Sleep and Exercise Difficulty

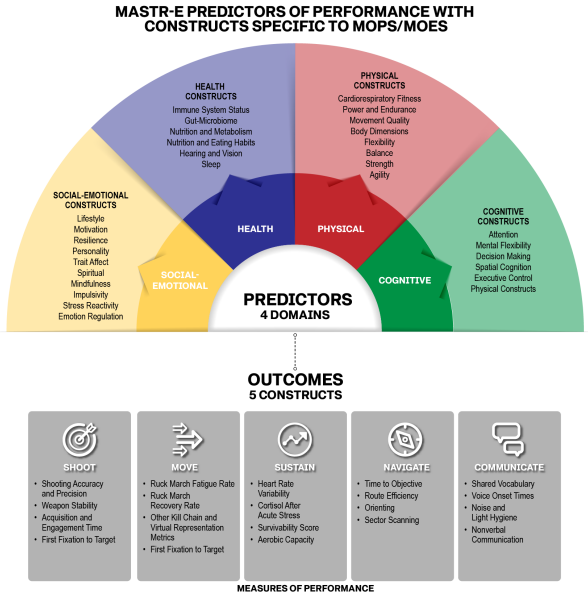
Author: Addy Moran

Abstract

Investment into personalized training for tactical forces have been a priority for the US military in recent years, focusing on a variety of mental and physical health "predictors" to optimize "outcomes" such as exercise sustainment, decision making, and communication. This analysis focuses on a single-subject dataset in a non-tactical environment to see if there is a correlation between various sleep metrics and exercise difficulty, by both quantitative data (max heart rate and workout duration) and qualitative (perceived difficulty).

Introduction

In recent years, the Department of Defense has invested in "data-driven personalization of training and readiness" [6]. The Army's Special Warfare Center and School (SWCS)'s Measuring and Advancing Soldier Tactical Readiness and Effectiveness (MASTER-E) program is one of the most well-funded research initiatives in history by the Department of Defense (DoD). It describes four predictor classes: social-emotional, health, physical, and cognitive; and five outcome classes: shoot, move, sustain, navigate, and communicate. [6]



MASTR-E Predictors to Outcomes [6]

In [7], George Matook asks if the unit got poor sleep, should it be a rest day or a ruck day for the unit? The analysis completed in this paper, was designed around this question.

Some of the variables explored in this analysis are:

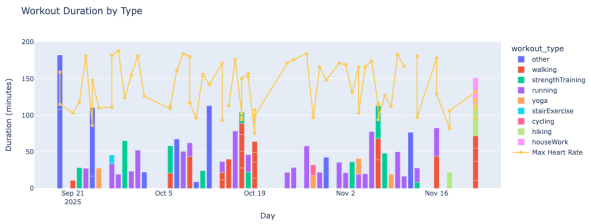
- Heart Rate Variability (HRV) and exercise intensity (by max heart rate)
- HRV and exercise duration
- HRV and exercise 'difficulty'
- Rapid Eye Movement (REM) duration and exercise 'difficulty'
- Deep sleep duration and exercise 'difficulty'
- Light sleep duration and exercise 'difficulty'
- Total sleep duration and exercise 'difficulty'

Published research has concluded that the higher one's HRV is, the more resilient they are to stress and are generally physically and mentally healthier. HRV may decrease due to stress, anxiety, or short and long term sicknesses. [8]

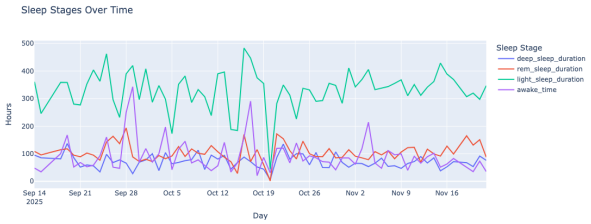
There is a significant correlation between regular exercise and improvements to sleep [1] however, much less research was found on the impact of one's sleep to the next day's exercise.

Methods

The data used in this analysis was collected using the Oura Generation 4 Ring, from 14 September 2025 through 22 November 2025. The ring was worn by the same person, on their dominant hand, on the same finger, for the duration of the collection. There are several gaps in the data due to the ring charging. Throughout this time period there were 82 exercise sessions recorded across 47 days, varying from yoga, (indoor and outdoor) running, rucking (labeled as "other" in the data), housework, and indoor biking.



Workout Duration by Workout Type



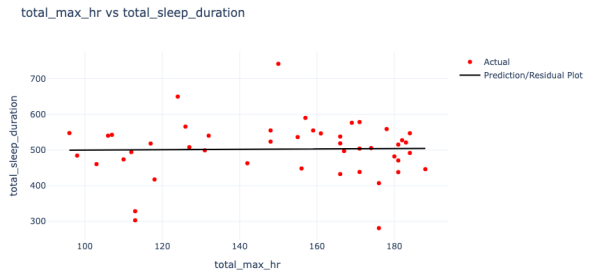
Overview of Sleep Stages

Sleep vs HRV

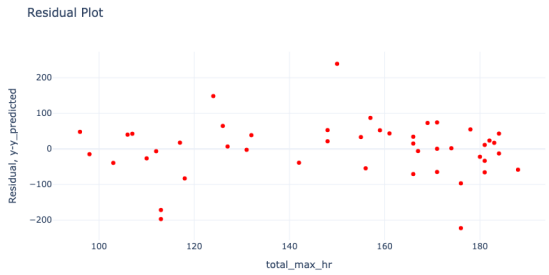
Residual plots were created to test for independence, and for the data available:

- Sleep HRV was independent of light sleep duration, deep sleep duration, and REM sleep duration.
- Max workout HR was independent of total sleep duration, REM sleep duration, light sleep duration, and sleep HRV.
- Total workout duration was independent of total sleep duration, REM sleep duration, light sleep duration, and sleep HRV.

The figures below show one of those residual plots.



Total Max HR vs Total Sleep Duration



Total Max HR vs Total Sleep Duration, Residual

All of the data used by the Oura was quantitative, however, some of the augmented fields, such as the workout day label, was qualitative.

