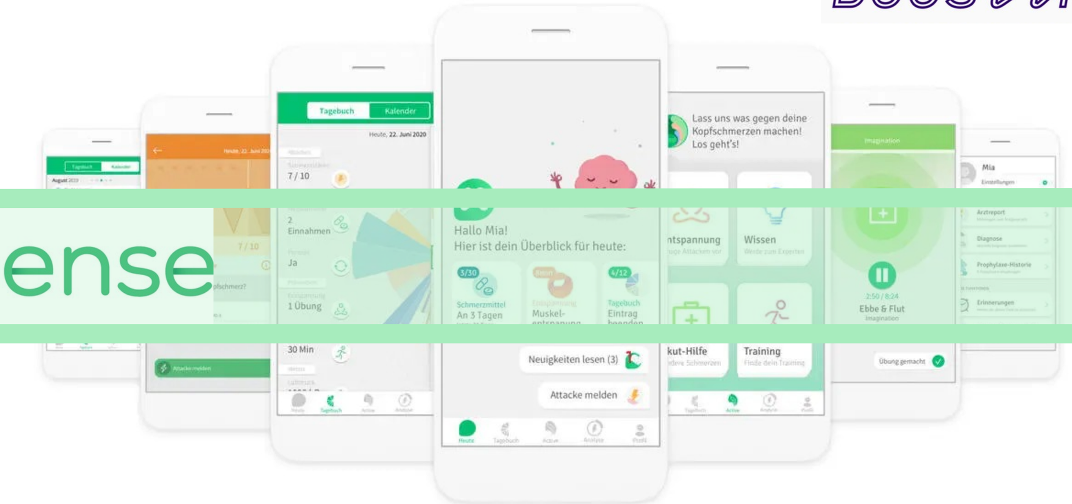
SPICED

# Migraine Data Analysis

Garlic Boosting

M-sense



scikit learn

PostgreSQL

pandas

SQLAlchemy

matplotlib

seaborn

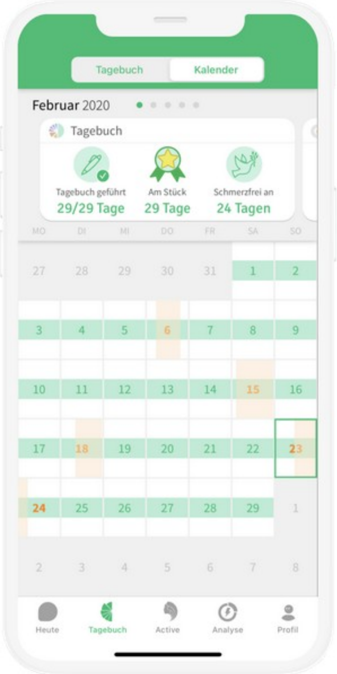
# Migraine Data Analysis

## Project Idea

- M-sense:
  - Headache Diary App
  - running for 6 years (2016-2022)
  - Data from 80000 users
- Backend: Postgres Database (running on aws and using Metabase )

**Questions:**

- Seasonality in the occurrence of headache days?
- What are potential headache triggers?
- Make use out of it for prediction?



Tagebuch Kalender

Februar 2020

Tagebuch

Tagebuch geführt 29/29 Tage

Am Stück 29 Tage

Schmerzfrei an 24 Tagen

| MO | DI | MI | DO | FR | SA | SO |
|---|---|---|---|---|---|---|
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |

M-sense

External Partner:
Markus Dahlem
(CEO Founder)

## I. Data Extraction

- Data were delivered as a binary Postgres dump
  Size: **26.2 GB**

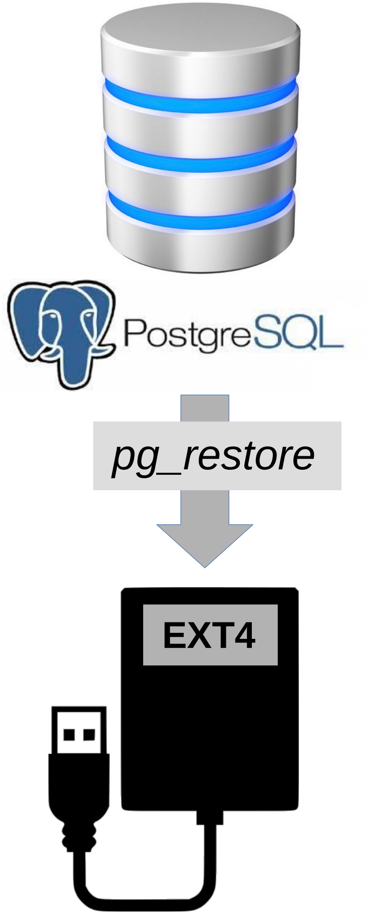
- First try: run it locally not possible due to HD restrictions

- Solutions:

  Set up database on external hard disk

  ➢ format external hard disk as EXT4
  ➢ mount hard disk as if it was an internal HD
  ➢ change postgres data path from /var/… to mount point

- Size of extracted data set: **137GB** Data of about 80.000 users

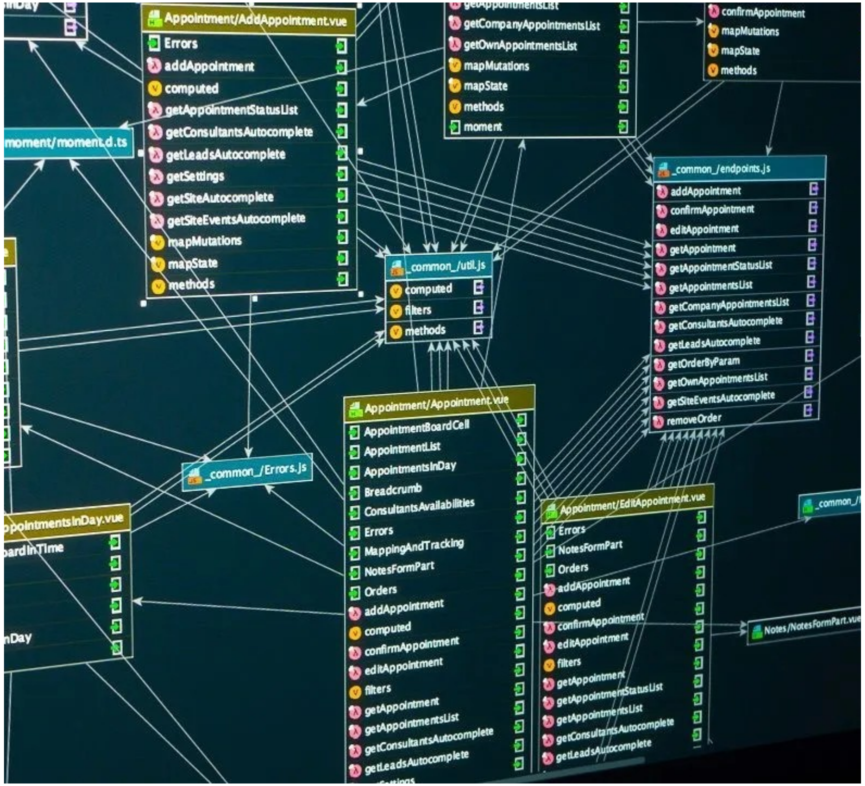


## II. Working with “Big Data”

Access the data:

- Database shape complicated
- advanced SQL-Queries are needed to retrieve headache time series data

- Solution:
  **python *Class* NslUser.py**
  - comfortable access data using SQLAlchemy
  - apply operations

