Vignette ifwtrends

03.12.2021

Inhaltsverzeichnis

1 Voraussetzungen

Zum Ausführen der Code-Blöcke in dieser Vignette brauchen wir folgende Pakete:

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

2 Einordnung

Es gibt eine umfangreiche Literatur, die Google Trends Daten 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 das Benutzen der Daten von Google Trends dort, wo schon andere Frühindikatoren vorliegen, zu keiner systematischen Verbesserung der Prognosen führt. Prognosen, die nur auf Google Trends 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 Trends Daten in der Prognose:

1. Als Frühindikator, wenn wenig andere Frühindikatoren vorliegen (z.B. Dienstleistungen).
2. Als Prognosevariablen für Länder, für die sonst nur wenige Indikatoren vorliegen.
3. Als tagesaktuelle Frühindikatoren für die wirtschaftliche Aktivität in Krisenzeiten, wenn schnell politische Entscheidungen gefällt werden müssen.

Diese Vignette soll im folgenden einen kurzen Überblick über Google Trends Daten und ihre Eigenheiten geben, erklären wie das Paket `ifwtrends` mithilfe funktioniert sowie einige Beispiele zur Effektivität von Google Trends Daten zeigen.

3 Google Trends: Die Basics

3.1 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 zum 1. Januar 2004 zurück und können geografisch eingeschränkt werden, z.B. nach Ländern oder subnationalen Entitäten. Allerdings ist Google als Suchmaschine in einigen Ländern (insbesondere in autoritär regierten Ländern wie z.B. China) nicht verfügbar. Für diese Länder liegen deshalb keine Daten vor. Da die Nutzung des Internets vor allem seit 2006 stark zunahm folgen wir hier der Literatur und verwenden Google Zeitreihen ab 2006.

Es gibt für `R` zwei Pakete mit einer Funktion zum Download von Google-Trends Daten. Im Paket `gtrendsR` gibt es die Funktion `gtrends` welche neben der eigentlichen Zeitreihe noch weitere Daten, wie z.B. die im Index enthaltenen Suchanfragen herunterlädt. Die Funktion `ts_gtrends` des Pakets `trendecon` (nur auf GitHub verfügbar) ist ein Wrapper für die Funktion `gtrends` und gibt lediglich die Zeitreihe als `tibble` zurück.

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. Es gibt 1426 Kategorien, die hierarchisch in Ober- und Unterkategorien unterteilt sind. 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."

Eine Übersicht über alle Google Trends Kategorien gibt es in `gtrendsR`:

```
gtrendsR::categories |> tibble::as_tibble()
#> # A tibble: 1,426 x 2
#>    name                              id
#>    <chr>                             <chr>
#>  1 All categories                    0
#>  2 Arts & Entertainment              3
#>  3 Celebrities & Entertainment News  184
#>  4 Comics & Animation                316
#>  5 Animated Films                    1104
#>  6 Anime & Manga                     317
#>  7 Cartoons                          319
#>  8 Comics                            318
#>  9 Entertainment Industry            612
#> 10 Film & TV Industry                1116
#> # ... with 1,416 more rows
```

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 standardmäßig verwendet.

3.2 Statistische Besonderheiten

3.2.1 Berechnung des Index

Der Google Trends Index ist ein relativer Index. Der Index SVI_{ct} gibt den Anteil der Suchanfragen eines Suchbegriffs/ einer 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):

$$SVI_{ct} = \frac{SV_{ct}}{SVT_t} \cdot C_c \tag{1}$$

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.

3.2.2 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 *selection bias* zu vermeiden. Da wir im folgenden Kategorien, 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).

3.2.3 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. 1). Abb. 2 zeigt die um diesen gemeinsamen Trend bereinigten Reihen.