Data Overview

Field Name	**Source**	**Notes**	**Units**
day	Oura	Day of exercise	
total_max_hr	Augmented	Max HR across all workout sessions that day	BPM
total_duration	Augmented	Summed duration for all workout sessions	min
workout_type	Oura	Most intensive workout	
intensity_label	Augmented	Low, Medium, High	
deep_sleep_duration	Oura	Deep sleep duration from previous night	min
rem_sleep_duration	Oura	REM sleep duration from previous night	min
light_sleep_duration	Oura	Light sleep duration from previous night	min
sleep_hrv	Oura	HRV from the night before	ms
sleep_breath	Oura	Average breaths per minute from the night before	Breath per min
total_sleep_duration	Oura	Total sleep duration from night before	min

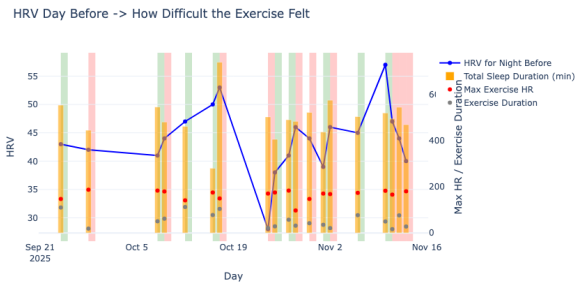
total_sleep_duration_hr	Augmented	Total sleep duration in hours (total_sleep_duration *60)*	hr
awake_time	Oura	Minutes awake from the night before	min
day_label	Manual	Good, Unremarkable, Hard (more details on what distinguishes these below)	
day_label_num	Manual	-1 if day_label is Hard 0 if day_label is Unremarkable 1 if day_label is Good	
rem_sleep_duration_qbin	Augmented	Bin 1: 28.5 to 87.2 minutes Bin 2: 87.2 to 96.0 minutes Bin 3: 96.0 to 114.8 minutes Bin 4: 114.8 to 193.5 minutes	
light_sleep_duration_qbin	Augmented	Bin 1: 173.0 to 294.5 minutes Bin 2: 294.5 to 342.5 minutes Bin 3: 342.5 to 372.5 minutes Bin 4: 372.5 to 483.5 minutes	
deep_sleep_duration_qbin	Augmented	Bin 1: 27.5 to 53.8 minutes Bin 2: 53.8 to 68.0 minutes Bin 3: 68.0 to 79.5 minutes Bin 4: 79.5 to 113.0 minutes	
total_sleep_duration_qbin	Augmented	Bin 1: 281.0 to 466.8 minutes Bin 2: 466.8 to 515.0 minutes Bin 3: 515.0 to 546.2 minutes Bin 4: 546.2 to 741.5 minutes	

sleep-_hrv_qbin	Augmented	Bin 1: 28.0 to 40.0 Bin 2: 40.0 to 43.0 Bin 3: 43.0 to 46.0 Bin 4: 46.0 to 57.0	
sleep-_breath_qbin	Augmented	Bin 1: 14.4 to 15.0 Bin 2: 15.0 to 15.4 Bin 3: 15.4 to 15.6 Bin 4: 15.6 to 16.8	
awake_-time_qbin	Augmented	Bin 1: 20.6 to 57.9 minutes Bin 2: 57.9 to 73.1 minutes Bin 3: 73.1 to 105.9 minutes Bin 4: 105.9 to 343.1 minutes	
deep_sleep-_duration_percentage	Augmented	Deep sleep / total sleep duration	percent
deep_sleep-_duration_is_ideal	Augmented	-1 if deep sleep duration is less than 23% 0 if deep sleep duration is between 23% and 27% 1 if deep sleep duration is more than 27% [4]	
light_sleep-_duration_percentage	Augmented	Light sleep / total sleep duration	percent
light_sleep-_duration_is_ideal	Augmented	-1 if light sleep duration is less than 48% 0 if light sleep duration is between 48% and 52% 1 if light sleep duration is more than 52% [4]	
rem_sleep-_duration_percentage	Augmented	REM sleep / total sleep duration	percent

rem_sleep_duration_is_ideal	Augmented	-1 if REM sleep duration is less than 23% 0 if REM sleep duration is between 23% and 27% 1 if REM sleep duration is more than 27% [4]	
total_sleep_duration_is_ideal	Augmented	-1 if deep sleep duration is less than 7 hours 0 if deep sleep duration is between 7 and 9 hours 1 if deep sleep duration is more than 9 hours [4]	

The null/NA policy was to drop (for example, if sleep data was not available for the day before the exercise).

The day_label field was augmented using a workout journal where the activity and distance (if applicable) were logged, some days felt harder than others (i.e. if a 1 mile run at the same pace as normal and it felt more difficult than usual) while others felt easier and the subject was able to push harder than normal (running longer, faster, more incline, etc.). For the days that felt notably harder than usual, the day_label is "Hard", for the days that the subject felt the same exercise was easier than normal or was able to push their limits more than they have done previously, the day_label is "Good", any other day is considered "Unremarkable".



Both the raw data and the augmented data can be found here https://github.com/addymmoran/data/tree/main/oura/09-2025_11-2025.

Limitations & Assumptions

This analysis assumes that each day is independent of the previous, for example, that a run on day 1 does not impact how "difficult" a run was on the next day. There are many factors that may impact sleep HRV, total sleep duration, individual sleep stage durations, or how "easy"/"hard" an exercise may be/feel. For example, sleep HRV may lower due to alcohol or stress [9] and the menstrual cycle may impact the subject's perceived exercise difficulty [5]. This analysis is trying to determine if, regardless of short-term sickness, life-stress, or other factors, sleep metrics (sleep stage durations, sleep HRV) can be used to predict exercise difficulty (by max HR, workout duration, or perceived difficulty).

The augmented data, specifically the "Day Label" fields are very subjective, there is discussion on how to improve this in the "Future Work" section.

To summarize the limitations of this study:

- Single-subject design
- Assumed day-to-day independence
- Subjective variables
- Small N
- Uncontrolled confounders

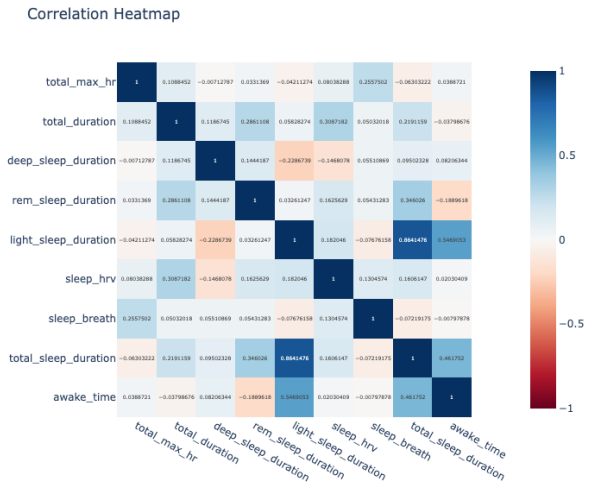
- Non-independence of sleep stages

Results and Interpretation

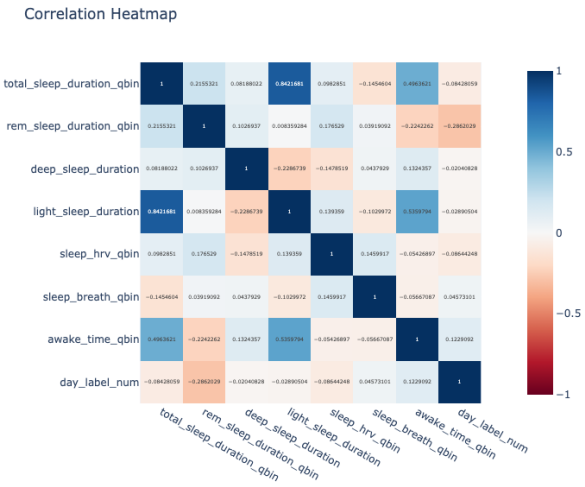
Correlation

The sleep and exercise difficulty metrics are non-parametric, so Spearman's correlation was used for the initial investigation.