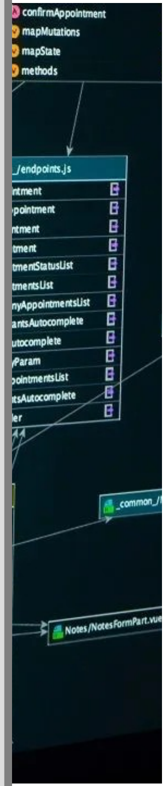


# Migraine Data Analysis

```
-- SQL-QUERY TO RETRIEVE HEADACHE TIME SERIES FOR USER 11 --

WITH CTE1 AS (
SELECT event_time_range, value FROM
      quantity q JOIN factor f
                   ON    q.user_id          =  f.user_id
                  AND    q.server_factor_id =  f.server_factor_id
                WHERE    q.user_id          =  11
                  AND    global_id          IN ('headacheType')
                  AND    q.deleted_at       IS null
             ORDER BY    LOWER(event_time_range)),

CTE2 AS (
SELECT GENERATE_SERIES(LOWER(event_time_range)::DATE,
                       UPPER(event_time_range)::DATE, '1 day') AS hd, value
FROM CTE1),

CTE_MIN_MAX AS (
SELECT MIN(hd) AS min_hd, MAX(hd) AS max_hd FROM CTE2),

CTE_TIMELINE AS (
SELECT GENERATE_SERIES(min_hd::DATE, max_hd::DATE, '1 day') AS timel FROM
CTE_MIN_MAX)

SELECT timel, CASE WHEN value IS null THEN 0 ELSE value END AS value_ FROM
      CTE_TIMELINE cte_t LEFT JOIN CTE2 c2
                                ON cte_t.timel = c2.hd
                          ORDER BY cte_t.timel;
```

Acce

- Da
- ac
  he
- Sc
  py

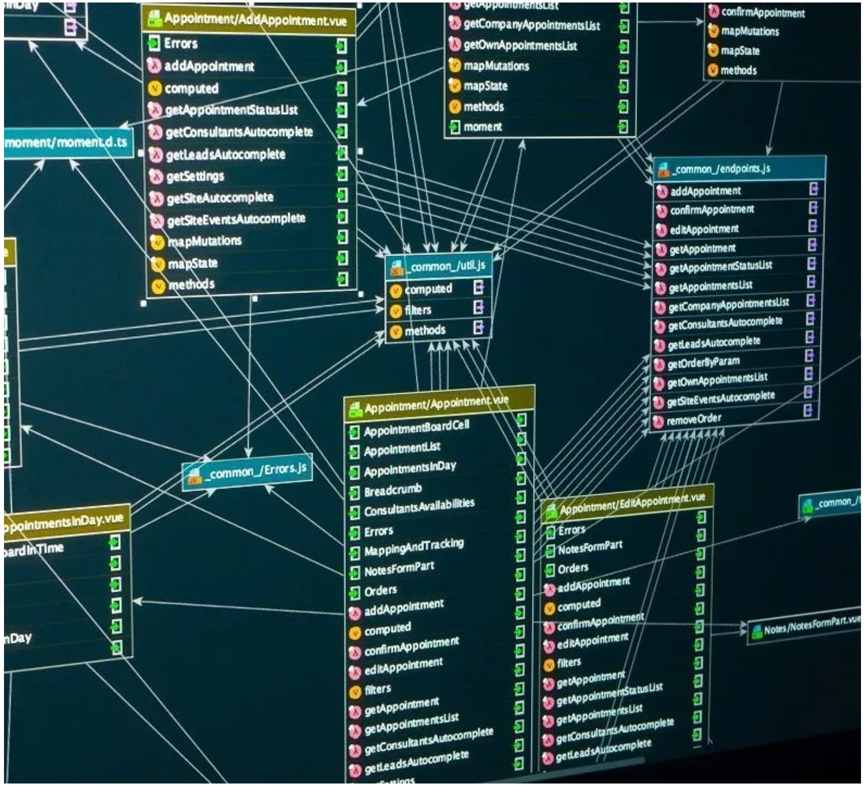


## II. Working with “Big Data”

Access the data:

- Database shape complicated
- advanced SQL-Queries are needed to retrieve headache time series data

- Solution:
  **python *Class* NslUser.py**
  - comfortable access data using SQLAlchemy
  - apply operations

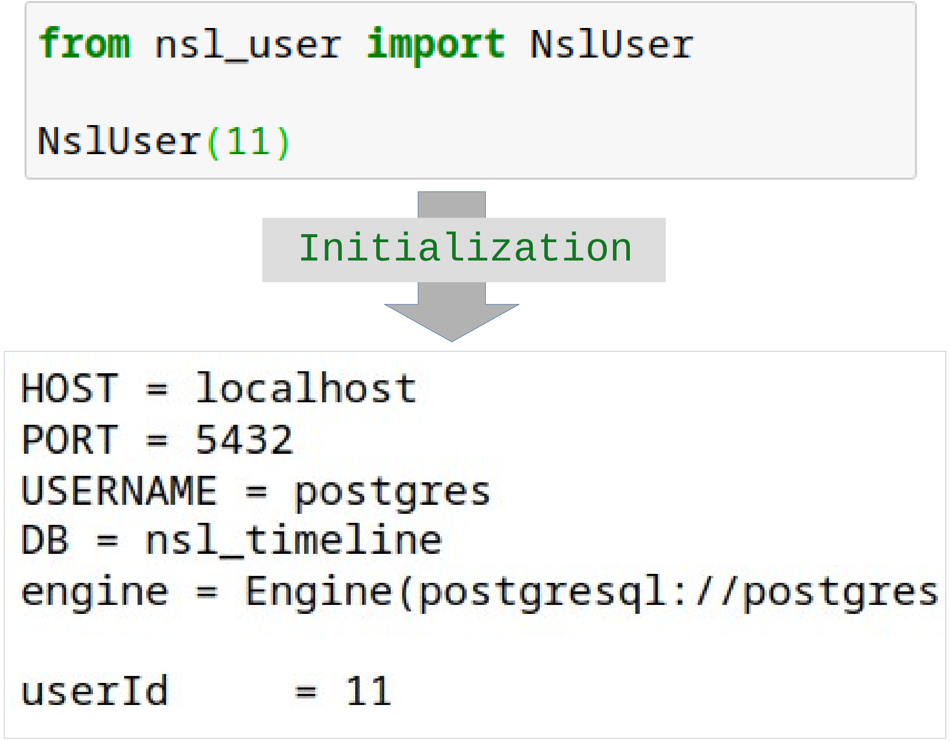


## II. Working with “Big Data”

Access the data:

- Database shape complicated
- advanced SQL-Queries are needed to retrieve headache time series data

- Solution:
  **python *Class* NslUser.py**
  - comfortable access data using SQLAlchemy
  - apply operations



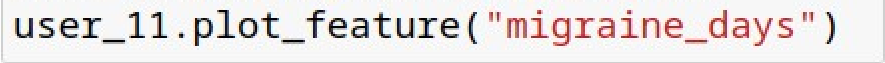
```
from nsl_user import NslUser

NslUser(11)
```

```
HOST = localhost
PORT = 5432
USERNAME = postgres
DB = nsl_timeline
engine = Engine(postgresql://postgres

userId     = 11
```

