    0.0323
print(rmse(drop_na(forec)))
#> [1] 0.0428797
forec %>%
  pivot_longer(cols = -time, names_to = "id", values_to = "value") %>%
  ggplot(aes(x=  time, y = value, color = id)) +
  geom_line()
```



Studie: Prognose der Einzelhandelsumsätze

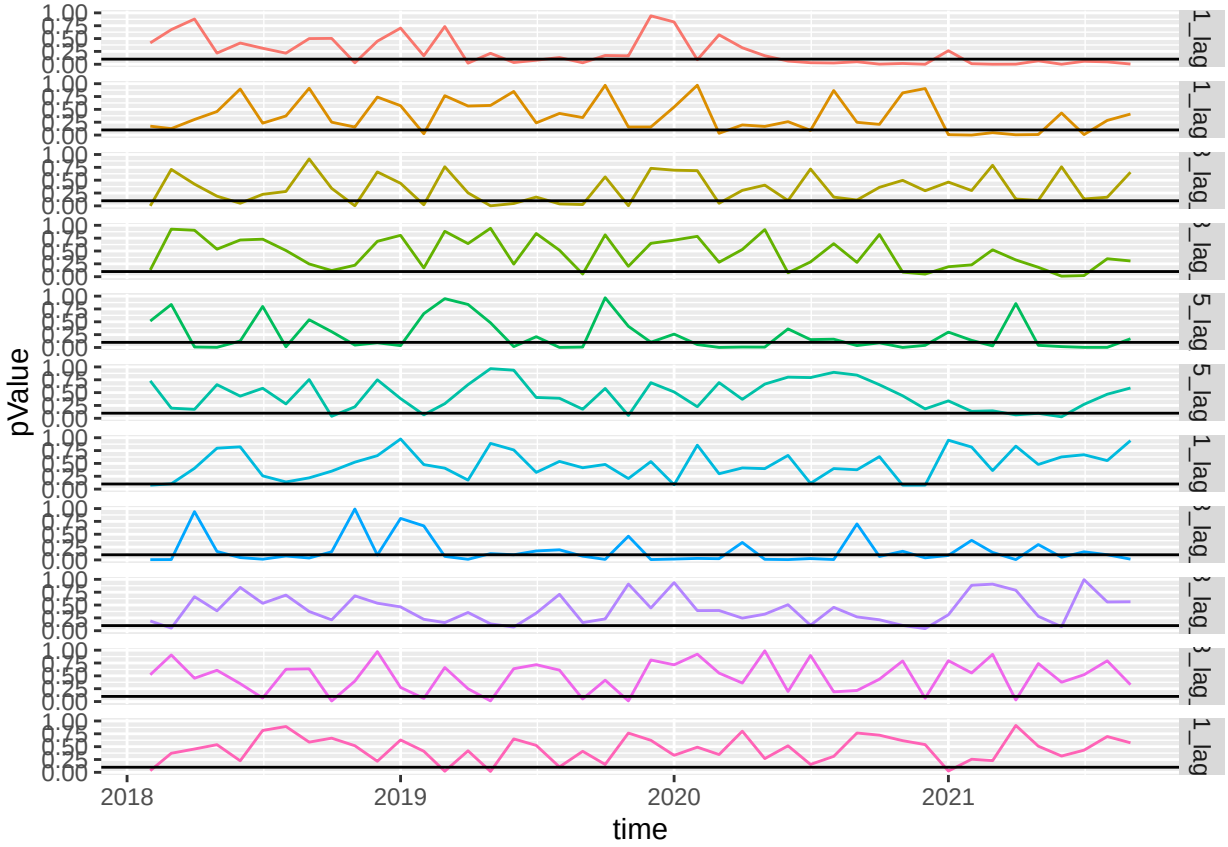
Hier soll mit dem gleichen Verfahren die monatliche Veränderung der Einzelhandelsumsätze für Deutschland prognostiziert werden. Wir verwenden dafür wieder eine Vorauswahl an Kategorien inklusive zweier lags.