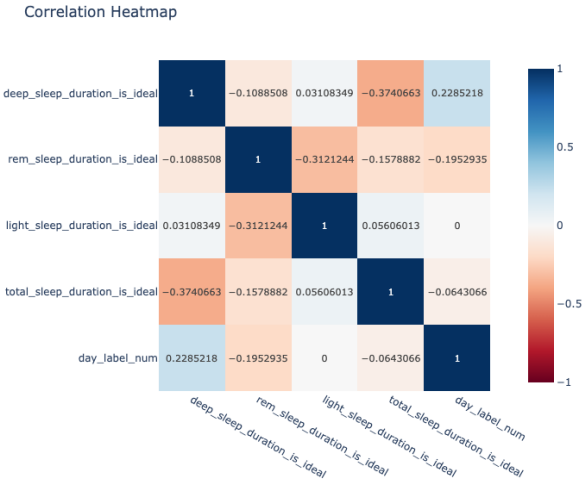
At first glance, the day label appears to correlate with the amount of REM sleep and time awake:



Binning on the most correlated variables confirms that REM sleep is the primary driver for day label differences, with a negative correlation. Interestingly, this indicates that nights with less REM sleep are associated with "better" days—contrary to typical expectations.



Further analysis using "ideal" sleep thresholds (as defined in [4]) showed that having enough deep sleep (>23%) but less-than-ideal REM sleep (<23%) resulted in the best days:



Bayes

A subjective intensity field was added to augment the dataset, binning activities by the level of effort (see table below) and assuming independence between sleep stages.

Intensity	Exercise Type
Low	Yoga, Walking, House Work

Medium	Strength Training, Hiking, Cycling, Stair Exercise
High	Running, Rucking

The prior was based on the frequency table below.

Intensity	**Count**
Low	33
Medium	16
High	26
Total	47

Both the data and code used to calculate these probabilities can be found in the appendix.

Single-Sleep Cycle Bayes

Using Bayes' theorem, with the prior based on overall workout frequency, we can estimate the probability of each intensity based on sleep quality (Too Little, Ideal, Too Much):

The goal is to determine whether the intensity of a workout can be predicted given a "poor" night of sleep. For example: $P(\text{Intensity} = \text{High} | \text{Deep} = \text{Light})$

$$P(\text{Intensity} = i \mid \text{Deep} = d) = \frac{P(\text{Deep} = d \mid \text{Intensity} = i) \cdot P(\text{Intensity} = i)}{\sum_{j \in \{\text{Low, Medium, High}\}} P(\text{Deep} = d \mid \text{Intensity} = j) \cdot P(\text{Intensity} = j)}$$

$$P(\text{Intensity} = i \mid \text{REM} = r) = \frac{P(\text{REM} = r \mid \text{Intensity} = i) \cdot P(\text{Intensity} = i)}{\sum_{j \in \{\text{Low, Medium, High}\}} P(\text{REM} = r \mid \text{Intensity} = j) \cdot P(\text{Intensity} = j)}$$

$$P(\text{Intensity} = i \mid \text{Light} = l) = \frac{P(\text{Light} = l \mid \text{Intensity} = i) \cdot P(\text{Intensity} = i)}{\sum_{j \in \{\text{Low, Medium, High}\}} P(\text{Light} = l \mid \text{Intensity} = j) \cdot P(\text{Intensity} = j)}$$

The code block below is the computation of Bayes:

```
# P( intensity_label | deep_sleep == Too Little )

bayes_all = []

for combo in counts[['sleep_category', 'ideal_value']].drop_duplicates().to_dict(orient='records'):

    subset = counts[
        (counts.sleep_category == combo['sleep_category']) &
        (counts.ideal_value == combo['ideal_value'])
    ]

    rows = []

    for intensity in subset.itertuples():
        bayes = {
            'intensity_label': intensity.intensity_label,
            'sleep_category': intensity.sleep_category,
            'sleep_quality': intensity.ideal_value
        }
        # Prior: global intensity distribution
        bayes['prior'] = intensity_counts[intensity.intensity_label] / sum(intensity_counts.values())

        # Likelihood: P(sleep_quality | intensity)
        bayes['likelihood'] = intensity.count / intensity_counts[intensity.intensity_label]

        # Unnormalized posterior
        bayes['unnormalized_posterior'] = bayes['prior'] * bayes['likelihood']

        rows.append(bayes)

    # Convert rows for this sleep category to a DataFrame
    df_tmp = pd.DataFrame(rows)

    # Normalize within THIS sleep category/value combo
    df_tmp['normalized_posterior'] = (
        df_tmp['unnormalized_posterior'] /
```

```
        df_tmp['unnormalized_posterior'].sum()
    )

    bayes_all.append(df_tmp)

# Concatenate all results
bayes_simple = pd.concat(bayes_all, ignore_index=True)
```

This produces posterior probabilities for individual sleep components:

Intensity	Sleep Category	Quality	Posterior
High	Deep Sleep	Ideal	1.0
High	Deep Sleep	Too Little	0.644
Low	Deep Sleep	Too Little	0.111
Medium	Deep Sleep	Too Little	0.244
High	Light Sleep	Ideal	1.0
High	Light Sleep	Too Much	0.652
Low	Light Sleep	Too Much	0.1087
Medium	Light Sleep	Too Much	0.240
High	REM Sleep	Ideal	0.75
Low	REM Sleep	Ideal	0.25
High	REM Sleep	Too Little	0.676
Low	REM Sleep	Too Little	0.108
Medium	REM Sleep	Too Little	0.216
High	REM Sleep	Too Much	0.5
Medium	REM Sleep	Too Much	0.5

While this shows which individual sleep components impact intensity, a consolidated view considering all sleep stages provides a clearer picture of the recommended workout intensity for a poor night of sleep.

$$P(\text{Intensity} = i \mid \text{REM} = r, \text{Light} = l, \text{Deep} = d) \propto \frac{P(\text{REM} = r \mid \text{Intensity} = i) \cdot P(\text{Light} = l \mid \text{Intensity} = i) \cdot P(\text{Deep} = d \mid \text{Intensity} = i) \cdot P(\text{Intensity} = i)}{\sum_{j \in \{\text{Low, Medium, High}\}} P(\text{REM} = r \mid \text{Intensity} = j) \cdot P(\text{Light} = l \mid \text{Intensity} = j) \cdot P(\text{Deep} = d \mid \text{Intensity} = j) \cdot P(\text{Intensity} = j)}$$

