New Data Sources and Official Statistics

An Exemplary Research Workflow in R Using Different API Wrapper Packages

Yannik Buhl,
Federal Statistical Office Germany (Destatis)

The Use of R in Official Statistics (uRos)
November 2024, Athens

New data sources in official statistics

Goals

- Official statistics need continuous development
- Main focus on reducing response burden
- Second goal: Make statistics more timely and precise

Examples at Destatis

- Data from AirBnB for tourism statistics
- Mobile Network Operator data for population statistics
- Satellite data for construction statistics

The example of pedestrian count data

Research idea

- Need for very timely data for high-street retail statistics
- Estimate most recent trends for economic statistics
- Rising digitisation in pedestrian counts

Pedestrian count data

- Working with German start-up company ‘Hystreet’
- Count pedestrians on high-streets of many German cities (and beyond)
- Started cooperating during Covid-19
- Laser sensors count pedestrians passing

The example of pedestrian count data

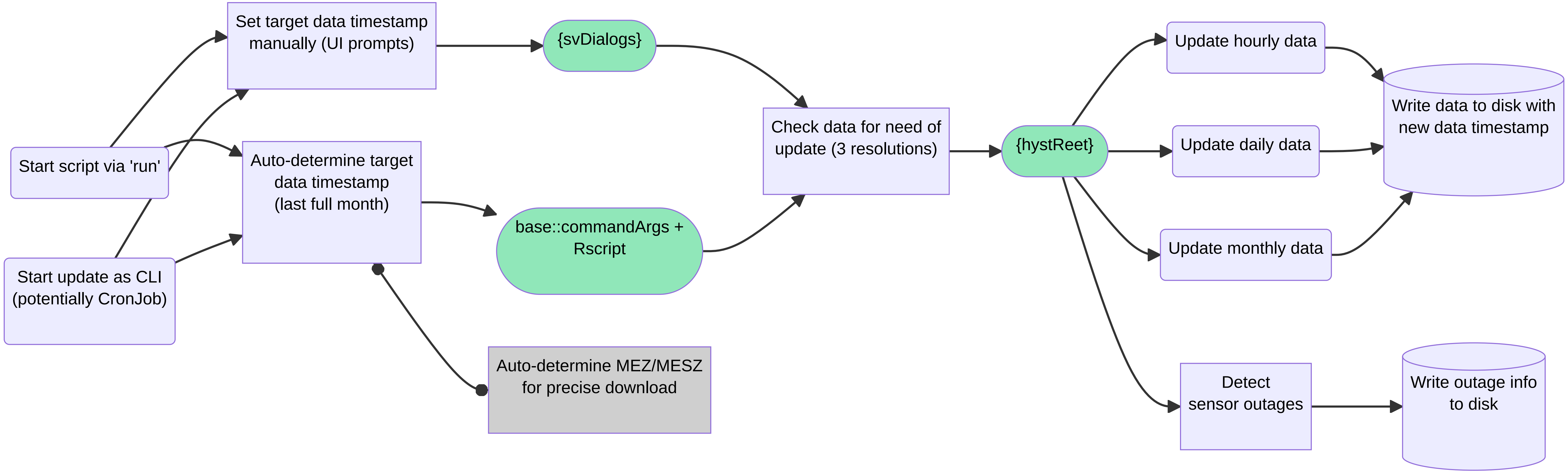
Use-case requirements

- Create a variety of data products, updated weekly
- Deliver data products to stakeholders and customers
- Maintain some degree of data quality (experimental, though)
- Calculate pedestrian count index (German Federal Bank)
- Automate workflow

Why use R?

- API wrapper for ‘Hystreet’ data: {hystReet}
- API wrapper for Destatis’s database: {restatis}
- Potential to output all formats needed (e.g., Excel, Graphs, databases, etc.)
- Easy way of automating scripts

Production process: Update workflow



Set target data timestamp manually (UI prompts)
{svDialogs}
Start script via 'run'
Auto-determine target data timestamp (last full month)
Start update as CLI (potentially CronJob)
base::commandArgs + Rscript
Auto-determine MEZ/MESZ for precise download
Check data for need of update (3 resolutions)
{hystReet}
Update hourly data
Update daily data
Update monthly data
Write data to disk with new data timestamp
Detect sensor outages
Write outage info to disk