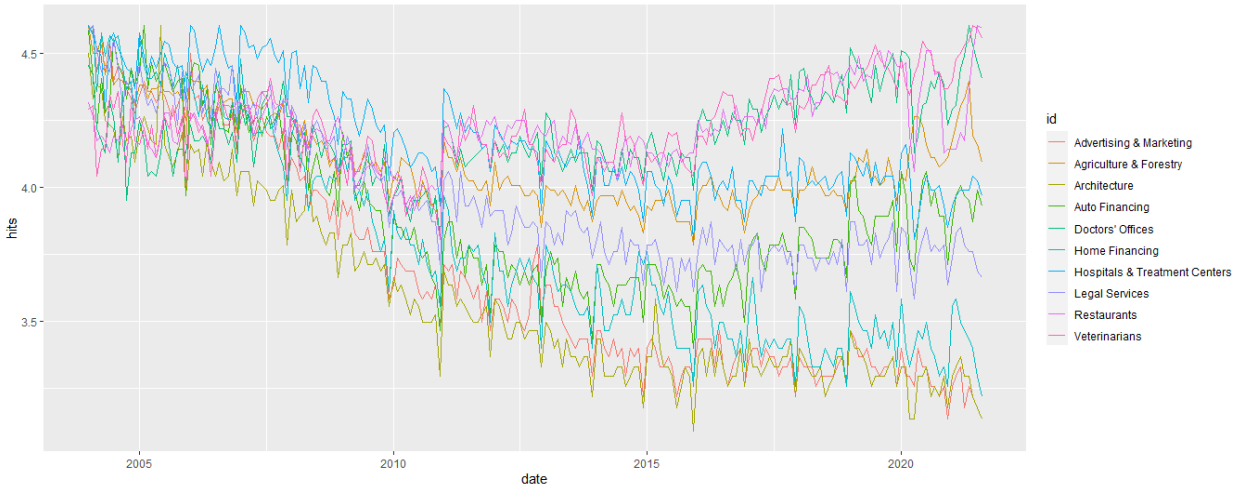


Abbildung 1: Google Trends Kategorien im zeitlichen Verlauf (Zufällige Auswahl)

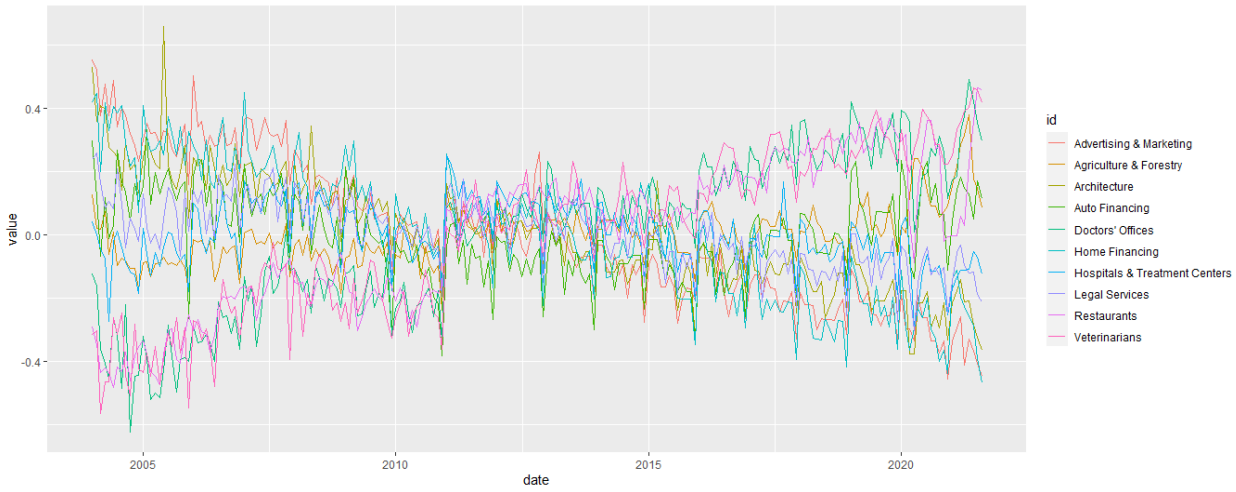


Abbildung 2: Trendbereinigte Google Kategorien

Wir benutzen zur Trendbereinigung die von Woloszko (2020) vorgeschlagene Methode. Zunächst berechnen wir den Logarithmus der Zeitreihe. Gleichung (1) transformiert sich dann zu

$$svi_{ct} = log(SVI_{ct}) = sv_{ct} - svt_t + C_c \tag{2}$$

Damit schätzen wir eine Panel-Regression mit Category Fixed-Effects.