The code block below is the computation of Bayes:

```
# Bayes, P( intensity level | Rem = Ideal, Light = Ideal, Deep Sleep = Too Little)

prior_strength = 5  # adjust based on confidence in prior

combinations = df_counts[["Rem Sleep Quality", "Light Sleep Quality", "Deep Sleep Quality"]].drop_duplicates().to_dict(orient='records')

n_all_workouts = df_counts['count'].sum()

# prior_strength * p of each intensity happening (regardless of sleep qualities)
alpha_priors = {
    'high_intensity': prior_strength * intensity_counts['high_intensity']/n_all_workouts,
    'medium_intensity': prior_strength * intensity_counts['medium_intensity']/n_all_workouts,
    'low_intensity': prior_strength * intensity_counts['low_intensity']/n_all_workouts
}

# Posterior
bayes_all = []
for combo in combinations:
    df = df_counts[ # will get all intensity labels with this sleep quality spread
        (df_counts['Rem Sleep Quality'] == combo['Rem Sleep Quality']) &
        (df_counts['Light Sleep Quality'] == combo['Light Sleep Quality']) &
        (df_counts['Deep Sleep Quality'] == combo['Deep Sleep Quality'])
    ][['intensity_label', 'count']].groupby('intensity_label').sum().reset_index()

    bayes = {
        'total_alpha_post': 0
    }

    bayes.update(combo)

    for intensity_label in intensity_counts.keys():
```

```
        num_intensity_seen_in_df =
df[df['intensity_label'] == intensity_label]
['count'].item() if not df[df['intensity_la-
bel'] == intensity_label].empty else 0
        bayes[f'alpha_posteriors_parame-
ter_{intensity_label}'] = alpha_priors[in-
tensity_label] + num_intensity_seen_in_df
        bayes['total_alpha_post'] +=
bayes[f'alpha_posteriors_parameter_{intensi-
ty_label}']

    # have to run seperately so we have the
total alpha post
    for intensity_label in intensity_-
counts.keys():
        bayes[f'posterior_p_{intensity_la-
bel}'] = bayes[f'alpha_posteriors_parame-
ter_{intensity_label}'] / bayes['total_al-
pha_post']

    bayes_all.append(bayes)

pd.DataFrame(bayes_all)[['Rem Sleep Quali-
ty', 'Light Sleep Quality', 'Deep Sleep
Quality', 'posterior_p_low_intensity', 'pos-
terior_p_medium_intensity', 'posteri-
or_p_high_intensity']].set_index(['Rem Sleep
Quality', 'Light Sleep Quality', 'Deep Sleep
Quality'])
```

This produces posterior probabilities for consolidated sleep components:

REM Duration	**Light Duration**	**Deep Duration**	**P(Low Intensity)**	**P(Medium Intensity)**	**P(High Intensity)**
Too Little	Too Much	Too Little	0.157	0.226	0.617
Too Little	Too Much	Ideal	0.277	0.171	0.552
Ideal	Too Much	Too Little	0.314	0.142	0.544
Too Much	Ideal	Too Little	0.308	0.190	0.502
Too Much	Too Much	Too Little	0.213	0.362	0.425

Comparison

As discussed earlier, there is relatively limited research focused specifically on how sleep quality from the previous night influences exercise performance the following day. Existing evidence, such as 3, indicates that exercising after a poor night of sleep can still offer long-term benefits, including improved energy, better sleep the following night, and support for maintaining consistent routines.

However, in scenarios where maintaining a strict routine is not the primary objective—or where sleep duration and quality are inherently unpredictable, such as during mission-driven operations—it may be more practical to rely on personal historical data to guide decisions. Reviewing one's own patterns can help determine when a rest day, lower-intensity session, or higher-intensity workout is likely to be most effective given the previous night's sleep.

Summary

Of the various sleep metrics, there was very little correlation to the max workout heart rate or to the workout duration. However, looking at the perceived difficulty showed a positive deep sleep duration and negative REM sleep duration correlation. Applying this to the original question, it is fair to say that exercising on days where the user gets between 23 and 27% deep and REM sleep and more than 27% light sleep result in most productive workouts (based on the ability to push harder), and days where deep and REM sleep are more than 27% and light sleep is less than 23%, rest days or light workout days should be considered.

Future Work

To improve accuracy of the predictions discussed in this paper, additional work needs to focus on:

- increasing the number of days worth of data will help ensure the outcomes are accurate as with only 47 days worth of data, the findings reported in this report need to be taken lightly.
- Implement the Perceived Exertion Scale [2] in the workout journal to remove the subjectivity.

As part of the subject's workout journal, knee pain is also recorded (along with how long it lasts and level of pain), future work could include looking for indicators of better/worse knee pain given sleep metrics or duration of previous workouts (potentially determining a "recovery" score).

Acknowledgments

ChatGPT 5.0 was used to help with data ingestion/processing, approach validations, and editing.

References

[1] Alnawwar MA, Alraddadi MI, Algethmi RA, Salem GA, Salem MA, Alharbi AA. The Effect of Physical Activity on Sleep Quality and Sleep Disorder: A Systematic Review. Cureus. 2023 Aug 16;15(8):e43595. doi: 10.7759/cureus.43595. PMID: 37719583; PMCID: PMC10503965.

[2] Borg, G. A. V. (1970). Perceived Exertion Scale (RPE) Database record. APA PsycTests. https://doi.org/10.1037/t58166-000

[3] Bradbury, A. (2024, September 26). *Should you workout after a terrible night of sleep? Here's the science-backed answer*. EnduraLAB. https://www.enduralab.com/should-you-workout-after-a-terrible-night-of-sleep-heres-the-science-backed-answer/ (enduralab.com)

[4] Cleveland Clinic. (n.d.). *Sleep: What it is, why it's important, stages, REM & NREM*. Cleveland Clinic. https://my.clevelandclinic.org/health/body/12148-sleep-basics (my.clevelandclinic.org)

[5] Harvard T.H. Chan School of Public Health. (2025, May). *Exploring exercise habits by menstrual cycle phase*. Apple Women's Health Study. https://hsph.harvard.edu/research/apple-womens-health-study/study-updates/exploring-exercise-habits-by-menstrual-cycle-phase/ (hsph.harvard.edu)

[6] Kwon, P. O., Zientara, G. P., Thota, D. S., Matook, G., Davis, W. T., Pike, W. Y., Brager, A. J., & Pamplin, J. C. (2025, May 1). *Digital twins for a digital world: Data-driven training optimizing the ready medical force*. Special Warfare Journal. Retrieved from https://www.swcs.mil/Special-Warfare-Journal/Article/4170157/digital-twins-for-a-digital-world-data-driven-training-optimizing-the-ready-med/ (swcs.mil)

[7] South, T. (2023, April 28). *Army fielding new tactical feedback tool with fitness program features. Army Times.* Retrieved from https://www.armytimes.com/news/your-army/2023/04/28/army-fielding-new-tactical-feedback-tool-with-fitness-program-features/ (armytimes.com)

[8] Sydney Pelvic Clinic. (n.d.). *Heart rate variability – why it's important and how to train it*. Retrieved from https://www.sydneypelvicclinic.com.au/heart-rate-variability-why-its-important-and-how-to-train-it (sydneypelvicclinic.com.au)

[9] Yates, V. P. (2023, August 8). *What causes low HRV during sleep?* The Pulse Blog. Oura. https://ouraring.com/blog/low-hrv-while-sleeping/ (ouraring.com)

