Regression Challenge
for
MLC 16-17

Estimating circadian phase from reaction time

by Florian Wahl

Date: November 14, 2016
Supervisors: Prof. Dr. Oliver Amft, Florian Wahl

1 Introduction

Working from anywhere at any time and having a full social agenda makes it increasingly difficult to maintain a healthy sleep schedule. Many struggle with keeping constant circadian phase and thus have to rely on alarm clocks to wake up on time in the morning.

The circadian clock is our internal Zeitgeber controlling processes occurring within a 24 hour period. The phase of the clock is entrained through the intensity and timing of light. Optimally our internal clock and the external clock are in sync but unfortunately this does not hold for everyone, e.g. shift workers. Chronobiologists have found multiple negative effects, if our circadian clock is not in sync with the external world such as obesity [1], decreased alertness, impaired performance, gastrointestinal distress [2] and even cancer [3].

Our cognitive performance also follows a circadian rhythm [4, 5, 6] and can be used as a circadian phase marker. In this project you will use regression to estimate circadian phase based on data from reaction time test results.

Today 90% of young people wake up with their smartphones [7] which have become a popular companion in everyday life. Smartphones are used heavily throughout the day and can be used to test the user's reaction time throughout the day [8, 6]. In this project you use data recorded by 7 participants which recorded PVT data over multiple days in 2 hours intervals while being awake.

2 Challenge description

In this challenge, you will build a regression model to estimate circadian phase from reaction time tests in free living. The data was collected from 7 participants using the PVT-touch Android application[8]. During the PVT test, the participant has to click the screen as fast as possible when an image is displayed. The reaction time is recorded as the time between the image becoming visible and the user touching the screen. Each PVT test lasted 5 minutes during which ≈40 reaction time measurements were taken. Participants filled out a diary where alcohol consumption, caffeine consumption, nicotine consumption, food intake, sleep times, sport times, and medication intake were annotated with 15 minute resolution. In addition, participants stated whether they used an alarm clock to wake up and whether the day was a work or free day.

To obtain a baseline of each participants phase, first calculate the chronotype MSF_{SC} according to the journal sleep information [9]. A persons chronotype is calculated from sleep onset on work and free days (SO_W and SO_F, respectively) and sleep duration on

work and free days (SD_W and SD_F, respectively). N_w and N_F denote the number of work and free days in a 7 day period, respectively.

$$MSF = SO_F + \frac{SD_F}{2} \quad (1)$$

$$SD_{week} = \frac{SD_W \cdot N_w + SD_F \cdot N_F}{7} \quad (2)$$

$$MSF_{SC} = \left\{ \begin{array}{ll} MSF, & \text{if } SD_F \leq SD_W \\ MSF - \frac{SD_F - SD_{week}}{2}, & \text{if} SD_F > SD_W \end{array} \right\} \quad (3)$$

Keep in mind, not all participants recorded the data over consecutive days, so you might not be able to use all sleep data.

Next, you should explore the reaction time data to identify a good features for your regression model. Cajochen et al. [4] found that the slowest 10% reaction times were a good indicator of phase. You should then build a regression model to estimate the circadian phase. In our case, data was recorded in free living and the data is expected to be more noisy compared to recordings in a sleep laboratory.

For performance analysis, the root mean square error (RMSe) will be used. Because there might be a constant offset between MSF_{SC} and the output of your model, you are allowed to subtract/add an offset parameter (constant) before computing the RMSe. Between estimated phase e from PVT tests and calculated phase reference measurements r from MSF_{SC} calculations for all N measurements it is defined as:

$$RMSe = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(e_i - r_i)^2}.$$

The following steps are recommended:

1. Read the study information sheet to understand the recording protocol. Look at the paper journal to understand how ground truth information was collected.
2. Set up your working environment. Load the dataset, compute MSF_{SC} for each participant and extract features to run a basic model using either sklearn [10] or PyMC3 regression models.
3. Perform outlier detection and filtering on single measurements to improve the data quality. Because the data was recorded in free living, measurements are expected to be noisy compared to the work of Cajochen [4] who recorded their dataset in a sleep laboratory. Investigate whether you can identify typical error patterns and how to handle them (recovery through filtering/dropping single measurements/etc.)
4. Find interesting measurement times/points. Reaction times in certain parts of the day (e.g. evening) are expected to provide more insight, than others. Improve your model by integrating information about the relevance of measurement times (e.g. by weighting measurement times).