$$svi_{ct} = \alpha_c + P(t)\beta + \varepsilon_{ct} \tag{3}$$

$P(t)$ ist dabei ein Polynom 5. Grades. Setzen wir dann die Gleichungen (2) und (3) gleich, so erhalten wir nach Koeffizientenvergleich $\varepsilon_{ct} = sv_{ct}$. $P(t)\beta$ ist der allgemeine Zeittrend. Für eine Umsetzung siehe den Abschnitt zu est_trend.

3.2.4 Frequenz

Die Google Trends Daten können für ein Zeitfenster von 9 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 9-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 daily_series und simple_daily_series) implementiert. Dadurch kann das Problem der Skalierung des Maximums in jedem Zeitfenster auf 100 umgangen werden.

3.2.5 Strukturbruch

Im Januar 2011 wurde die regionale Erfassung der Suchanfragen geändert. Dadurch wird in regional eingeschränkten Reihen in 2011 ein Bruch sichtbar (vgl. Abb. 1). 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 Verwendung von Änderungsraten die betreffenden Zeiträume auszulassen. Wir wenden im Folgenden dieses Vorgehen an. Woloszko (2020) adressiert diese Strukturbrüche genauer.

4 Funktionen

Für unsere Analysen haben wir einige Funktionen in `R` implementiert. Die Downloads der Daten erfolgen immer mit den oben genannten Funktionen. So konnten bestimmte Abläufe automatisiert werden und die In- und Outputs der einzelnen Funktionen aufeinander abgestimmt werden.

4.1 daily_series und simple_daily_series

Für lange Zeitfenster mit einem Zeitraum von über fünf Jahren liegen keine täglichen Daten vor, sondern nur monatliche. Die Funktion `daily_series()` ist im Wesentlichen eine leicht modifizierte Variante der Funktion `ts_gtrends_mwd()` aus dem Paket `trendecon`, die aber nicht immer funktioniert (Stand: Oktober 2021). Die Funktion 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. Genauere Tests konnten jedoch wegen dem noch nicht gelösten IP-Problem noch nicht durchgeführt werden; dies steht also noch aus.

Alternativ wurde allerdings auch `simple_daily_series()` implementiert, die ebenfalls versucht, tägliche Datensätze für lange Zeitfenster bereitzustellen. Allerdings funktioniert diese Funktion etwas anders, da sie nicht auf die internen Funktionen von dem Paket `trendecon` setzt. Die Ergebnisse beider hier implementierten Funktionen stimmen daher nicht vollständig überein. Der Vorteil letzterer Funktion ist aber, das konstruktionsbedingt kein Wert über 100 liegen kann. Aufgrund der enormen Menge an Anfragen, die beide Funktionen an Google stellen, werden die hier nicht ausgeführt.

```
daily_series(keyword = c("arbeitslos"),
             geo = "DE",
             from = "2021-06-01")
```

```
simple_daily_series(
  keyword = "covid-19",
  geo = "DE",
  from = "2020-04-01",
  verbose = FALSE
)
```

4.2 est_trend

Aufgrund des stetig gewachsenen Internets in den letzten beiden Jahrzehnten verändert sich die relative Suchquote eines jeden Suchbegriffs oder einer Kategorie. Dies stellt einen berechenbaren Trend dar. `est_trend()` berechnet auf Basis von 250 zufällig ausgesuchten Google Kategorien diesen gemeinsamen Trend. Der Nutzer kann diese Funktion selbst nutzen, allerdings wird einmal zu Beginn des Monats dieser gemeinsame Trend berechnet und in diesem Paket innerhalb eines tibbles gespeichert, worauf die anderen Funktionen dieses Pakets zugreifen können. Allerdings muss das Paket dafür neu installiert werden.