```r
filter(gtrendsR::categories, id %in% c(18,78,68,531,355,121,841))
#>                                name  id
#> 1                     Book Retailers 355
#> 2           Grocery & Food Retailers 121
#> 3                       Retail Trade 841
#> 4               Consumer Electronics  78
#> 5           Grocery & Food Retailers 121
#> 6                           Shopping  18
#> 7                            Apparel  68
#> 8               Consumer Electronics  78
#> 9                     Book Retailers 355
#> 10 Shopping Portals & Search Engines 531

r1 <- readRDS("retail_gindex_roll_1219.rds")
r2 <- readRDS("retail_gindex_roll_0921.rds")


r_list <- c(r1,r2)

retail <- readxl::read_xlsx("retail.xlsx") %>%
  transmute(time = floor_date(as.Date(Name), "month"), value = as.numeric(`BD RETAIL SALES EXCL CARS (CA
#> Warning in mask$eval_all_mutate(quo): NAs durch Umwandlung
#> erzeugt


dat <- retail %>%
  mutate(value = c(0, diff(log(value),1)) ) %>%
  filter(time > as.Date("2006-02-01"))




forec <- forecast_m(r_list, dat, fd = T)



rmse(forec$forec %>%
       left_join(dat, by = "time") %>%
       drop_na())
#> [1] 0.03570859

forec$forec %>%
  left_join(dat, by = "time") %>%
  pivot_longer(cols = -time, names_to = "id", values_to = "value") %>%
  ggplot(aes(x=  time, y = value, color = id)) +
  geom_line()
#> Warning: Removed 1 row(s) containing missing values
#> (geom_path).
```



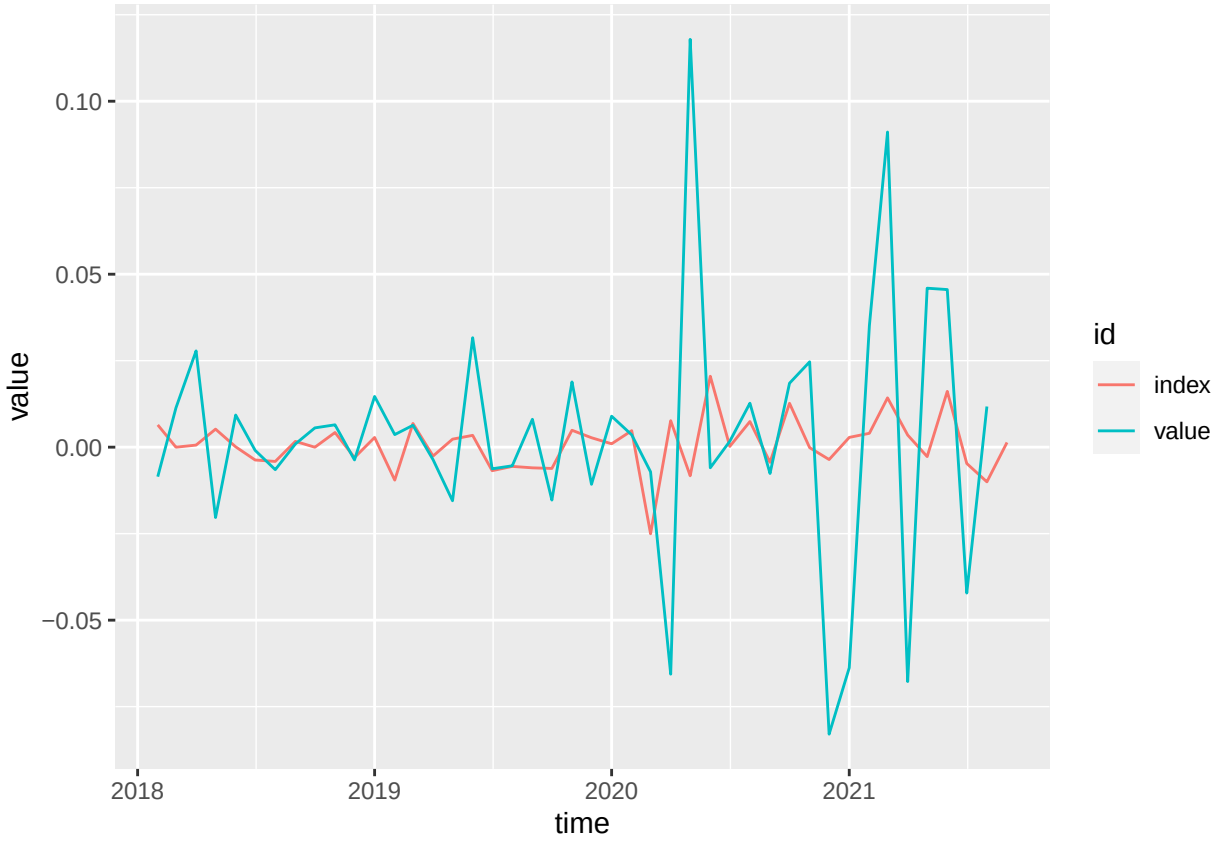
Hier sehen wir die p-Werte der Hälfte der Koeffizienten mit dem niedrigsten mittleren p-Wert. Die horizontale Linie ist das 10% Signifikanzniveau.