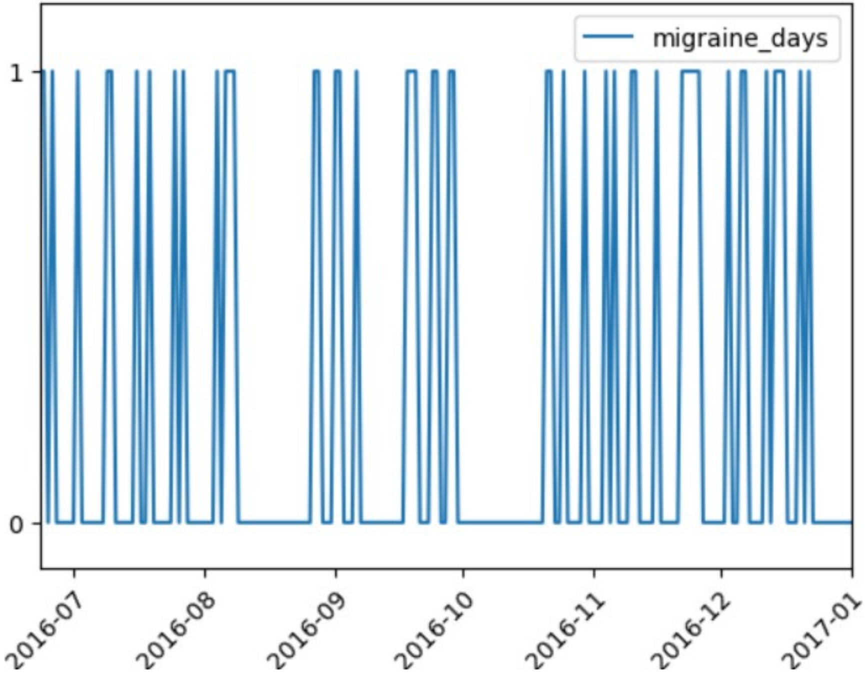
## II. Working with “Big Data”

Access the data:

- Database shape complicated
- advanced SQL-Queries are needed to retrieve headache time series data
- Solution:
  **python *Class* NslUser.py**
  - comfortable access data using SQLAlchemy
  - apply operations



```
user_11.plot_feature("migraine_days")
```



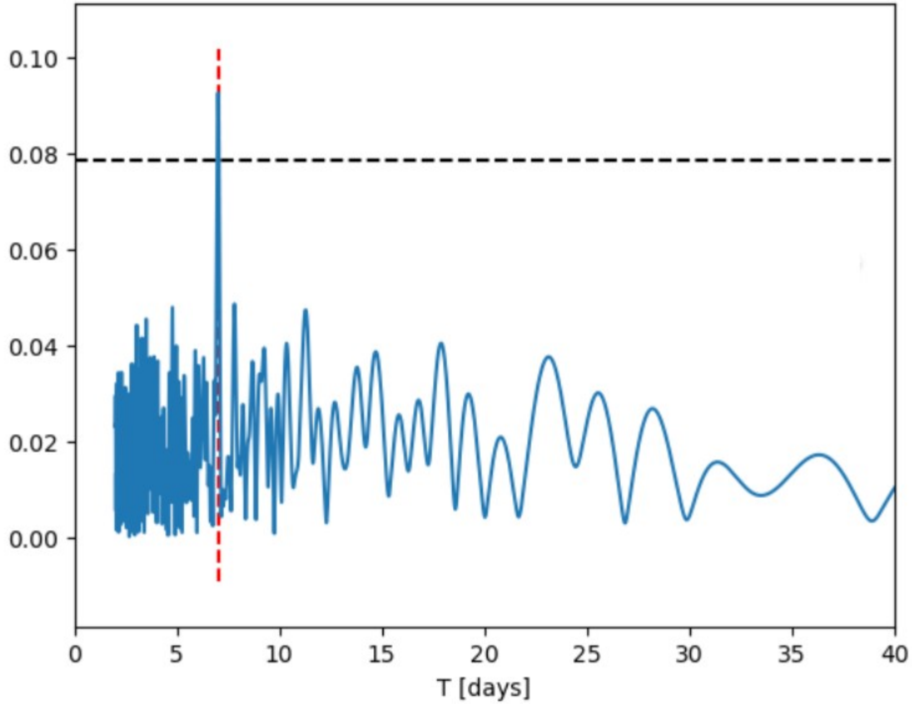
## II. Working with “Big Data”

Access the data:

- Database shape complicated
- advanced SQL-Queries are needed to retrieve headache time series data

- Solution:
  **python *Class* NslUser.py**
  - comfortable access data using SQLAlchemy
  - apply operations

```
user_11.plotspec()
```



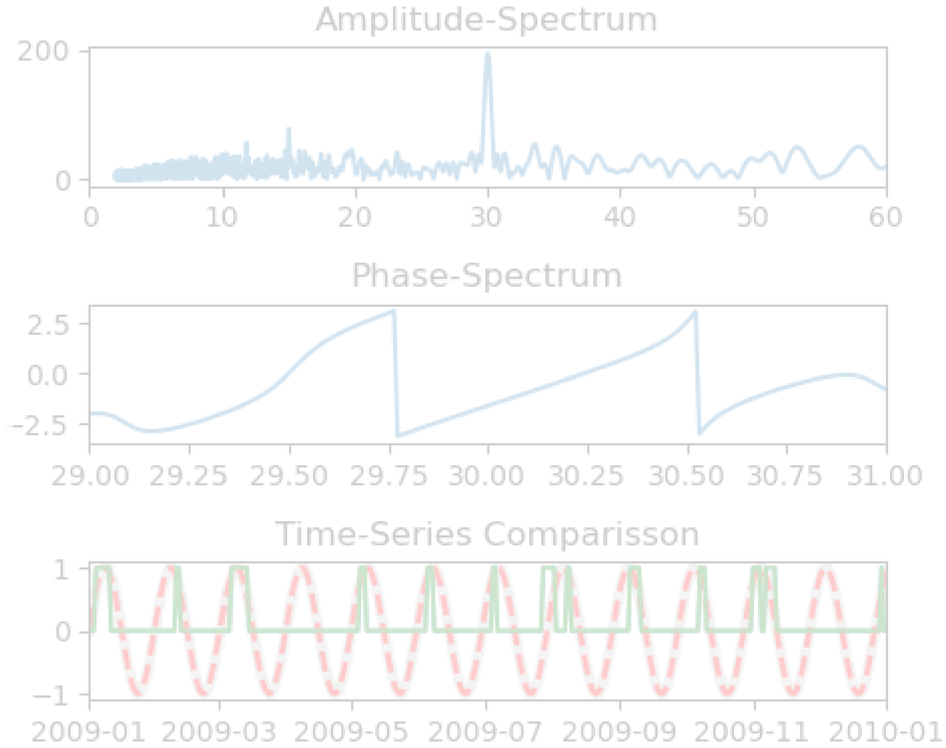
## Spectral Analysis

Each Time-Series can be approximated by sum of *cos* functions with different amplitudes, frequencies ω and phase φ

$$\Sigma A_k * \cos(\omega_k t + \varphi_k)$$

Fourier-Transformation (numpy.fft.fft):

- Unfolding spectral components:
  - *Amplitude* and *Phase* Spectrum: **A(ω)** *cos*(ωt + ***φ*(ω)**)
- Analyze whole dataset: ⇒ Spectra of 40000 user