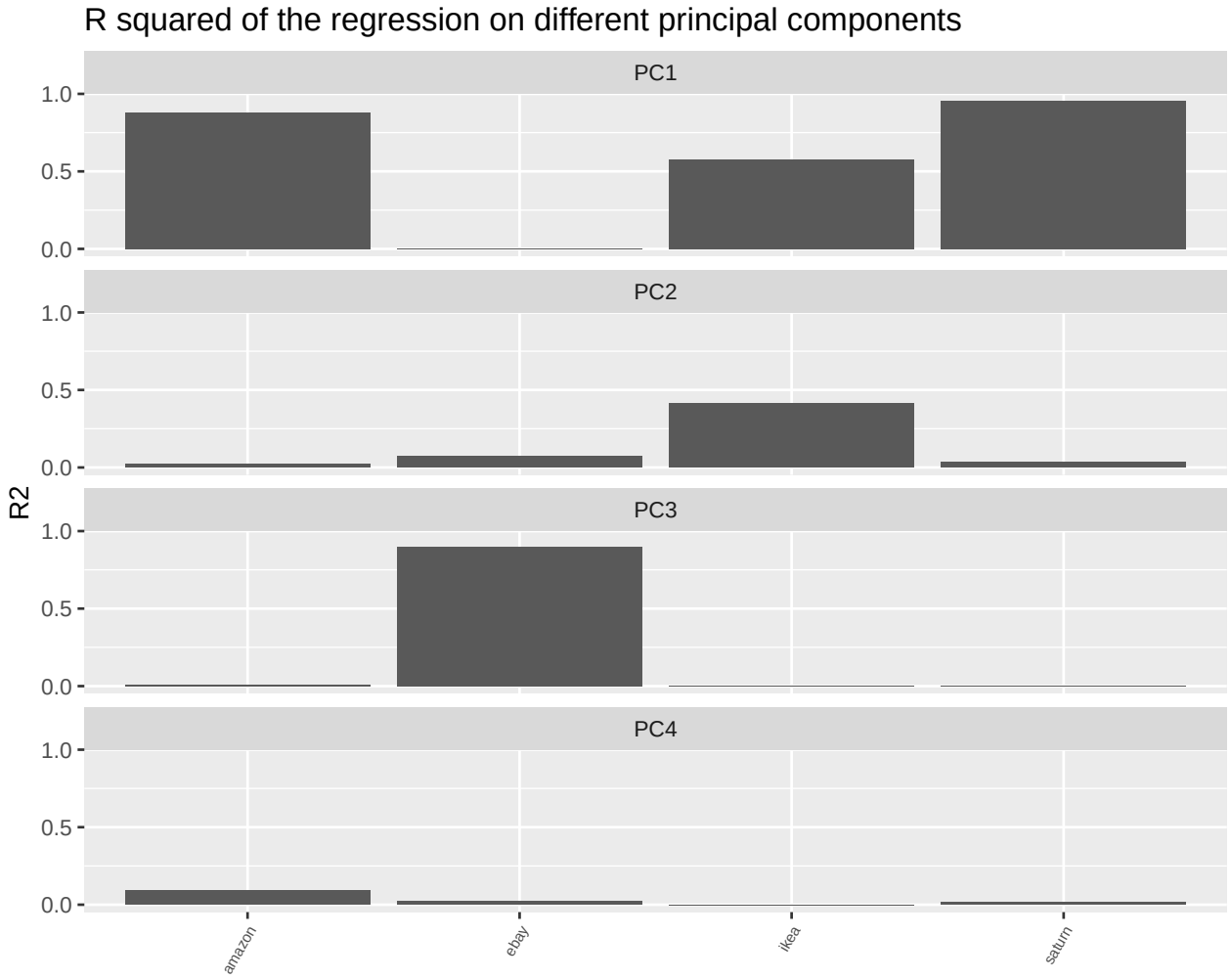
4.3 factorR2

Für schon berechnete Faktoren (`factors`) aus Zeitreihen (`series`) ist dies eine Methode, um 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 = "2018-01-01 2020-01-01")
#> [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     0.580      0.953    0.880   0.00492
#> 2 PC2     0.415      0.0329   0.0218  0.0737
#> 3 PC3     0.00468    0.000328 0.00903 0.896
#> 4 PC4     0.00000533 0.0135   0.0888  0.0250
#>
#> $plot
```



4.4 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. In `forecast_q` werden dabei immer die Google Daten des ganzen Quartals am Ende des Quartals aggregiert. 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 der Zielvariable erstellt. Das benutzte Modell `last_model` wird auch zurückgegeben.

4.4.1 Beispiel: Dienstleistungsimporte

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 den Evaluationszeitraum `start_period` bis `end_period` in diesem Schritt aufzuteilen und an zwei Tagen herunterzuladen, um sie daraufhin in 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. Hier sind die benutzten Kategorien aufgeführt:

```
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
```

Zunächst werden die Daten eingelesen und die Liste so gekürzt, dass die Google Daten für ein Quartal mehr vorliegen als die Zielvariable. Für dieses Quartal muss in der Zielvariable aber auch ein Eintrag mit `NA` enthalten sein.

```
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`))


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)
print(rmse(drop_na(forec)))
```

```
forec %>%
  pivot_longer(cols = -time, names_to = "id", values_to = "value") %>%
  ggplot(aes(x=  time, y = value, color = id)) +
  geom_line()
```

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. Der RMSE beträgt 0.12.

4.4.2 Beispiel : VDAX