Figure 1: Update data workflow

Production process: {hystReet}

```r
df <- hystReet::get_hystreet_station_data(hystreetId = 148, # Limburg
                                          query = list(from = "2024-11-01T00:00:00+01:00:00
                                                       to = "2024-11-01T23:00:00+01:00:00",
                                                       resolution = "hour"),
                                          API_token = Sys.getenv("HYSTREET_KEY")) %>%
        magrittr::extract2("measurements") %>%
        rowwise() %>%
        filter(isFALSE(unverified)) %>%
        select(-c(min_temperature, unverified, collection_type, details))

head(as_tibble(df))
```

```
# A tibble: 6 × 4
  timestamp           weather_condition   temperature pedestrians_count
  <dttm>              <chr>                     <dbl>             <int>
1 2024-11-01 00:00:00 partly-cloudy-night         9.8                23
2 2024-11-01 01:00:00 cloudy                      8.5                13
3 2024-11-01 02:00:00 cloudy                      8.5                 5
4 2024-11-01 03:00:00 cloudy                      8.1                 1
5 2024-11-01 04:00:00 cloudy                      8.1                11
6 2024-11-01 05:00:00 cloudy                      8.1                14
```

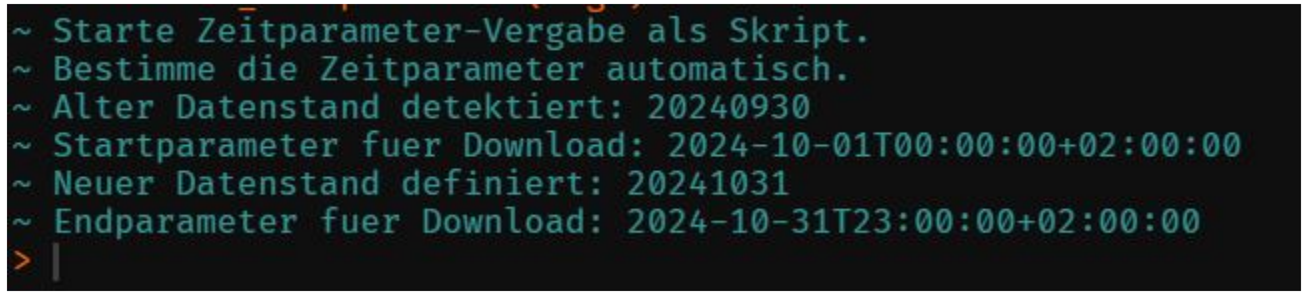
How do we put R in production?

Situation

- Weekly updates
- Weekly data exchanges
- Send and receive data
- Post-process and publish
- Few human resources

Solution

- Automate as far as possible
- Enable people not apt to work with R to do updates
- Steer them through the process (GUI style, e.g. w/ {svDialogs})
- Reliable production independent of acting staff



```
~ Starte Zeitparameter-Vergabe als Skript.
~ Bestimme die Zeitparameter automatisch.
~ Alter Datenstand detektiert: 20240930
~ Startparameter fuer Download: 2024-10-01T00:00:00+02:00:00
~ Neuer Datenstand definiert: 20241031
~ Endparameter fuer Download: 2024-10-31T23:00:00+02:00:00
> 
```

Figure 2: Console logging

Production process: R as CLI

- One step further: Integrating R with the command line
- R can work well used in, e.g., Bash scripts
- *Rscript* command available to run entire scripts
- Pass on any amount of parameters to the R script
  - *‘~$ Rscript update-data.R 20241101 20241130’*
  - Fetch with, e.g., *base::commandArgs()*