Appendices

Appendix A: Data

Table 1: Analysis Fields

Field Description	mean	std	min	max
Total duration of activity in minutes	51.338997	36.859206	8.977600	182.230950
Minutes spent in deep sleep	67.851064	18.755554	27.500000	113.000000
Minutes spent in REM sleep	102.308511	28.581606	28.500000	193.500000
Minutes spent in light sleep	333.010638	68.054685	173.000000	483.500000
Heart rate variability during sleep	43.127660	5.198733	28.000000	57.000000
Average breaths per minute during sleep	15.337766	0.422624	14.375000	16.750000
Total sleep duration in minutes	503.170213	79.101770	281.000000	741.500000
Minutes awake during sleep period	95.143972	64.751393	20.633333	343.100000
Numeric label for day type (-1, 0, 1)	0.000000	0.466252	-1.0	1.0
Deep sleep as percentage of total sleep	0.137954	0.043405	0.051306	0.245700
Deep sleep within ideal range (-1, 0, 1)	-0.957447	0.204030	-1.0	0.0
Light sleep as percentage of total sleep	0.658727	0.065450	0.515119	0.783582
Light sleep within ideal range (-1, 0, 1)	0.978723	0.145865	0.0	1.0
REM sleep as percentage of total sleep	0.203318	0.046639	0.101423	0.309953
REM sleep within ideal range (-1, 0, 1)	-0.659574	0.700205	-1.0	1.0

Total sleep within ideal range (-1, 0, 1)	0.191489	0.612844	-1.0	1.0
Total sleep duration in hours	8.386170	1.318363	4.683333	12.358333

Single Class Bayes

intensity_label	sleep_category	ideal_value	count
high_intensity	Deep Sleep	Ideal	2
high_intensity	Deep Sleep	Too Little	29
high_intensity	Light Sleep	Ideal	1
high_intensity	Light Sleep	Too Much	30
high_intensity	Rem Sleep	Ideal	3
high_intensity	Rem Sleep	Too Little	25
high_intensity	Rem Sleep	Too Much	3
low_intensity	Deep Sleep	Too Little	5
low_intensity	Light Sleep	Too Much	5
low_intensity	Rem Sleep	Ideal	1
low_intensity	Rem Sleep	Too Little	4
medium_intensity	Deep Sleep	Too Little	11
medium_intensity	Light Sleep	Too Much	11
medium_intensity	Rem Sleep	Too Little	8
medium_intensity	Rem Sleep	Too Much	3

Multi-Class Bayes

intensity_label	Rem Sleep Quality	Light Sleep Quality	Deep Sleep Quality	count
high_intensity	Too Little	Too Much	Too Little	23
high_intensity	Too Little	Too Much	Ideal	2
high_intensity	Ideal	Too Much	Too Little	3
high_intensity	Too Much	Ideal	Too Little	1
high_intensity	Too Much	Too Much	Too Little	2
low_intensity	Too Little	Too Much	Too Little	4
low_intensity	Ideal	Too Much	Too Little	1
medium_intensity	Too Little	Too Much	Too Little	8
medium_intensity	Too Much	Too Much	Too Little	3

Appendix A: Raw Analysis

See the Jupyter Notebook below or see it here: https://github.com/addymmoran/data/blob/main/oura/09-2025_11-2025/hrv_sleep_exercise_analysis.ipynb

hrv_sleep_exercise_analysis

December 1, 2025

[455]:
```
%pip install pandas tabulate
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in
/Users/addy/Library/Python/3.9/lib/python/site-packages (2.3.3)
Requirement already satisfied: tabulate in
/Users/addy/Library/Python/3.9/lib/python/site-packages (0.9.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/Users/addy/Library/Python/3.9/lib/python/site-packages (from pandas)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/Users/addy/Library/Python/3.9/lib/python/site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/Users/addy/Library/Python/3.9/lib/python/site-packages (from pandas) (2025.2)
Requirement already satisfied: numpy>=1.22.4 in
/Users/addy/Library/Python/3.9/lib/python/site-packages (from pandas) (2.0.2)
Requirement already satisfied: six>=1.5 in /Library/Developer/CommandLineTools/L
ibrary/Frameworks/Python3.framework/Versions/3.9/lib/python3.9/site-packages
(from python-dateutil>=2.8.2->pandas) (1.15.0)
WARNING: You are using pip version 21.2.4; however, version 25.3 is
available.
You should consider upgrading via the
'/Library/Developer/CommandLineTools/usr/bin/python3 -m pip install --upgrade
pip' command.
Note: you may need to restart the kernel to use updated packages.
```

[456]:
```
from datetime import datetime, timedelta
import pandas as pd
import plotly.graph_objects as go
import plotly.io as pio
import math
import numpy as np
import plotly.express as px
import scipy.stats as stats
pio.renderers.default = "jupyterlab"
pd.options.plotting.backend = "plotly"
```