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 - PRICE INDEX`))


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)
print(rmse(drop_na(forec)))
```

```
forec %>%
  pivot_longer(cols = -time, names_to = "id", values_to = "value") %>%
  ggplot(aes(x=  time, y = value, color = id)) +
  geom_line()
```

4.5 gtpreparation

Die Funktion `gtpreparation()` lädt für verschiedene Suchbegriffe oder Kategorien Zeitreihen herunter, logarithmiert und saisonal bereinigt diese. Danach werden erste Differenzen berechnet. Diese werden im Endeffekt auch zurückgegeben.

```
gtpreparation(keyword = "ikea", time = "2015-01-01 2021-01-01")
#> # A tsibble: 73 x 3 [1D]
#> # Key:       id [1]
#>    time       id             lag_0
#>    <date>     <chr>          <dbl>
#>  1 2015-01-01 All categories 0.385
#>  2 2015-02-01 All categories 0.366
#>  3 2015-03-01 All categories 0.374
#>  4 2015-04-01 All categories 0.395
#>  5 2015-05-01 All categories 0.411
#>  6 2015-06-01 All categories 0.415
#>  7 2015-07-01 All categories 0.411
#>  8 2015-08-01 All categories 0.428
#>  9 2015-09-01 All categories 0.427
#> 10 2015-10-01 All categories 0.387
#> # ... with 63 more rows
```

4.6 gtsearch

Dies ist eine sehr simple Wrapper-Funktion um `gtrends()` herum, die die Google Daten schon ganz grundlegend für unsere Zwecke aufbereitet.

```
gtsearch(keyword = c("pluto", "saturn"), timeframe = "2020-01-01 2020-06-01")
#> # A tsibble: 306 x 3 [1D]
#> # Key:       keyword [2]
#>    date       hits  keyword
#>    <date>     <chr> <chr>
#>  1 2020-01-01 <1    pluto
#>  2 2020-01-02 1     pluto
#>  3 2020-01-03 <1    pluto
#>  4 2020-01-04 2     pluto
#>  5 2020-01-05 2     pluto
#>  6 2020-01-06 2     pluto
#>  7 2020-01-07 1     pluto
#>  8 2020-01-08 1     pluto
#>  9 2020-01-09 1     pluto
#> 10 2020-01-10 2     pluto
#> # ... with 296 more rows
```