```
pvalue <- as_tibble(t(as.data.frame(forec$s_niv))) %>%
  bind_cols(time = forec$forec$time)

pvalue %>%
  pivot_longer(cols = -time, names_to = "coef", values_to = "pValue") %>%
  group_by(coef) %>%
  mutate(mean = mean(pValue)) %>%
  ungroup() %>%
  filter(mean <= quantile(mean, .5)) %>%
  ggplot(aes(x = time, y = pValue, color = coef)) +
  geom_line() +
  geom_hline(yintercept = 0.1) +
  theme(legend.position = "none") +
  facet_grid(coef~.)
```

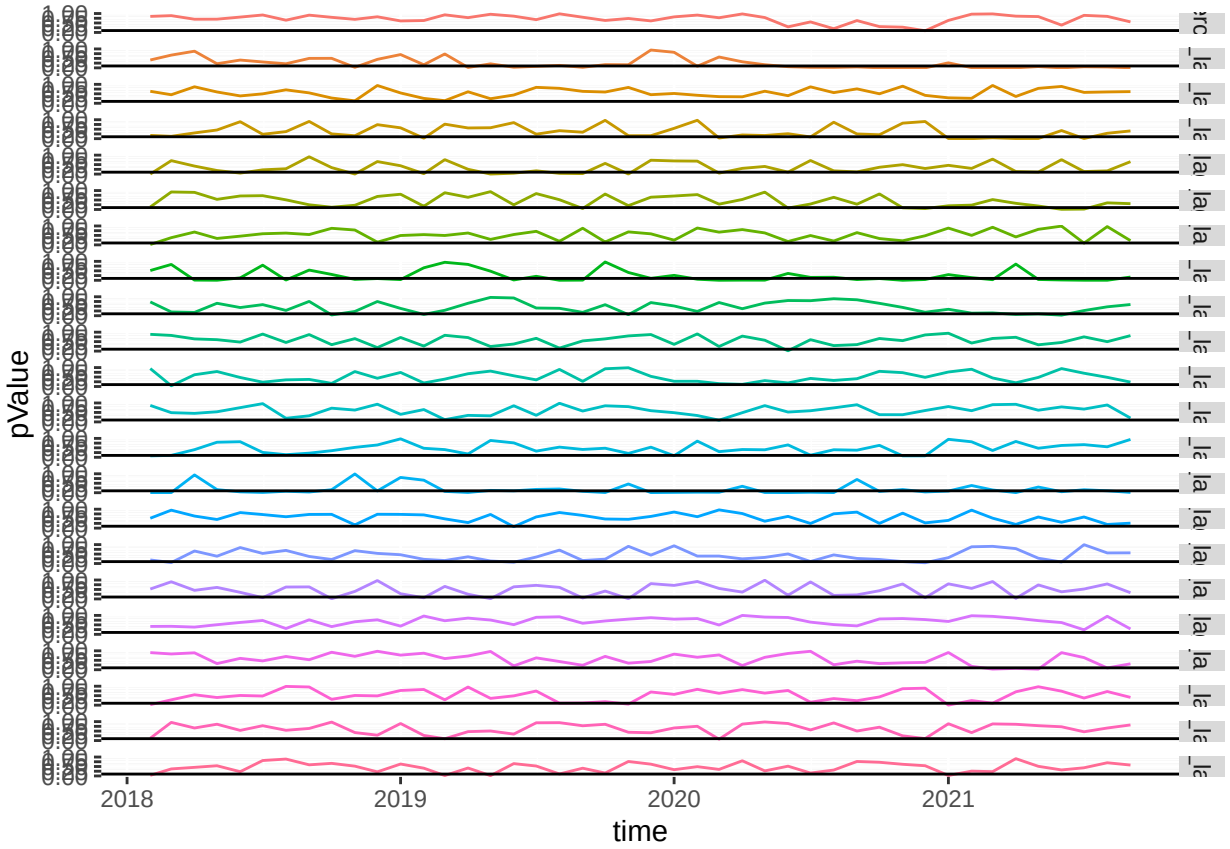


Man sieht deutlich, dass die Koeffizienten der Reihen `121_lag_0`, `355_lag_0` und `68_lag_0` die einzigen sind, welche über längere Zeit eine hohe Signifikanz aufweisen. Betrachten wir also nun nochmals eine SChätzung in welche nur diese drei Reihen eingehen.