[457]:
```
# timestamp;bpm;source
heartrate_df = pd.read_csv('./data/heartrate.csv', sep=";", dtype={
        "bpm": "Int64",
        "source": "string",
    },
    parse_dates=[
        "timestamp"
    ])
heartrate_df["timestamp"] = heartrate_df["timestamp"].dt.tz_convert("America/
 ↪Denver")

#id;average_breath;average_heart_rate;average_hrv;awake_time;bedtime_end;
 ↪bedtime_start;day;deep_sleep_duration;efficiency;heart_rate;hrv;latency;
 ↪light_sleep_duration;low_battery_alert;lowest_heart_rate;movement_30_sec;
 ↪period;readiness;readiness_score_delta;rem_sleep_duration;restless_periods;
 ↪sleep_algorithm_version;sleep_analysis_reason;sleep_phase_30_sec;
 ↪sleep_phase_5_min;sleep_score_delta;time_in_bed;total_sleep_duration;type
sleep_df_tmp = pd.read_csv('./data/sleepmodel.csv', sep=";", dtype={
        "id": "string",
        "average_breath": "float",
        "average_heart_rate": "float",
        "average_hrv": "float",
        "awake_time": "Int64",
        "deep_sleep_duration": "Int64",
        "efficiency": "Int64",
        "latency": "Int64",
        "light_sleep_duration": "Int64",
        "low_battery_alert": "boolean",
        "lowest_heart_rate": "Int64",
        "movement_30_sec": "string",
        "period": "Int64",
        "readiness_score_delta": "Int64",
        "rem_sleep_duration": "Int64",
        "restless_periods": "Int64",
        "sleep_algorithm_version": "string",
        "sleep_analysis_reason": "string",
        "sleep_score_delta": "Int64",
        "time_in_bed": "Int64",
        "total_sleep_duration": "Int64",
        "type": "string",
    },
    parse_dates=[
        "bedtime_start",
        "bedtime_end",
        "day"
    ])
```

```
sleep_df = sleep_df_tmp[['day', 'total_sleep_duration', 'rem_sleep_duration',␣
 ↪'light_sleep_duration', 'deep_sleep_duration', 'awake_time', 'average_hrv',␣
 ↪'average_heart_rate', 'average_breath']]

sleep_df['total_sleep_duration'] = sleep_df['total_sleep_duration']/60
sleep_df['rem_sleep_duration'] = sleep_df['rem_sleep_duration']/60
sleep_df['light_sleep_duration'] = sleep_df['light_sleep_duration']/60
sleep_df['deep_sleep_duration'] = sleep_df['deep_sleep_duration']/60
sleep_df['awake_time'] = sleep_df['awake_time']/60

sleep_df = sleep_df.groupby('day', as_index=False, group_keys=True).
 ↪apply(lambda x: x.loc[x.total_sleep_duration.idxmax()]).dropna()

## Find Heart Rate During Each Period of Exercise
datetime_str = "%Y-%m-%dT%H:%M:%S.%f%z"

# id;activity;calories;day;distance;end_datetime;intensity;label;source;
 ↪start_datetime
data = []
with open('data/workout.csv', 'r') as fp:
    for row in fp:
        if 'id;activity;' in row: # header row
            continue
        fields = row.strip().split(';')

        entry = {
            'day': datetime.strptime(fields[3], "%Y-%m-%d"),
            'workout_type': fields[1],
            'workout_time': [datetime.strptime(fields[9], datetime_str),␣
 ↪datetime.strptime(fields[5], datetime_str)],

        }
        entry['day_before'] =  entry['day'] - timedelta(days=1)
        #sleep_day = sleep_df[sleep_df["day"] == entry['day'] -␣
 ↪timedelta(days=1)]
        #if sleep_day.empty:
        #    continue
        """
        entry['awake_time'] = sleep_day['awake_time'].item() if␣
 ↪sleep_day['awake_time'].any() else None
        entry['light_sleep_duration'] = sleep_day['light_sleep_duration'].
 ↪item()[0] if sleep_day['light_sleep_duration'].any() else None,
        entry['deep_sleep_duration'] = sleep_day['deep_sleep_duration'].item()␣
 ↪if sleep_day['deep_sleep_duration'].any() else None,
        entry['rem_sleep_duration'] = sleep_day['rem_sleep_duration'].item() if␣
 ↪sleep_day['rem_sleep_duration'].any() else None,
```

```
        entry['total_sleep_duration'] = sleep_day['total_sleep_duration'].
 ↪item() if sleep_day['total_sleep_duration'].any() else None,
        entry['hrv'] = sleep_day['average_hrv'].item() if␣
 ↪sleep_day['average_hrv'].any() else None,
        entry['sleep_hr'] = sleep_day['average_heart_rate'].item() if␣
 ↪sleep_day['average_heart_rate'].any() else None,
        entry['sleep_breath'] = sleep_day['average_breath'].item() if␣
 ↪sleep_day['average_breath'].any() else None"""

        entry['workout_duration'] = (entry['workout_time'][1] -␣
 ↪entry['workout_time'][0]).total_seconds()/60
        entry['workout_hr'] = heartrate_df[(heartrate_df["timestamp"] >=␣
 ↪entry['workout_time'][0]) & (heartrate_df["timestamp"] <=␣
 ↪entry['workout_time'][1])]
        entry['max_workout_hr'] = entry['workout_hr']['bpm'].max()
        entry['recovery_hr'] = heartrate_df[(heartrate_df["timestamp"] >=␣
 ↪entry['workout_time'][1]) & (heartrate_df["timestamp"] <=␣
 ↪entry['workout_time'][1] + timedelta(minutes=15))]
        entry['lowest_post_workout_hr'] = entry['recovery_hr']['bpm'].min()

        data.append(entry)

workout_sessions_df = pd.DataFrame(data)

workout_sessions_df = workout_sessions_df.merge(sleep_df, left_on='day_before',␣
 ↪right_on='day')
```

```
/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_94335/2665461091.py:4
4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_94335/2665461091.py:4
5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_94335/2665461091.py:4
6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_94335/2665461091.py:4
7: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_94335/2665461091.py:4
8: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_94335/2665461091.py:5
0: FutureWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is
deprecated, and in a future version of pandas the grouping columns will be
excluded from the operation. Either pass `include_groups=False` to exclude the
groupings or explicitly select the grouping columns after groupby to silence
this warning.
```

[458]:
```
len(set(workout_sessions_df['day_x'].to_list())) # number of days that have␣
↪exercise data
```

[458]:
```
47
```

[459]:
```
daily_df = workout_sessions_df.assign(
        day=lambda df: df['day_x'].dt.date,
```

```
        duration=lambda df: [row.workout_duration if row is not None else 0 for␣
 ↪row in df.itertuples()],
        max_exercise_hr=lambda df: [row.max_workout_hr if row is not None else␣
 ↪0 for row in df.itertuples()]
    )

#daily_df = daily_df.dropna(subset=['average_hrv', 'deep_sleep_duration',␣
 ↪'rem_sleep_duration', 'average_breath'])
daily_df = daily_df.dropna(subset=['light_sleep_duration',␣
 ↪'deep_sleep_duration', 'rem_sleep_duration'])
daily_df = daily_df.groupby('day', as_index=False).agg(
        total_max_hr=('max_exercise_hr', 'max'),
        total_duration=('duration', 'sum'),  # total duration in minutes per day
        deep_sleep_duration=('deep_sleep_duration', 'max'), # expected there is␣
 ↪only one field, so max doesn't do anything (including so the field is␣
 ↪included)
        rem_sleep_duration=('rem_sleep_duration', 'max'),
        light_sleep_duration=('light_sleep_duration', 'max'),
        sleep_hrv=('average_hrv', 'max'),
        sleep_breath=('average_breath', 'max'),
        total_sleep_duration=('total_sleep_duration', 'max'),
        awake_time=('awake_time', 'max')

)
daily_df
```

[459]:

	day	total_max_hr	total_duration	deep_sleep_duration	\
0	2025-09-19	159.0	182.230950	81.5	
1	2025-09-21	103.0	11.000000	85.5	
2	2025-09-22	118.0	28.308317	51.5	
3	2025-09-23	181.0	27.482133	59.5	
4	2025-09-24	148.0	110.382883	56.0	
5	2025-09-25	110.0	27.684667	34.0	
6	2025-09-27	182.0	45.610183	68.0	
7	2025-09-28	188.0	19.347450	77.5	
8	2025-09-29	124.0	65.000000	66.0	
9	2025-09-30	155.0	23.479233	27.5	
10	2025-10-01	181.0	52.285583	69.5	
11	2025-10-02	126.0	22.237100	77.5	
12	2025-10-06	113.0	58.148567	63.5	
13	2025-10-07	161.0	67.497917	68.5	
14	2025-10-08	184.0	50.620067	75.0	
15	2025-10-09	180.0	62.338150	78.5	
16	2025-10-10	96.0	8.977600	113.0	
17	2025-10-11	156.0	24.169450	43.0	
18	2025-10-12	142.0	113.093317	95.0	
19	2025-10-14	171.0	36.669183	94.0	