4.7 gtseas_adj

Falls man eine saisonale Bereinigung von Google Zeitreihen braucht, bietet sich diese Funktion an. Entweder übergibt man ihr eine Zeitreihe in Form eines tibbles oder tsibbles oder übergibt einen Suchbegriff/eine Kategorie, nach der gesucht wird und dann direkt eine saisonale Bereinigung durchgeführt wird. Die saisonale Bereinigung wird mittels dem X-13ARIMA-SEATS-Verfahren oder mithilfe der Bildung von ersten Differenzen durchgeführt.

```
# Saisonale Bereinigung einer bereits erstellten Google Zeitreihe
series <- trendecon::ts_gtrends(c("ikea", "saturn"), time = "2020-01-01 2021-06-01")
gtseas_adj(series, freq = "month", log.traf = TRUE, method = "firstdiff")
#> # A tsibble: 148 x 3 [7D]
#> # Key:       id [2]
#> # Groups:    id [2]
#>    id    time       value
#>    <chr> <date>     <dbl>
#>  1 ikea  2020-01-05     0
#>  2 ikea  2020-01-12     0
#>  3 ikea  2020-01-19     0
#>  4 ikea  2020-01-26     0
#>  5 ikea  2020-02-02     0
#>  6 ikea  2020-02-09     0
#>  7 ikea  2020-02-16     0
#>  8 ikea  2020-02-23     0
#>  9 ikea  2020-03-01     0
#> 10 ikea  2020-03-08     0
#> # ... with 138 more rows

# Erstellen und dann direkte Saisonbereinigung einer Google Zeitreihe
gtseas_adj(category = 179, timeframe = "2015-01-01 2021-01-01", method = "arima")
#> # A tsibble: 73 x 2 [1D]
#>    time       value
#>    <date>     <dbl>
```

```
#>  1 2015-01-01  73.9
#>  2 2015-02-01  75.3
#>  3 2015-03-01  74.5
#>  4 2015-04-01  75.0
#>  5 2015-05-01  75.8
#>  6 2015-06-01  76.3
#>  7 2015-07-01  73.6
#>  8 2015-08-01  71.5
#>  9 2015-09-01  72.9
#> 10 2015-10-01  72.3
#> # ... with 63 more rows
```