Production process: Data processing

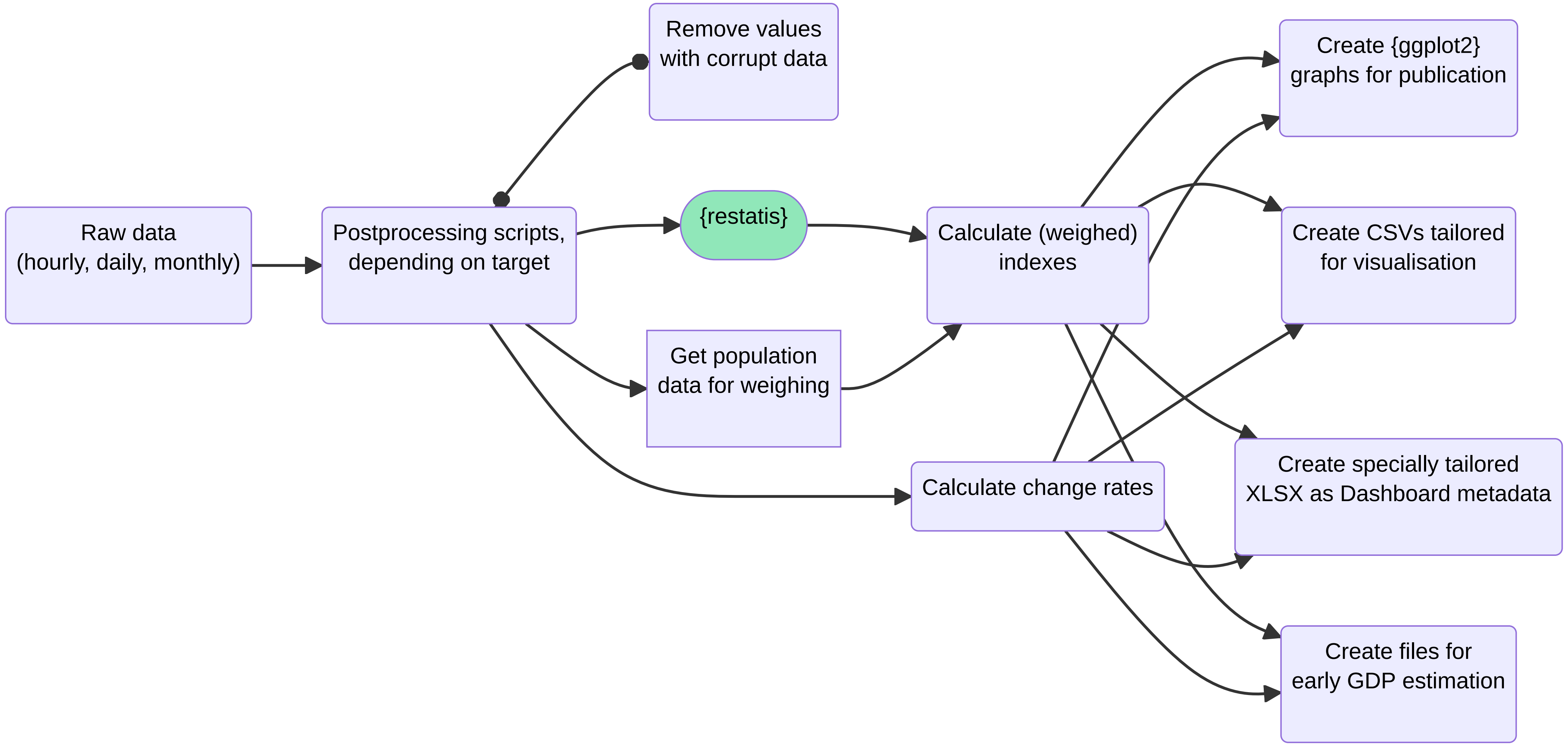

Remove values with corrupt data
Raw data (hourly, daily, monthly)
Postprocessing scripts, depending on target
{restatis}
Calculate (weighed) indexes
Create {ggplot2} graphs for publication
Create CSVs tailored for visualisation
Get population data for weighing
Calculate change rates
Create specially tailored XLSX as Dashboard metadata
Create files for early GDP estimation

Figure 3: Basic data processing workflow

Production process: {restatis}

```r
df2 <- restatis::gen_table("45212-0004", # Internet retail index
                           startyear = 2023,
                           endyear = 2024,
                           classifyingvariable1 = "WERTE4", classifyingkey1 = "REAL",
                           classifyingvariable2 = "WZ08E6", classifyingkey2 = "WZ08-4791")
      janitor::clean_names() %>%
      select(statistics_code, statistics_label, time,
             x1_variable_attribute_label, value, value_unit) %>%
      filter(value_unit == "%")

head(df2)
```

```
# A tibble: 6 × 6
  statistics_code statistics_label time  x1_variable_attribut…¹ value value_unit
  <chr>           <chr>            <chr> <chr>                  <chr> <chr>
1 45212           Monthly statist… 2024  May                    -4.5  %
2 45212           Monthly statist… 2024  April                  7.1   %
3 45212           Monthly statist… 2024  July                   3.4   %
4 45212           Monthly statist… 2024  June                   -5.3  %
5 45212           Monthly statist… 2024  September              16.3  %
6 45212           Monthly statist… 2023  January                -3.3  %
# ℹ abbreviated name: ¹x1_variable_attribute_label
```