20	2025-10-15	113.0	40.000000	44.0
21	2025-10-16	176.0	78.394883	68.0
22	2025-10-17	150.0	104.208983	88.0
23	2025-10-18	157.0	45.995383	71.5
24	2025-10-19	107.0	64.000000	52.0
25	2025-10-24	171.0	21.920500	80.5
26	2025-10-25	176.0	28.485233	100.0
27	2025-10-27	184.0	57.943533	60.0
28	2025-10-28	98.0	32.041100	104.5
29	2025-10-29	166.0	22.000000	50.0
30	2025-10-30	148.0	42.467650	49.5
31	2025-11-01	171.0	35.605433	68.5
32	2025-11-02	169.0	21.351833	51.0
33	2025-11-03	131.0	36.284517	65.5
34	2025-11-04	166.0	40.873617	64.5
35	2025-11-05	166.0	20.053817	53.5
36	2025-11-06	174.0	77.587150	65.5
37	2025-11-07	117.0	113.000000	84.0
38	2025-11-08	127.0	48.055017	54.0
39	2025-11-09	112.0	19.309933	56.5
40	2025-11-10	183.0	49.989017	47.0
41	2025-11-11	167.0	16.712000	64.5
42	2025-11-12	NaN	76.659633	71.5
43	2025-11-13	181.0	27.914783	89.5
44	2025-11-16	178.0	82.516083	38.0
45	2025-11-18	106.0	22.000000	71.5
46	2025-11-22	132.0	151.000000	91.5

	rem_sleep_duration	light_sleep_duration	sleep_hrv	sleep_breath	\
0	114.5	359.0	41.0	15.000	
1	95.0	280.0	40.0	15.500	
2	89.0	277.0	37.0	15.500	
3	102.5	353.0	37.0	15.250	
4	95.0	404.0	43.0	15.875	
5	76.5	363.0	40.0	15.125	
6	163.5	296.0	43.0	15.000	
7	136.5	232.0	42.0	15.500	
8	193.5	390.0	53.0	16.250	
9	88.5	420.0	44.0	15.500	
10	72.0	296.5	49.0	15.750	
11	80.0	408.0	42.0	14.625	
12	92.0	173.0	39.0	15.000	
13	126.0	352.0	40.0	15.375	
14	90.0	382.0	41.0	15.625	
15	117.5	286.0	44.0	15.750	
16	101.5	333.0	39.0	15.250	
17	98.0	307.0	46.0	15.625	

18	129.5	238.5	47.0	15.625
19	87.5	396.5	40.0	15.625
20	71.0	188.0	46.0	15.625
21	28.5	184.5	50.0	15.125
22	170.0	483.5	53.0	15.250
23	71.5	447.0	40.0	15.500
24	115.0	375.5	39.0	15.000
25	111.0	312.5	28.0	16.750
26	81.0	226.0	38.0	15.000
27	100.0	331.5	41.0	15.750
28	90.0	290.0	46.0	15.125
29	89.5	293.0	39.0	15.125
30	118.5	355.5	44.0	14.875
31	87.0	283.0	39.0	14.375
32	114.5	411.0	46.0	15.000
33	91.0	342.5	35.0	14.625
34	85.0	369.0	42.0	15.250
35	78.5	405.5	48.0	15.000
36	107.5	332.5	45.0	15.375
37	96.0	338.0	48.0	14.875
38	110.5	343.5	42.0	15.375
39	82.0	355.5	44.0	15.500
40	106.0	368.0	57.0	15.500
41	122.0	310.5	47.0	15.500
42	123.5	351.0	44.0	15.500
43	69.0	312.0	40.0	15.875
44	91.5	429.0	53.0	15.125
45	99.0	369.5	43.0	14.875
46	151.0	297.5	43.0	15.250

	total_sleep_duration	awake_time
0	555.0	100.500000
1	460.5	51.766667
2	417.5	68.066667
3	515.0	52.766667
4	555.0	58.266667
5	473.5	92.766667
6	527.5	51.183333
7	446.0	46.933333
8	649.5	250.500000
9	536.0	343.100000
10	438.0	69.000000
11	565.5	118.666667
12	328.5	42.133333
13	546.5	116.233333
14	547.0	145.066667
15	482.0	66.750000

```
16                547.5   77.766667
17                448.0   57.466667
18                463.0   38.616667
19                578.0  141.083333
20                303.0   33.666667
21                281.0   67.133333
22                741.5  158.500000
23                590.0  290.816667
24                542.5   20.633333
25                504.0   66.933333
26                407.0  138.983333
27                491.5   92.500000
28                484.5   85.083333
29                432.5   71.400000
30                523.5   69.450000
31                438.5   84.916667
32                576.5   84.683333
33                499.0   62.400000
34                518.5  117.533333
35                537.5  214.300000
36                505.5   66.316667
37                518.0   47.183333
38                508.0  111.366667
39                494.0   96.283333
40                521.0   99.550000
41                497.0   39.900000
42                546.0   90.866667
43                470.5   65.583333
44                558.5   51.383333
45                540.0   82.683333
46                540.0   73.083333
```

[460]:
```python
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# --- Create a subplot with secondary y-axis ---
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['deep_sleep_duration'],
        name="Deep Sleep",
    ),
    secondary_y=False,
)
```

```
fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['rem_sleep_duration'],
        name="REM Sleep",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['light_sleep_duration'],
        name="Light Sleep",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['awake_time'],
        name="Awake",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Scatter(
        x=daily_df['day'],
        y=daily_df['sleep_hrv'],
        mode="markers+lines",
        name="HRV",
        marker=dict(size=8),
    ),
    secondary_y=True,
)


fig.update_layout(
    title="Sleep vs HRV",
    barmode="stack",
    xaxis_title="Date",
    legend_title="Metrics",
    height=600,
)
```

```
fig.update_yaxes(
    title_text="Sleep Duration (minutes)",
    secondary_y=False,
)

fig.update_yaxes(
    title_text="HRV",
    secondary_y=True,
)

fig.show()
```

[461]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# --- Create a subplot with secondary y-axis ---
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['deep_sleep_duration'],
        name="Deep Sleep",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['rem_sleep_duration'],
        name="REM Sleep",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['light_sleep_duration'],
        name="Light Sleep",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Bar(
```

```
        x=daily_df['day'],
        y=daily_df['awake_time'],
        name="Awake",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Scatter(
        x=daily_df['day'],
        y=daily_df['total_max_hr'],
        mode="markers+lines",
        name="Max Workout HR",
        marker=dict(size=8),
    ),
    secondary_y=True,
)

fig.update_layout(
    title="Sleep vs Max Workout HR",
    barmode="stack",
    xaxis_title="Date",
    legend_title="Metrics",
    height=600,
)

fig.update_yaxes(
    title_text="Sleep Duration (minutes)",
    secondary_y=False,
)

fig.update_yaxes(
    title_text="HR",
    secondary_y=True,
)

fig.show()
```

[462]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# --- Create a subplot with secondary y-axis ---
fig = make_subplots(specs=[[{"secondary_y": True}]])

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
```

```
        y=daily_df['deep_sleep_duration'],
        name="Deep Sleep",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['rem_sleep_duration'],
        name="REM Sleep",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['light_sleep_duration'],
        name="Light Sleep",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Bar(
        x=daily_df['day'],
        y=daily_df['awake_time'],
        name="Awake",
    ),
    secondary_y=False,
)

fig.add_trace(
    go.Scatter(
        x=daily_df['day'],
        y=daily_df['total_duration'],
        mode="markers+lines",
        name="Workout Duration",
        marker=dict(size=8),
    ),
    secondary_y=True,
)


fig.update_layout(
    title="Sleep vs Total Workout Duration",
```

```
    barmode="stack",
    xaxis_title="Date",
    legend_title="Metrics",
    height=600,
)

fig.update_yaxes(
    title_text="Sleep Duration (minutes)",
    secondary_y=False,
)

fig.update_yaxes(
    title_text="Workout Duration (minutes)",
    secondary_y=True,
)

fig.show()
```