4.8 gttrend_adj

Falls eine Trendbereinigung von Google Zeitreihen benötigt wird, bietet sich stattdessen diese Funktion an. Hier haben wir drei Möglichkeiten der Trendbereinigung: Mittels erste Differenzen; Mittels eines gleitenden Durchschnitts oder eine Bereinigung des allgemeinen Trends (siehe oben). Genauso wie `gtseas_adj()` kann hier auch entweder eine fertige Zeitreihe oder ein Suchbegriff/eine Kategorie, die direkt trendbereinigt werden, übergeben werden.

```
# Trendbereinigung einer bereits erstellten Google Zeitreihe
series <- trendecon::ts_gtrends("ikea", time = "all")
gttrend_adj(series, log.trafo = TRUE, method = "moving_avg")
#>               Jan          Feb          Mar          Apr          May
#> 2004  -2.91447851   3.13125885   1.17699622  -2.80574091  -4.78847804
#> 2005   2.08373736   0.60262172  -1.87849393  -3.22429277   0.42990838
#> 2006  -0.94110061  -3.40071451   0.13967158  -0.33929961  -3.81827081
#> 2007   4.39704864  -0.08712570  -4.57130005  -0.97076664   0.62976676
#> 2008   0.61363668   2.21734209  -3.17895250  -3.54095009  -5.90294769
#> 2009   1.01229597   0.94064679  -3.13100239  -1.24013267  -3.34926294
#> 2010  -0.39917048  -3.54602955   2.30711138  -1.89028285   0.91232292
#> 2011   3.78489454   0.11509824  -4.55469805   0.58715553  -4.27099089
#> 2012   1.43606564  -4.68783372  -4.81173308   2.13733594  -4.91359503
#> 2013  -1.91273104  -2.21747645  -0.52222185   2.89889584  -1.67998647
#> 2014   5.54237536   0.55374909  -1.43487718   2.88608045  -8.79296191
#> 2015  -0.82058677  -5.11261139  -2.40463602   1.13825972  -2.31884455
#> 2016   3.01409067  -2.19990732  -2.41390531   1.19696458  -2.19216554
#> 2017  -0.05418384  -2.21981712  -3.38545039   4.14834633  -7.31785696
#> 2018   6.90437006  -3.08366419  -0.07169843   0.05221573  -3.82387011
#> 2019   2.58823552   0.45956314  -3.66910924   1.14986592  -5.03115892
#> 2020  -0.24317823  -7.74946331 -18.25574839  -8.59711859  17.06151120
#> 2021  11.10452124  -5.05005352  13.79537172  10.14678234  -0.50180703
#>               Jun          Jul          Aug          Sep          Oct
#> 2004  -1.80355143   1.18137519  -3.84123549   0.13615382   0.02042898
#> 2005  -1.73408313   0.10192536   1.98500960  -0.13190615  -0.27686109
#> 2006  -2.27689689  -1.73552297   5.79906233  -0.66635237  -1.11891819
#> 2007  -2.60702378   2.15618568   0.01670678  -2.12277213   1.86259979
#> 2008  -4.07977035   3.74340698   5.57078049   0.39815400   2.25465673
#> 2009  -3.29911703   0.75102888   1.81780236   3.88457584   2.83196501
#> 2010  -5.30119131  -0.51470554   2.29670820   0.10812194   3.87147569
#> 2011  -2.03287187  -2.79475286   2.76838530   1.33152345   6.06069781
#> 2012  -4.01340444  -1.11321386   2.62501176   2.36323737   1.01742670
#> 2013  -7.61671268  -5.55343890  -0.61915854  -2.68487818   1.37386988
#> 2014 -11.02732080  -0.26167969   9.59928669   0.46025306   4.16370755
```

```
#> 2015  -7.94484582  -1.57084709   5.82328306  -0.78258679   2.79829639
#> 2016 -11.46708575  -7.74200597   4.04884257   4.83969111   6.58657183
#> 2017 -12.21596142  -4.11406588  -0.04948450   1.01509688   9.31296475
#> 2018 -13.83614950  -5.84842890  -0.04607673   1.75627544   5.44292439
#> 2019 -11.38594942  -5.74073992   1.95405021   0.64884033   1.77787648
#> 2020   8.60659089  -6.84832941  -1.29067110  -1.73301279   5.77308421
#> 2021 -15.01588979  -7.52997254   2.11800341   2.76597937   3.52684868
#>               Nov          Dec
#> 2004   2.90470413   1.49422075
#> 2005   0.57818397   5.31854168
#> 2006   1.42851600   1.91278232
#> 2007   3.84797170   1.73080419
#> 2008   1.11115946   1.06172771
#> 2009  -1.22064582   0.69009185
#> 2010  -2.36517056  -0.79013801
#> 2011   4.78987216   4.61296890
#> 2012   0.67161602   5.37944249
#> 2013   4.43261795   8.48749665
#> 2014   2.86716203   3.52328763
#> 2015   2.37917957   7.19663512
#> 2016   9.33345254  -0.86036565
#> 2017   5.61083261   8.25760133
#> 2018   5.12957335   7.85890443
#> 2019  10.90691263   6.33186720
#> 2020  10.27918121   0.19185123
#> 2021   0.28771799

# Erstellen und dann direkte Trendbereinigung einer Google Zeitreihe
gttrend_adj(
  category = 179, timeframe = "2015-01-01 2021-01-01",
  method = "moving_avg", log.trafo = FALSE
)
#>              Jan         Feb         Mar         Apr         May         Jun
#> 2015   3.2913222  -0.1612362  -3.6137947  -4.1649171   2.2839605   7.6106489
#> 2016   8.8226199   2.2403590  -0.3419019  -3.7893386  -0.2367754   3.6811845
#> 2017   1.6664662  -3.9195465  -6.5055592  -4.0854457   1.3346678   6.8541882
#> 2018   5.3622244  -2.5300390  -2.4223023  -5.3487832  -1.2752642   4.6728292
#> 2019   5.1500209  -3.0939016  -4.3378241  -4.4836223   5.3705795   4.3814668
#> 2020  12.7718047   6.6137978 -16.5442092 -31.4554027 -11.3665962   8.1327083
#> 2021  -6.4850249
#>              Jul         Aug         Sep         Oct         Nov         Dec
#> 2015  11.9373374   7.2135867  -0.5101640  -7.4442455 -13.3783269 -19.7778535
#> 2016  13.5991443  14.7412400   3.8833357  -3.2471874 -13.3777105 -14.8556222
#> 2017  15.3737086  16.0347032   3.6956978   1.5445149 -14.6066680 -17.6222218
#> 2018  17.6209225  10.5270895   1.4332565  -2.8408263 -12.1149091 -14.4824441
#> 2019  15.3923541   9.2646590   7.1369640   0.2444691  -5.6480257  -4.4381105
#> 2020  23.6320128  23.0313600  15.4307072   0.7066166 -15.0174739 -12.7512494
#> 2021
```