5. Model reaction time influencers, such as caffeine or alcohol consumption, to improve reduce RMSe. Alcohol consumption and food intake are known to slow down reaction times, while sleeping, caffeine and nicotine consumption are known to shorten reaction times [6]. Participants indicated when certain influences took place, thus you could model the expected effect of the influencers to improve your model accuracy.
6. Record additional data. Recording additional data may increase your understanding of the dataset. If you like to extend the dataset with your own recordings, please come to the first challenge consultation hour on Thursday, November 24 at 8am with your Android smartphone to enrol in the study.

Out of the points 3-6 feel free to pick 2-3 areas where you expect to improve the result. In the report, please reason why you picked those topics over others (why did you think this was the best option).

3 Results

Analyse the RMSe for each model you build. In addition, you can attempt to find a better measure for the prediction error. The function to calculate RMSe can be written in python as follows:

```
1  import numpy as np
2  def RMSe(e, r):
3      return np.sqrt(np.sum((((e - r)**2)))/len(e))
```

Document all your work and results in the report.

4 Data access

The dataset was recorded by 7 participants who carried an Android smartphone with the PVT touch application installed and a paper diary. The protocol is available on the ILIAS platform together with the sample paper diary. Participants were asked to follow the protocol for 10 consecutive days, but only few managed to comply.

4.1 PVT touch data

To load the raw data from PVT touch into Python you can use the following command.

```
1  import pandas as pd
2  raw_df = pd.read_hdf('dataset/data.h5', 'raw')
```

You will load a DataFrame where each line contains one test result from the PVT touch application.

1. timestamp: The time when the sample was taken.
2. foreperiod_start: The begin of
3. foreperiod_end: Beginning of the foreperiod. For a detailed explanation see the PVT touch documentation.
4. response_received: End of the foreperiod. For a detailed explanation see the PVT touch documentation.
5. subject: Participant id.
6. trial: Stimulus number within one test.
7. test: Test number.
8. requested_foreperiod: Randomly sampled moment when the image is shown within the foreperiod.
9. actual_foreperiod: Moment at which the image was actually displayed.
10. response_time: Time in ms that passed until the participant reacted. Negative on a false start (participant responds before stimulus is shown).
11. note: Automatic annotation by the app (see the manual for more information).
12. tag: Annotation by the user.
13. garbage_collection: Whether the app was garbage collected or not.

4.2 Diary information

```
1 import pandas as pd
2 labels_df = pd.read_hdf('dataset/data.h5', 'labels')
```

This will load a DataFrame with a row for every 15 minute interval and the following columns:

1. Alarmclock: Whether the participant was using an alarm clock that day.
2. Alcohol: Whether the participant was consuming alcohol in the time window of interest.
3. Caffeine: Whether the participant was consuming caffeine in the time window of interest.
4. Food: Whether the participant was consuming food in the time window of interest.
5. Medication: Whether the participant was taking medication in the time window of interest.

6. Nicotine: Whether the participant was consuming nicotine in the time window of interest.
7. Participant_ID: The participant ID.
8. Sleep: Whether the participant was asleep in the time window of interest.
9. Sports: Whether the participant was performing sports in the time window of interest.
10. Time: The beginning of the 15 minute time block of interest.
11. Workday: Whether this day was a workday.