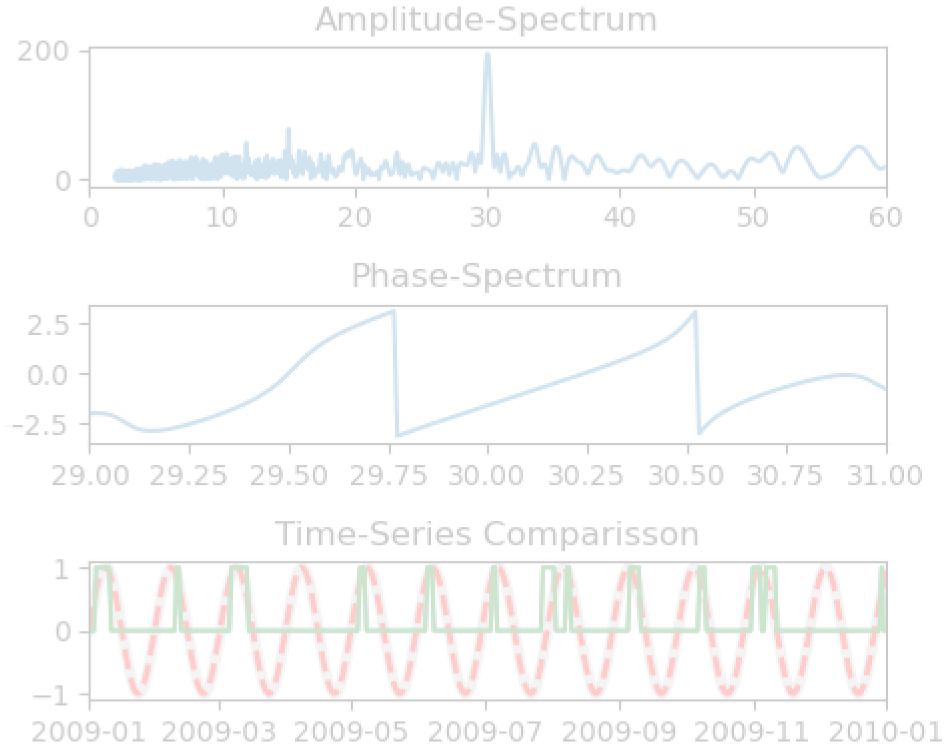
## Spectral Analysis

Each Time-Series can be approximated by sum of *cos* functions with different amplitudes, frequencies ω and phase φ

$$\Sigma A_k * cos(\omega_k t + \varphi_k)$$

Fourier-Transformation (numpy.fft.fft):

- Unfolding spectral components:
  - *Amplitude* and *Phase* Spectrum: **A(ω)** *cos*(ωt + ***φ(ω)***)
- Analyze whole dataset:
  ⇒ Spectra of 40000 user



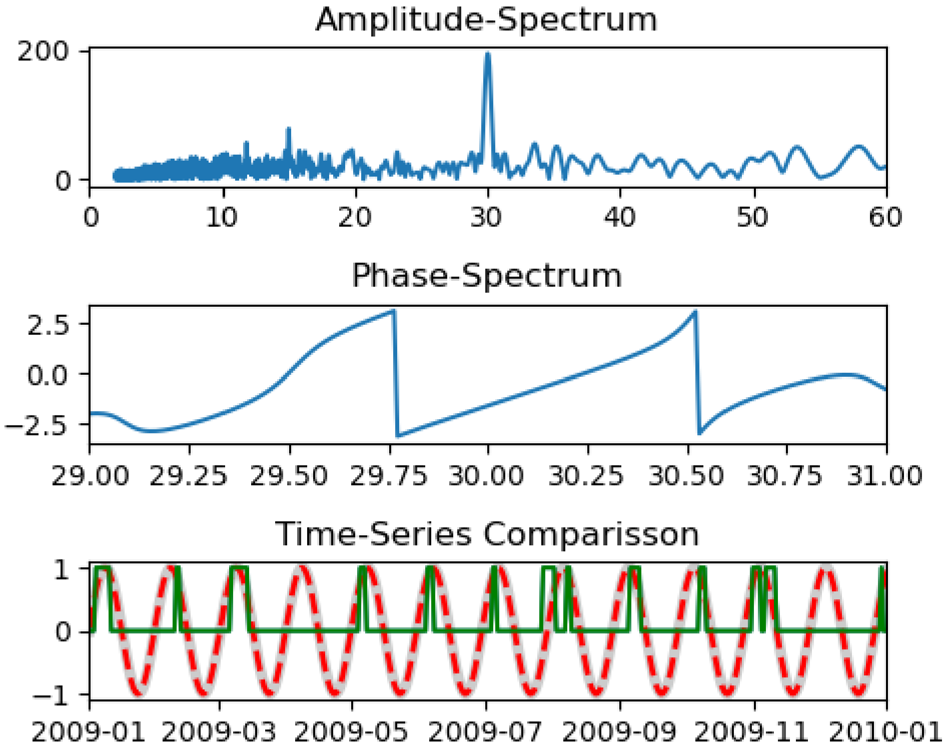
## Spectral Analysis

Each Time-Series can be approximated by sum of *cos* functions with different amplitudes, frequencies ω and phase φ

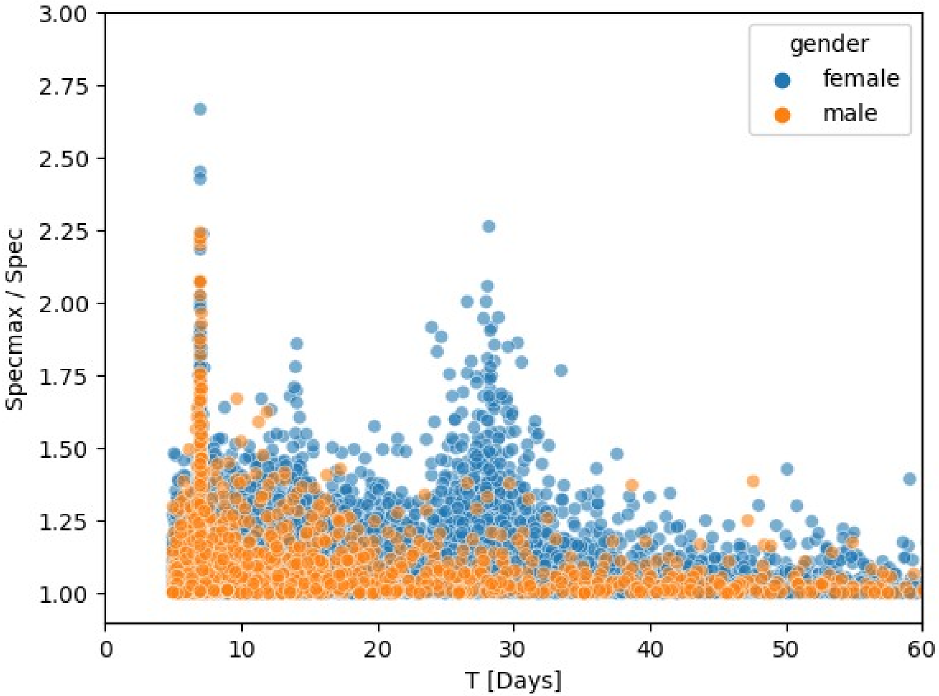
$$\Sigma A_k * \ cos(\omega_k t + \varphi_k)$$

Fourier-Transformation ( numpy.fft.fft):

- Unfolding spectral components:
  - *Amplitude* and *Phase* Spectrum:
    $\mathbf{A(\omega)}\ cos(\omega t + \boldsymbol{\varphi(\omega)})$
- Analyze whole dataset:
  ⇨ Spectra of 40000 user