4.9 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 = "2018-01-01 2020-01-01")
#> # A tibble: 25 x 5
#>    date          PC1    PC2  ikea saturn
#>    <date>      <dbl>  <dbl> <int>  <int>
#>  1 2018-01-01   4.78  6.72     81     56
#>  2 2018-02-01  -5.35  0.933    71     50
#>  3 2018-03-01  -5.35  0.933    71     50
#>  4 2018-04-01 -13.0  -0.496    66     44
#>  5 2018-05-01 -14.7  -7.58     59     46
#>  6 2018-06-01 -14.3  -8.94     58     47
#>  7 2018-07-01 -12.4  -5.45     62     47
#>  8 2018-08-01  -3.39  4.42     75     50
#>  9 2018-09-01   5.44  1.77     77     59
#> 10 2018-10-01   2.05  5.96     79     54
#> # ... with 15 more rows
```

4.10 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 Funktionen können Download-Funktionen 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)
```

5 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. Abbildung **??** 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))


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 CONA`))


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)
print(rmse(drop_na(forec)))
```

```
forec %>%
  pivot_longer(cols = -time, names_to = "id", values_to = "value") %>%
  ggplot(aes(x=  time, y = value, color = id)) +
  geom_line()
```

6 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.

```
filter(gtrendsR::categories, id %in% c(18,78,68,531,355,121,841))

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 (CAL ADJ) X-12-ARIMA VOLA`)
    )


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())
```

```
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()
```

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(forec2$forec %>%
       left_join(dat, by = "time") %>%
       drop_na())
```

```
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()
```

```
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~.)
```

7 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.

8 Bekannte 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. Das bedeutet, dass nun solange gewartet werden muss, bis dieser Bann abgeklungen ist (erfahrungsgemäß ist das etwa nach 24 Stunden der Fall.)

9 Quellen

Google 2021a: https://support.google.com/trends/answer/4359550?hl=en
Google 2021b: https://support.google.com/trends/answer/4359597?hl=en
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.
Götz und Knetsch (2019): *Google data in bridge equation models for German GDP*. International Journal of Forecasting 35 (2019) 45-66