5 Feature computation

The following code will compute some basic features for each PVT test to allow you a smooth start into the challenge.

```
import pandas as pd
raw_df = pd.read_hdf('dataset/data.h5', 'raw')

def get_quantile(data, q):
    """Takes series of values and returns quantile limit as well as the mean of the values above the quantile.
    data: Data as pandas Series.
    q: Quantile (0.75 -> 75%)
    returns: quantile limit, mean value of elements above quantile limit
    """

    quantile_limit = data.quantile(q=q)
    quantile_mean = data[data >= quantile_limit].mean()
    return quantile_limit, quantile_mean


def compute_features(test_df, verbose=False):
    """ Takes PVT test results and returns feature vector as a result.
    test_df: Dataframe containing PVT test results.
    Returns: Series containing the feature vector.
    """
    test_time = test_df.timestamp.iloc[0]
    n = test_df.shape[0]
    positive_data = test_df[test_df.response_time > 0] # drop all "too early samples"
    n_positive = positive_data.shape[0]
    positive_mean = positive_data.response_time.mean()
    positive_median = positive_data.response_time.median()
    positive_std = positive_data.response_time.std()
    q50_lim, q50_mean = get_quantile(positive_data.response_time, 0.50)
    q75_lim, q75_mean = get_quantile(positive_data.response_time, 0.75)
    q90_lim, q90_mean = get_quantile(positive_data.response_time, 0.90)
    q95_lim, q95_mean = get_quantile(positive_data.response_time, 0.95)
    features = pd.Series({'Test_time' : test_time,
                          'Subject' : test_df.subject.iloc[0],
                          'Test_nr' : test_df.test.iloc[0],
                         'n_total' : n,
                         'n_positive' : n_positive,
                         'positive_mean' : positive_mean,
                         'positive_median' : positive_median,
```

```
                        'positive_std' : positive_std,
                        'q50_lim' : q50_lim,
                        'q75_lim' : q75_lim,
                        'q90_lim' : q90_lim,
                        'q95_lim' : q95_lim,
                        'q50_mean' : q50_mean,
                        'q75_mean' : q75_mean,
                        'q90_mean' : q90_mean,
                        'q95_mean' : q95_mean})
    if verbose:
        print(features)
    return features

for subject_id, subject_df in raw_df.groupby(raw_df.subject):
    for test_id, test_df in subject_df.groupby(subject_df.test):
        feature_df = feature_df.append(compute_features(test_df), ignore_index=True)
feature_df.reset_index(inplace=True, drop=True)

# Compute the time of day as a float
h = feature_df.Test_time.apply(lambda x: x.hour)
m = feature_df.Test_time.apply(lambda x: x.minute)
feature_df['time_of_day'] = h + (m/60.0)
```

This will result in a DataFrame with one entry per test and the following columns:

1. Subject: Participant id.
2. Test_nr: Test id.
3. Test_time: Timestamp of the test.
4. n_positive: Number of samples in test with positive reaction time.
5. n_total: Number of samples in test.
6. positive_mean: Average reaction time of all samples with positive reaction time.
7. positive_median: Median reaction time of all samples with positive ration time.
8. positive_std: Standard deviation of all samples with positive ration time.
9. q50_lim: Limit of the 50% quantile after dropping all samples with negative response time.
10. q50_mean: Mean of the 50% quantile after dropping all samples with negative response time.
11. q75_lim: Limit of the 75% quantile after dropping all samples with negative response time.
12. q75_mean: Mean of the 75% quantile after dropping all samples with negative response time.
13. q90_lim: Limit of the 90% quantile after dropping all samples with negative response time.

14. q90_mean: Mean of the 90% quantile after dropping all samples with negative response time.
15. q95_lim: Limit of the 95% quantile after dropping all samples with negative response time.
16. q95_mean: Mean of the 95% quantile after dropping all samples with negative response time.
17. time_of_day: The test time as a float.

6 Administrative aspects

6.1 Working in student groups

The challenge shall be approached by teams of 3-4 students. Please address this challenge in the same group as the previous challenge. Discuss your approach in the team, work in parallel, where useful and consolidate your findings in the report.

6.2 Relevant literature material

References for initial literature are provided in this project description. If needed, additional literature should be searched and considered independently. It is strongly recommended to keep a literature database and add comments on conclusions of individual papers, e.g. using JabRef.

For understanding how PyMC works, the blog found at `http://twiecki.github.io/` is a good start. If you like to read more about regression, the book of Gelman and Hill [11] is a good source.

6.3 Software tools

The following list of software packages are useful to solve the challenges, but not mandatory to use. The code snippets presented in this guide utilise these packages.

- IPython [12].
- Pandas [13].
- numpy [14].
- Scikit-Learn `http://scikit-learn.org/stable/tutorial/basic/tutorial.html` [10].

- PyMC3 `https://pymc-devs.github.io/pymc3/getting_started/`.