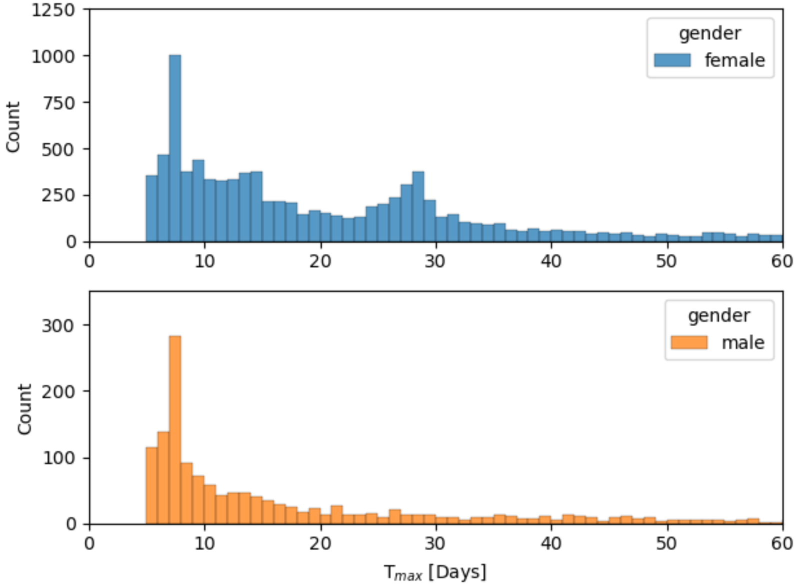


# Dominant Period



Overall about 35000

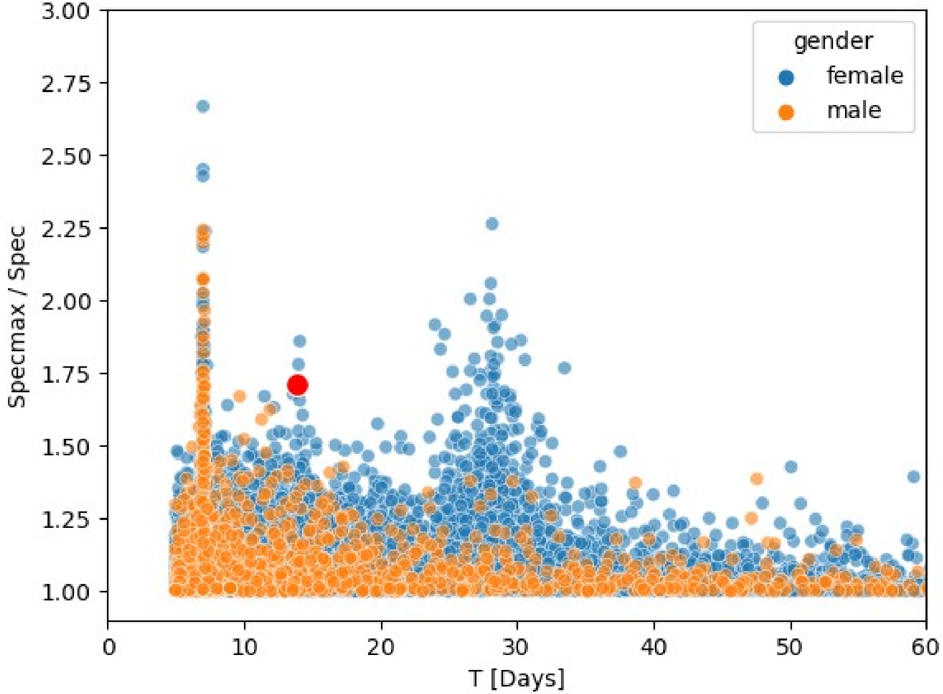
Count Plot of dominant Period:



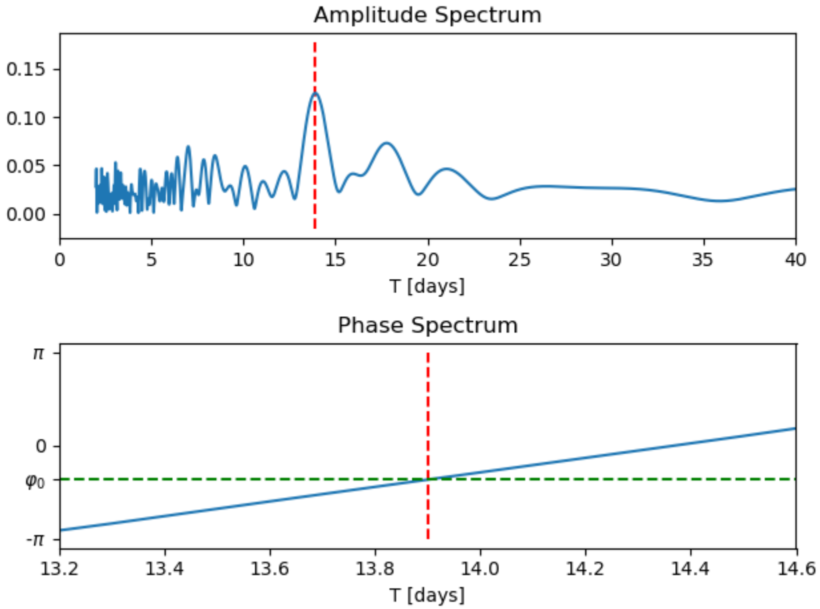
male + female: dominant peek only at 7 days