```
r_list2 <- lapply(r_list, function(x) x %>%
                    select(time, `121_lag_0`, `355_lag_0`, `68_lag_0`))

forec2 <- forecast_m(r_list, dat, fd = T)

rmse(forec$forec %>%
       left_join(dat, by = "time") %>%
       drop_na())
#> [1] 0.03570859


forec2$forec %>%
  left_join(dat, by = "time") %>%
  pivot_longer(cols = -time, names_to = "id", values_to = "value") %>%
  ggplot(aes(x=  time, y = value, color = id)) +
  geom_line()
#> Warning: Removed 1 row(s) containing missing values
#> (geom_path).
```




```
pvalue <- as_tibble(t(as.data.frame(forec$s_niv))) %>%
  bind_cols(time = forec$forec$time)

pvalue %>%
  pivot_longer(cols = -time, names_to = "coef", values_to = "pValue") %>%
  group_by(coef) %>%
  mutate(mean = mean(pValue)) %>%
  ungroup() %>%
  ggplot(aes(x = time, y = pValue, color = coef)) +
  geom_line() +
  geom_hline(yintercept = 0.1) +
  theme(legend.position = "none") +
  facet_grid(coef~.)
```



Ein Vergleich des jeweiligen RMSE zeigt leider keine deutliche Verbesserung.

5 Ausblick

Bisher wurden die Zeitreihen welche in das Modell eingehen immer "von Hand" nach ökonomischer Plausibilität ausgewählt. Dort wo noch keine guten Prognosen erzielt werden konnten, wäre ein nächster Schritt eine rein datengetriebene Auswahl. So könnte das volle Potenzial der Google Daten im Sinne einer Big-Data Analyse ausgenutzt werden. Götz und Knetsch (2019) evaluieren einige dafür geeignete Methoden. In unseren Analysen hat sich gezeigt, dass PCA, RIDGE und LASSO bei einer Vorauswahl "von Hand" immer schlechtere Ergebnisse liefern als eine OLS Schätzung.

6 Fehlermeldungen

Einige Funktionen, besonders `daily_series` und `roll` verursachen viele Download-Anfragen an Google. Problematischerweise sperrt Google die IP für weitere Anfragen nach ca. 1000 Downloads pro Tag. Dies sollte bei der Analyse großer Datensätze beachtet werden und der Download gegebenenfalls auf mehrere Tage verteilt werden.

Sollte beim Benutzen der Fehler

```
Error widget$status_code == 200 is not TRUE
```

auftreten, so bedeutet dies, dass das Paket nicht mehr auf Google Trends Daten zugreifen kann, da die IP des benutzten Rechners gesperrt wurde. Beheben kann man dies natürlich nicht. Die einzigen beiden Alternativen, um weiterarbeiten zu können, sind:

- Mit einem anderen Rechner weiterarbeiten.
- Warten, bis die IP-Sperre aufgehoben wurden ist. I.d.R. hält der Bann etwa einen Tag bzw. bis zum Ende des Tages an.

7 Quellen

Google 2021a: https://support.google.com/trends/answer/4359550?hl=en
Google 2021b: https://support.google.com/trends/answer/4359597?hl=en
Woloszko 2020: Wolozsko, N. (2020): *Tracking activity in real time with Google Trends.* OECD Department working Paper No. 1634.
Eichenauer et. al 2020 *Constructing daily economic sentiment indices based on Google trends.* KOF Working Papers No. 484, 2020.