6.4 Challenge advisor consultations

Team meetings with the challenge advisor shall be planned to discuss achieved work results, define next steps. The team is asked to attend with all members and *thoroughly prepare* for the meeting, including presenting the work already performed and proposed next steps. Meeting have to be scheduled in advance using `http://doodle.com/fwahl`. Walk-in consultation is discouraged.

6.5 Project documentation, presentations, and report

It is recommended to keep an adequate record (e.g. in a draft report) of project results, conclusions, and decisions made. This information can be used to prepare the final report and the presentation.

A report should be submitted by or before the deadline. See Section 6.6 for details on the presentation dates. The report shall describe the team's effort in the challenge (prepared using LaTeX, in English language). The report should be max. 5 pages long, plus cover page. A report template will be provided on Stud.IP. The challenge report shall be accompanied by an electronic appendix (e.g., zip file), containing all literature used, code developed, or otherwise acquired during the challenge. It should be possible to reproduce your results with the code. A brief documentation shall be added to navigate in the electronic appendix.

Assessment criteria for the report are (1) quality (specifically including the following aspects: hypotheses, rationale, analysis, conclusions), (2) clarity & systematic working style, (3) quantity of results.

At the end of the course, teams are invited to give a presentation of their results for a selected challenge. Which team will present the challenge as well as detailed instructions on the presentation will announced after all challenges are completed. The presentation shall illustrate the method used, the strong and weak points of the approach, the results obtained, and conclusions.

6.6 Important dates for this challenge

- The Challenge starts as the task description is made available.
- Challenge report and electronic appendix is due by Dec 4, 2016. Please upload the report and the electronic appendix to ILIAS. No submission will be accepted after the deadline.

- The final presentation will be on Feb 08, 2017. The presenting teams for this challenge will be announced once all the challenges are completed.

References

[1] T. Roenneberg, K. V. Allebrandt, M. Merrow, and C. Vetter, "Social Jetlag and Obesity," *Current Biology*, vol. 22, no. 10, pp. 939–943, May 2012.

[2] V. L. Revell and C. I. Eastman, "How to trick mother nature into letting you fly around or stay up all night." *J Biol Rhythms*, vol. 20, no. 4, pp. 353–365, Aug. 2005.

[3] T. Kantermann, M. Juda, C. Vetter, and T. Roenneberg, "Shift-work research: Where do we stand, where should we go?" *Sleep and Biological Rhythms*, vol. 8, no. 2, pp. 95–105, Apr. 2010.

[4] C. Cajochen, S. B. S. Khalsa, J. K. Wyatt, C. A. Czeisler, and D.-J. Dijk, "EEG and ocular correlates of circadian melatonin phase and human performance decrements during sleep loss," *American Journal of Physiology - Regulatory, Integrative and Comparative Physiology*, vol. 277, no. 3, pp. R640–R649, Sep. 1999.

[5] K. Blatter and C. Cajochen, "Circadian rhythms in cognitive performance: Methodological constraints, protocols, theoretical underpinnings," *Physiology & behavior*, vol. 90, no. 2, pp. 196–208, 2007.

[6] S. Abdullah, E. L. Murnane, M. Matthews, M. Kay, J. A. Kientz, G. Gay, and T. Choudhury, "Cognitive Rhythms: Unobtrusive and Continuous Sensing of Alertness Using a Mobile Phone," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '16. New York, NY, USA: ACM, 2016, pp. 178–189.

[7] Mary Gorges, "90 percent of young people wake up with their smartphones," http://www.ragan.com/Main/Articles/45989.aspx.

[8] M. Kay, K. Rector, S. Consolvo, B. Greenstein, J. O. Wobbrock, N. F. Watson, and J. A. Kientz, "PVT-touch: Adapting a reaction time test for touchscreen devices," in *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, May 2013, pp. 248–251.

[9] The WeP, "MCTQ variables," http://www.thewep.org/documentations/mctq/item/mctq-variables.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] A. Gelman and J. Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2006.

[12] F. Perez and B. E. Granger, "IPython: A System for Interactive Scientific Computing," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 21–29, 2007.

[13] W. McKinney, "Pandas: A Foundational Python Library for Data Analysis and Statistics," *Python for High Performance and Scientific Computing*, pp. 1–9, 2011.

[14] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011.