Production process: Example results

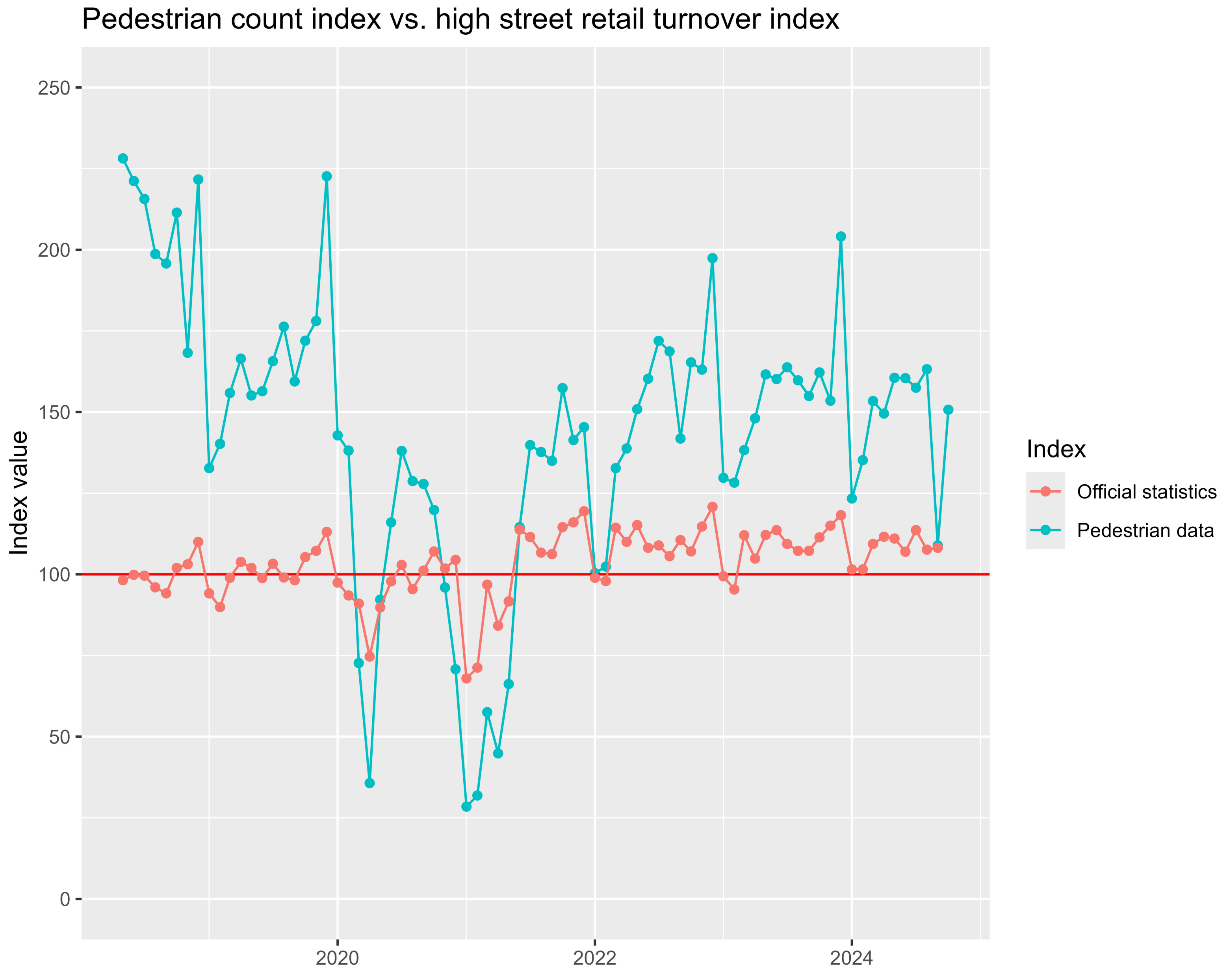


Figure 4: Timeseries of indexes

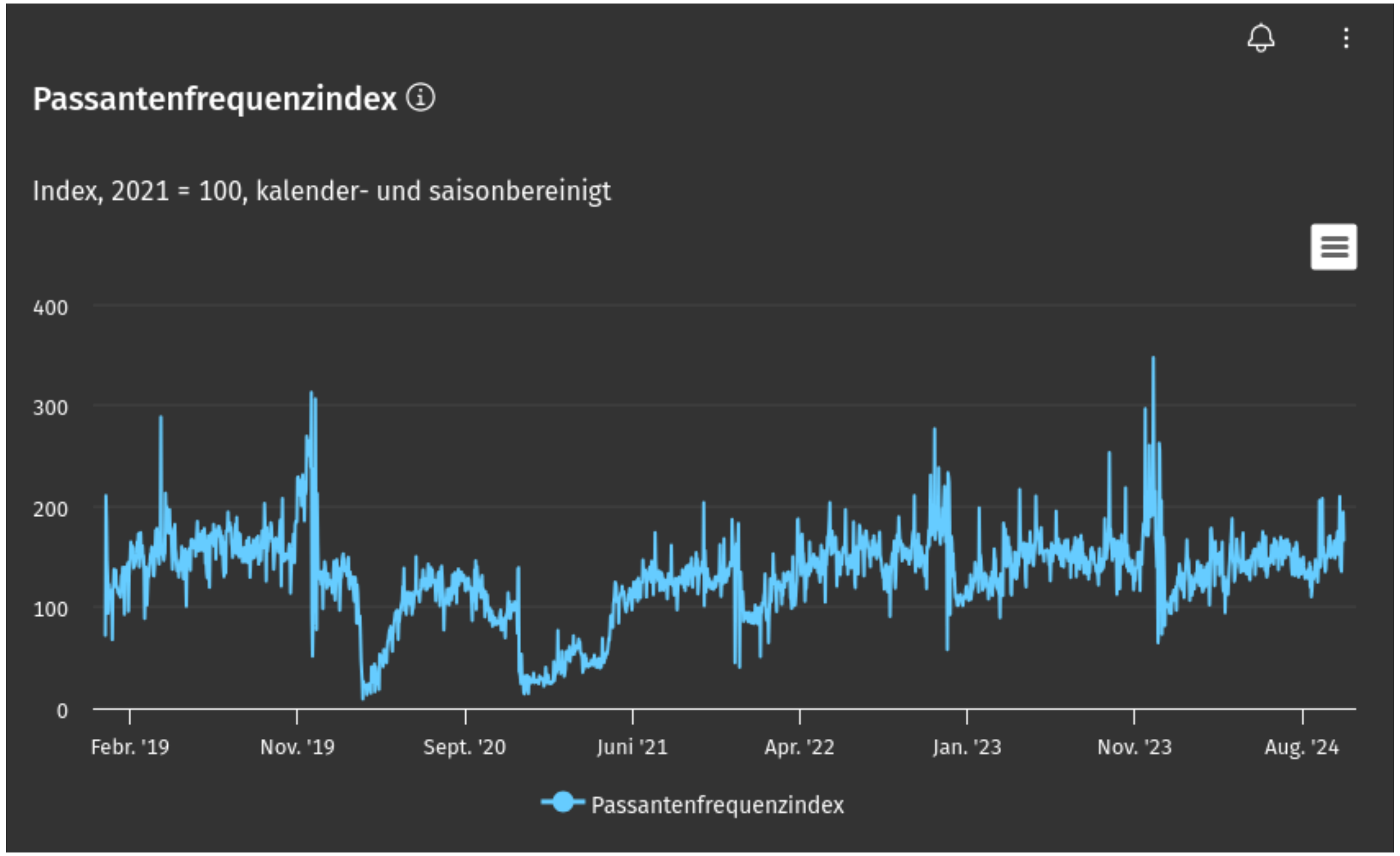


Figure 5: Daily pedestrian count index (seasonally adjusted, cooperation with German Federal Bank)

Summary: Takeaways & room for improvement

Takeaways

- Importance of APIs for automation in experimental statistics
- If there is no API wrapper, write one
- Putting R in production is nice especially for ‘smaller’ tasks

Room for improvement

- Better automation (CronJob)
- Potentially automate retries in case of HTTP error 5xx
- Potential future use-case for {targets}

Thank you!

Get in touch

yannik.buhl /at/ destatis.de

yannikbuhl (GitHub)