female : 2$^{nd}$ peak at menstruation cycle

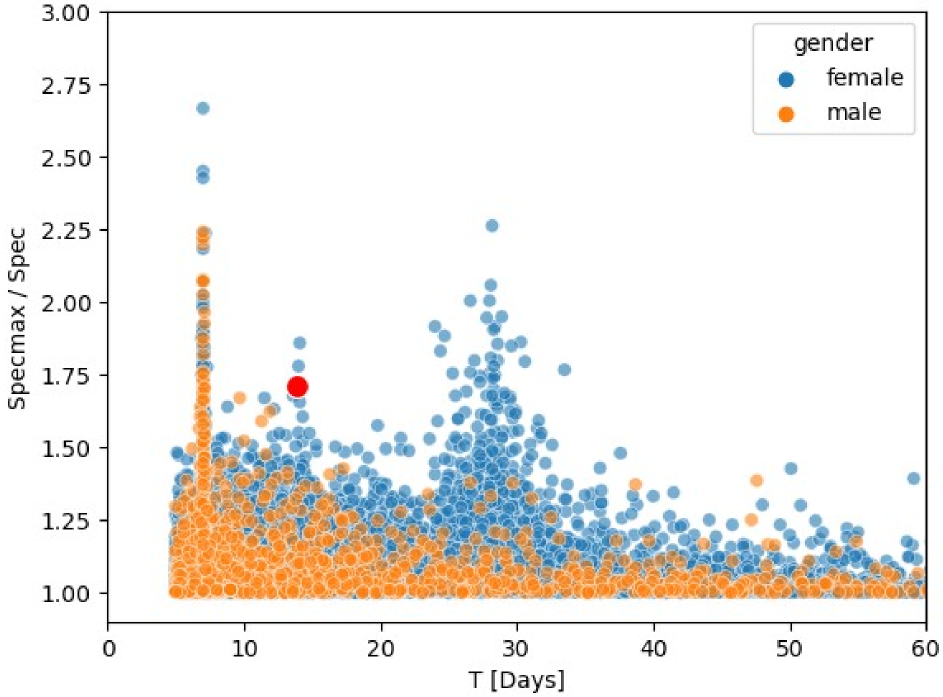
## Dominant Period



Overall about 35000

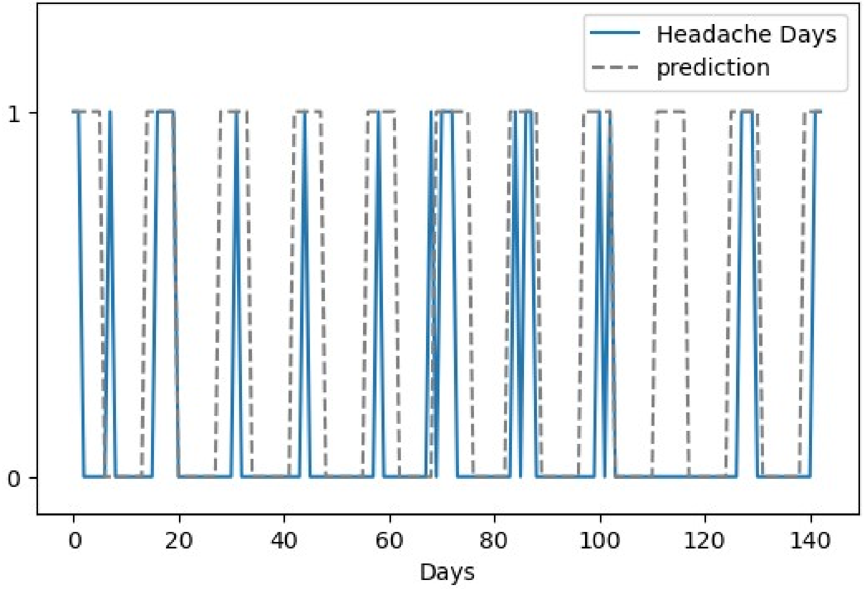


# Dominant Period



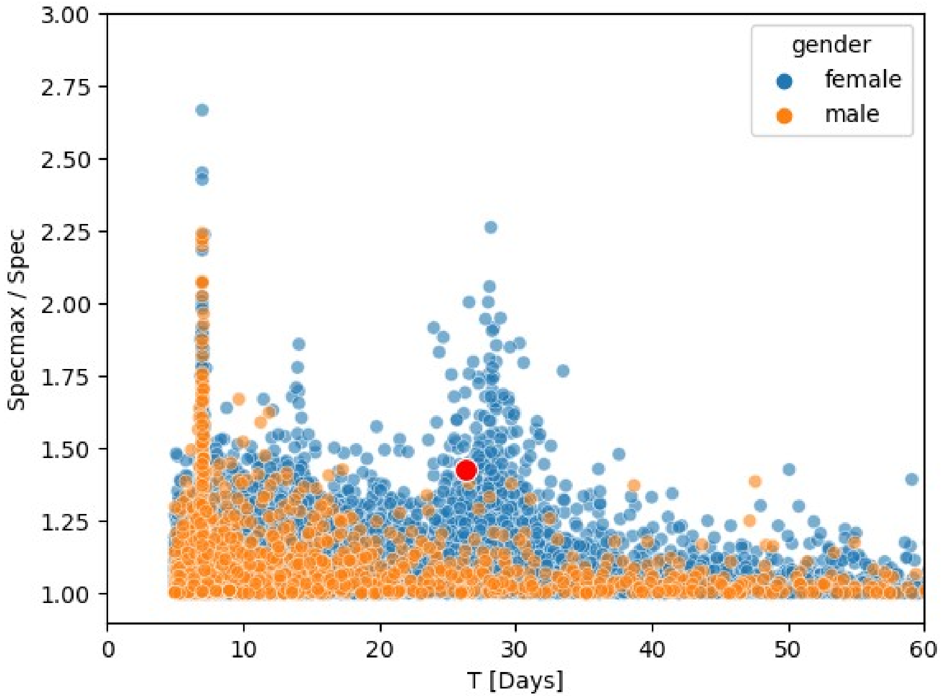
Overall about 35000

Periodicity based Prediction:



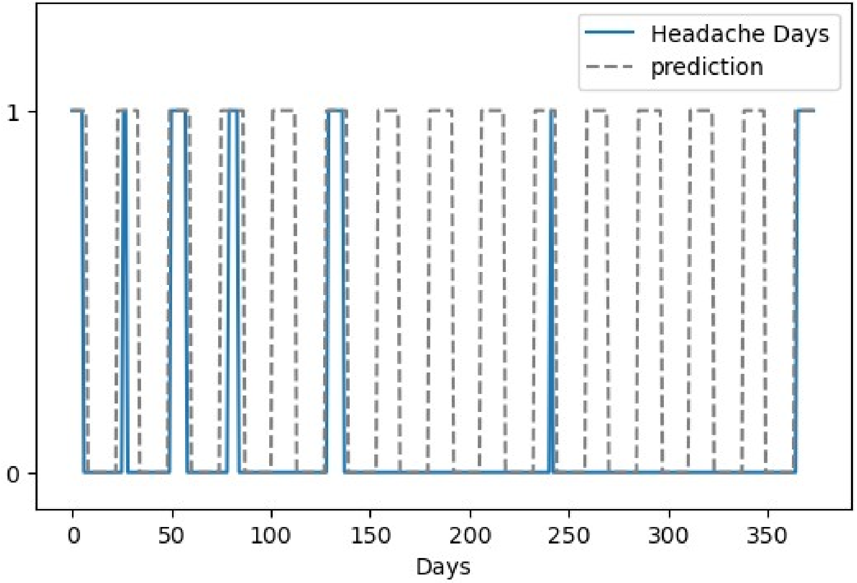
accuracy  and recall relatively high

## Dominant Period



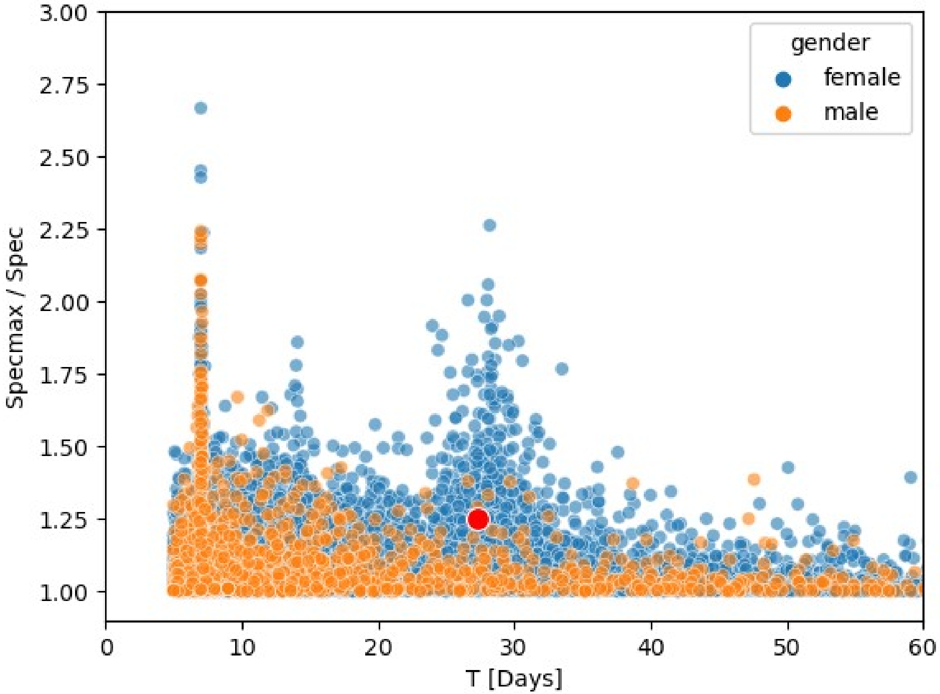
Overall about 35000

Periodicity based Prediction:



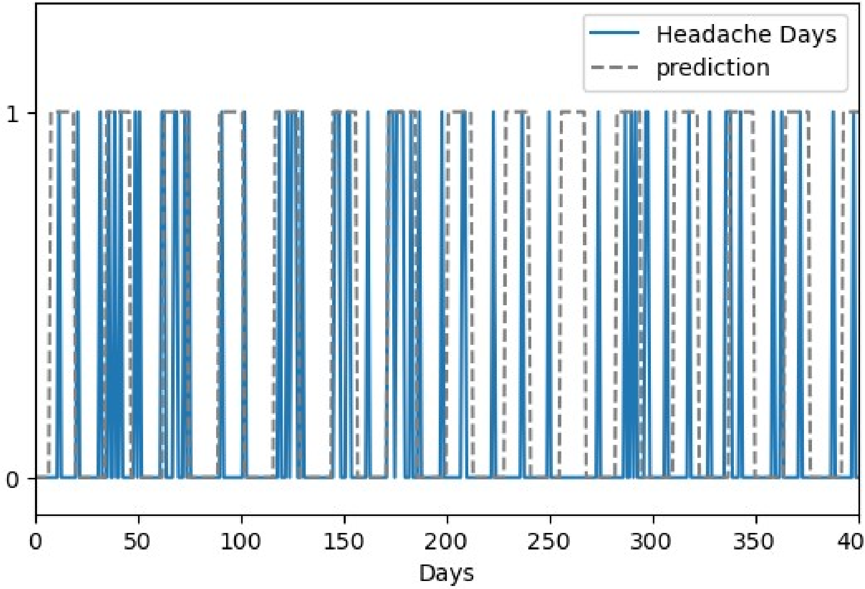
accuracy and recall relatively high

## Dominant Period



Overall about 35000

Periodicity based Prediction:



accuracy and lower recall
(many false negative)

# Thank You!

- Markus Dahlem M-sense
- Garlic Boosting - Cohort

  Especially:

  - Liljana: conceptual discussion
  - Florian: SQL-support

  - ALL GARLICS for a very good time

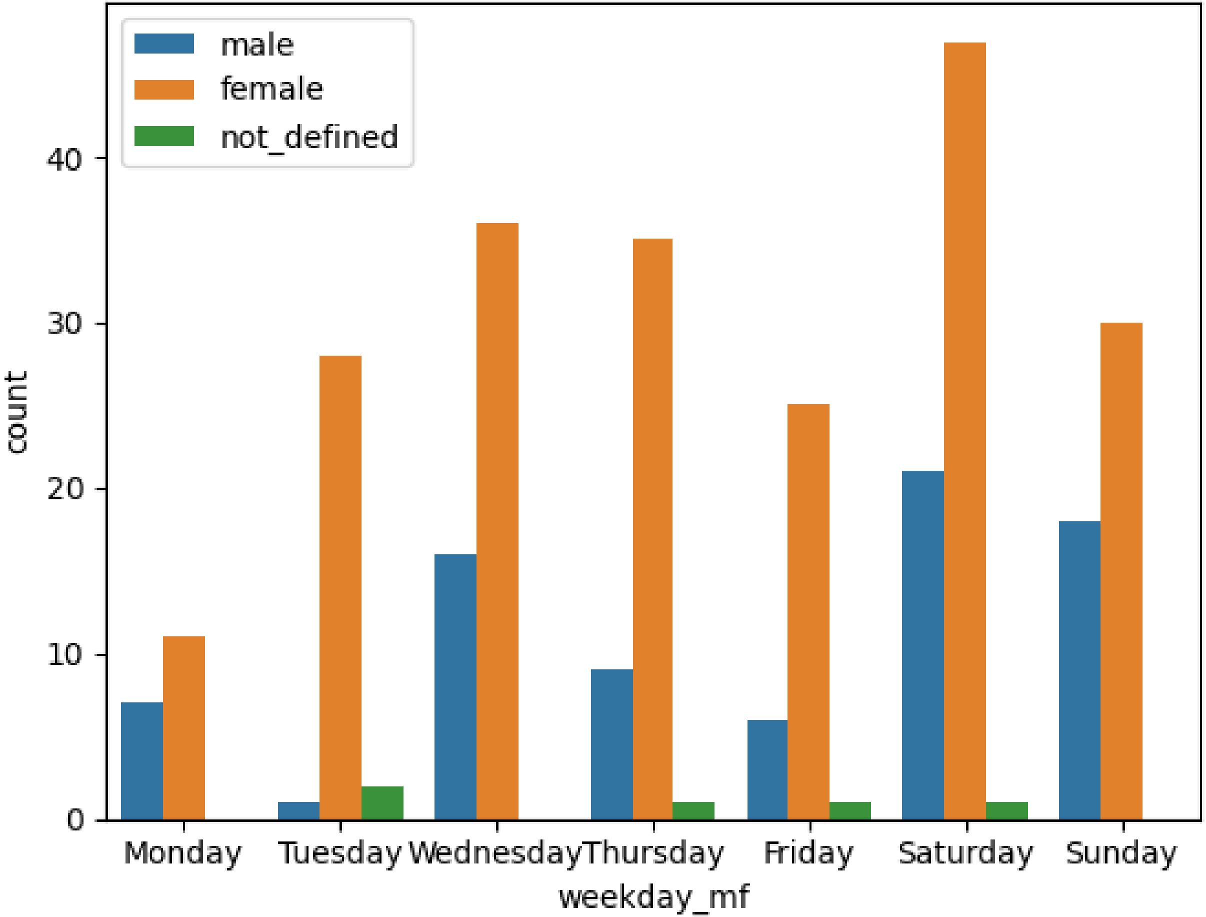
SPICED -Team / Teachers

- **Dina**
- Jens
- Rakib
- Carmine
- Maria
- ...

Bundesagentur für Arbeit (Funding)

- My familiy: Anja, Paul, Hannah and Luise ❤ ❤ ❤ ❤

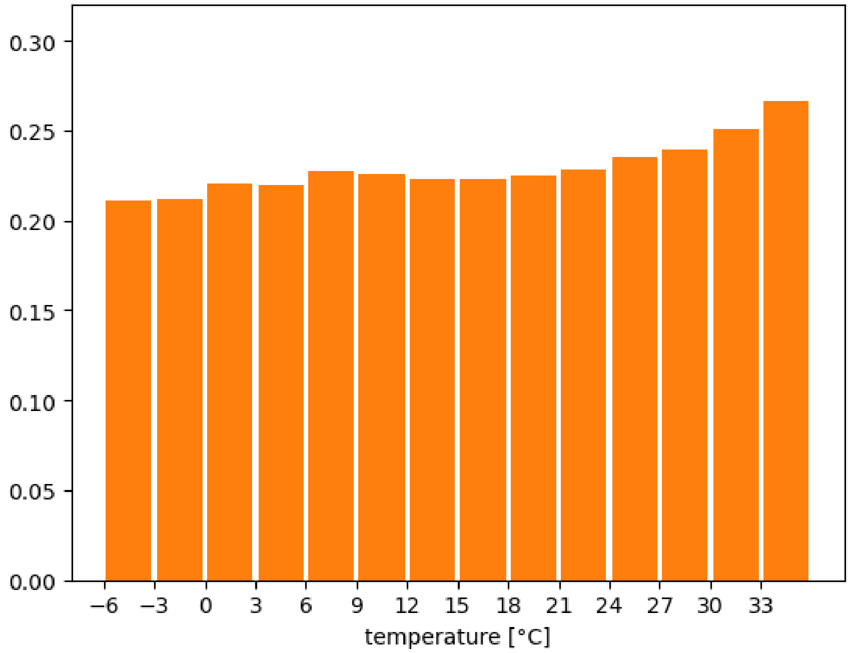
# Most frequent day of week for T=7.0 users



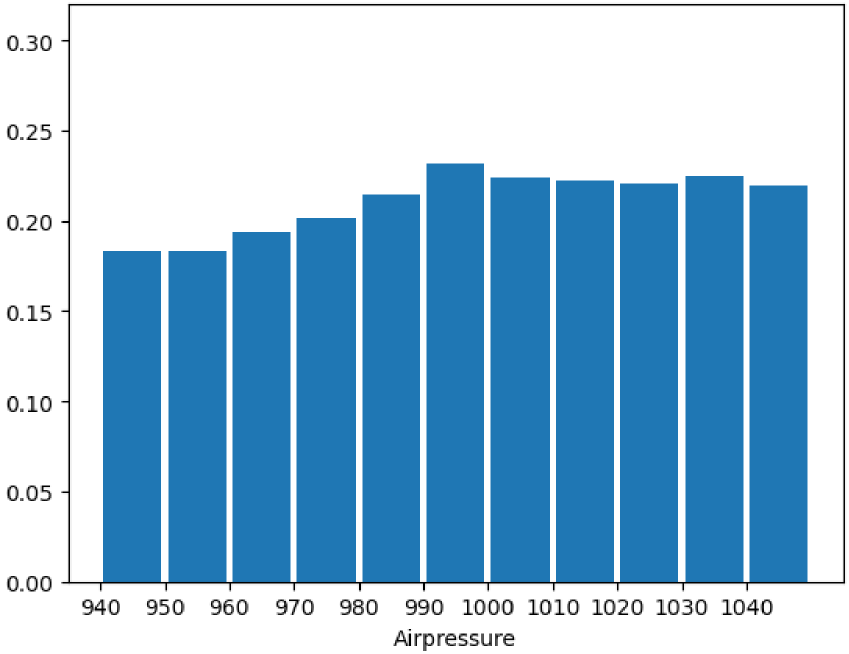
# Dependency on Weather Data

Normalized Distributions:  Fraction of data with headachday = 1

## Temperature



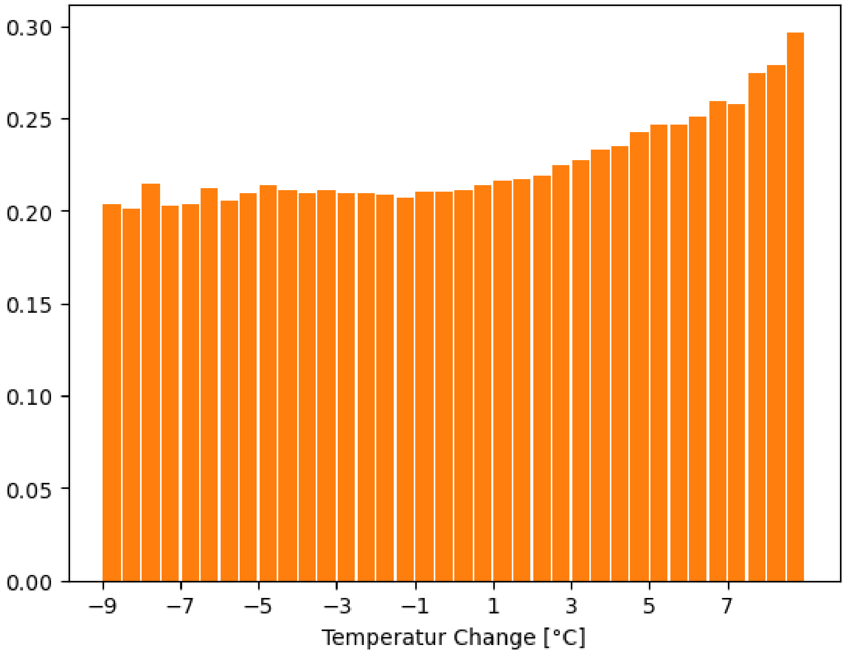
## Airpressure



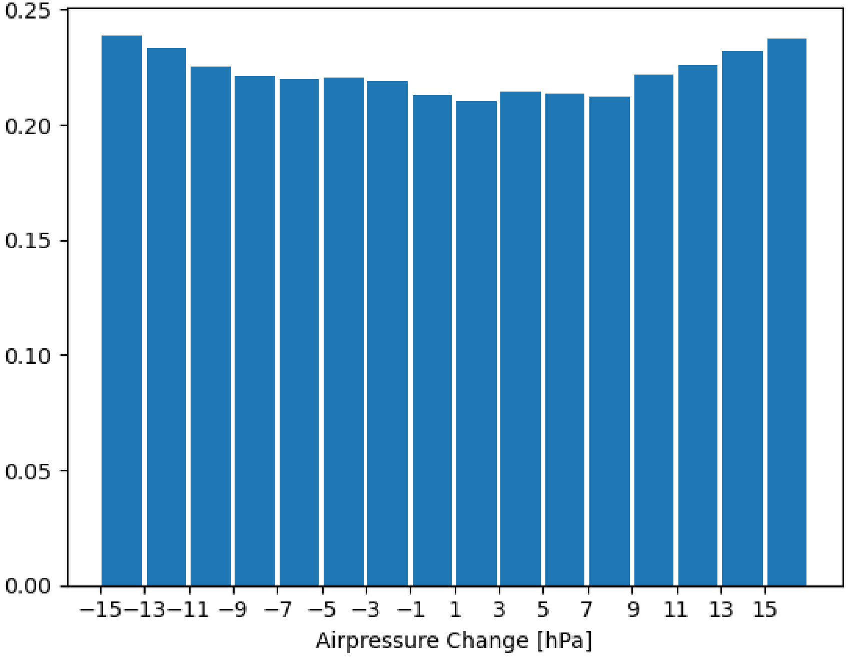
# Dependency on Weather Data

Normalized Distributions:  Fraction of data with headachday = 1

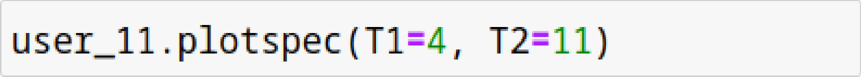
## Temperature Change



## Airpressure Change



## II. Working with “Big Data”

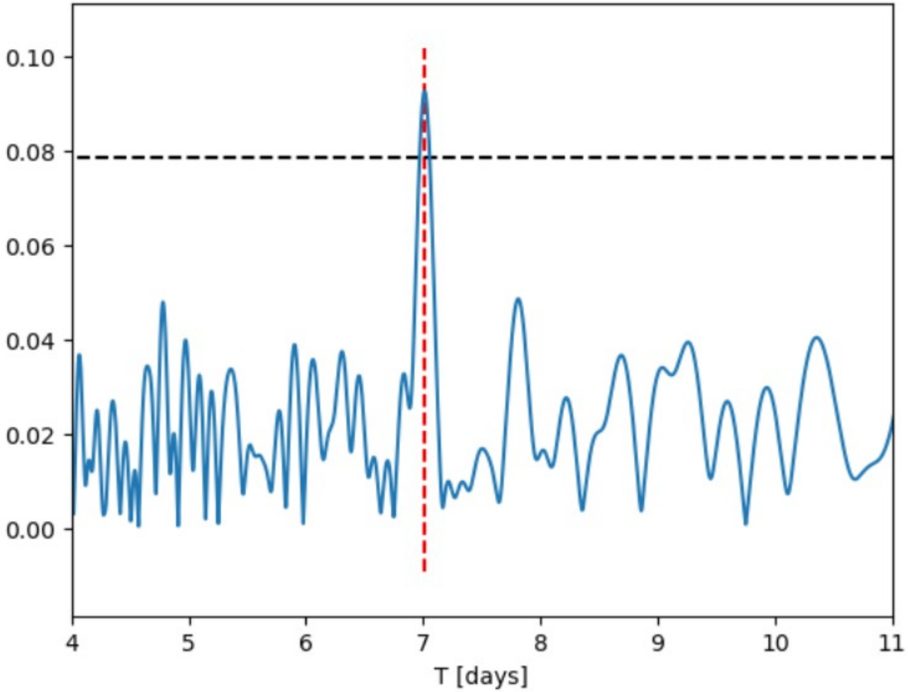


```
user_11.plotspec(T1=4, T2=11)
```

Access the data:

- Database shape complicated
- advanced SQL-Queries are needed to retrieve headache time series data

- Solution:
  **python *Class* NslUser.py**
  - comfortable access data using SQLAlchemy
  - apply operations


T [days]