[463]:
```
def residual(df, x_field, y_field):

    fig = go.Figure()

    fig.add_trace(go.Scatter(
        x=df[x_field],
        y=df[y_field],
        mode='markers',
        name='Actual',
        line=dict(color='red', width=2),
        marker=dict(size=6)
    ))

    # residual plot
    sum_x = df[x_field].sum()
    sum_x_2 = sum([x**2 for x in list(df[x_field])])
    sum_y = df[y_field].sum()
    sum_y_2 = sum([y**2 for y in list(df[y_field])])
    sum_x_y = sum([row[x_field] * row[y_field] for _, row in df.iterrows()])

    n = len(df)
    b = (n*sum_x_y - sum_x*sum_y)/(n*sum_x_2 - sum_x**2)
    a = (sum_y - b*sum_x)/n
    x = list(df[x_field].drop_duplicates())
    y_predicted = [a + b*x_i for x_i in x]

    fig.add_trace(go.Scatter(
        x=x,
        y=y_predicted,
```

```
        mode='lines',
        name='Prediction/Residual Plot',
        line=dict(color='black', width=2),
        marker=dict(size=6)
    ))

    # Layout
    fig.update_layout(
        title=f"{x_field} vs {y_field}",
        xaxis=dict(title=x_field),
        yaxis=dict(title=y_field),
        template="plotly_white",
        barmode='overlay'
    )

    diff_y_y_predicted = []

    for i in range(len(df)):
        #print(df.iloc[i][x_field], df.iloc[i][y_field], y_predicted[x.index(df.
↪iloc[i][x_field])])
        diff_y_y_predicted.append(df.iloc[i][y_field] - y_predicted[x.index(df.
↪iloc[i][x_field])])

    residual_fig = go.Figure()

    # Add Max HR line
    residual_fig.add_trace(go.Scatter(
        x=df[x_field],
        y=diff_y_y_predicted,
        mode='markers',
        name='Actual',
        line=dict(color='red', width=2),
        marker=dict(size=6)
    ))

    # Layout
    residual_fig.update_layout(
        title="Residual Plot",
        xaxis=dict(title=x_field),
        yaxis=dict(title="Residual, y-y_predicted"),
        template="plotly_white",
        legend=dict(x=0.01, y=0.99),
        barmode='overlay'
    )

    return fig, residual_fig
```

[464]:
```
fig, residual_fig = residual(daily_df.dropna(), 'total_duration',␣
 ↪'rem_sleep_duration')
fig.show()
residual_fig.show()
```

[465]:
```
fig, residual_fig = residual(daily_df.dropna(), 'total_max_hr',␣
 ↪'rem_sleep_duration')
fig.show()
residual_fig.show()
```

[466]:
```
fig, residual_fig = residual(daily_df.dropna(), 'total_max_hr', 'sleep_hrv')
fig.show()
residual_fig.show()
```

[467]:
```
fig, residual_fig = residual(daily_df.dropna(), 'total_max_hr',␣
 ↪'total_sleep_duration')
fig.show()
residual_fig.show()
```

[468]:
```
fig, residual_fig = residual(daily_df, 'sleep_hrv', 'rem_sleep_duration')
fig.show()
residual_fig.show()
```

[469]:
```
fig, residual_fig = residual(daily_df, 'sleep_hrv', 'deep_sleep_duration')
fig.show()
residual_fig.show()
```

[470]:
```
fig, residual_fig = residual(daily_df, 'sleep_hrv', 'light_sleep_duration')
fig.show()
residual_fig.show()
```

[471]:
```
# pretty reliably say these are all nonparametric distribution

corr = daily_df.drop('day', axis=1).corr('spearman')

fig = px.imshow(
        corr,
        text_auto=True,  # Display correlation values on the heatmap
        color_continuous_scale='RdBu',  # Choose a diverging color scale
        zmin=-1, zmax=1,  # Set the color scale range for correlations
        title="Correlation Heatmap",
        width=700,
        height=600
)

fig.show()
```

Spearman	Interpretation
+1.00	Perfect monotonic increase
+0.50	Moderate monotonic increase
0	No monotonic association
–0.50	Moderate monotonic decrease
–1.00	Perfect monotonic decrease

The sleep fields that were most correlated (both positive and negative), in order of most correlated, for the following exercise fields are: * total_max_hr: sleep_breath (positive) * total_(workout)_duration: sleep_hrv (positive), rem_sleep_duration (positive), total_sleep_duration (positive)

[472]:
```
# Define good and hard days
hard_days = [datetime.strptime(d, "%m/%d/%Y").date() for d in
             ["11/19/2025", "11/13/2025", "11/12/2025", "11/11/2025", "11/02/
 ↪2025", "10/30/2025", "10/28/2025", "10/24/2025", "10/17/2025", "10/09/2025",␣
 ↪"09/28/2025", "09/15/2025", "09/04/2025"]]
good_days = [datetime.strptime(d, "%m/%d/%Y").date() for d in
             ["11/10/2025", "11/06/2025", "11/01/2025", "10/27/2025", "10/25/
 ↪2025","10/16/2025", "10/12/2025", "10/08/2025",  "09/24/2025", "09/16/2025",␣
 ↪"09/14/2025", "09/11/2025", "09/09/2025", "09/05/2025"]]


tmp = daily_df[daily_df['day'].isin(hard_days) | daily_df['day'].
 ↪isin(good_days)]

days_have_data_for = list(tmp['day'])

hard_days_tmp = [day for day in hard_days if day in days_have_data_for]
good_days_tmp = [day for day in good_days if day in days_have_data_for]

fig = go.Figure()

fig.add_trace(go.Scatter(
    x=tmp['day'],
    y=tmp['sleep_hrv'],
    mode='markers+lines',
    name='HRV for Night Before',
    line=dict(color='blue', width=2),
    yaxis='y1'
))

fig.add_trace(go.Bar(
    x=tmp['day'],
    y=tmp['total_sleep_duration'],
    name='Total Sleep Duration (min)',
```

```
    marker=dict(color='orange', opacity=0.6),
    yaxis='y2'
))

# Add Max HR line
fig.add_trace(go.Scatter(
    x=tmp['day'],
    y=tmp['total_max_hr'],
    mode='markers',
    name='Max Exercise HR',
    line=dict(color='red', width=2),
    marker=dict(size=6),
    yaxis='y2'
))

fig.add_trace(go.Scatter(
    x=tmp['day'],
    y=tmp['total_duration'],
    mode='markers',
    name='Exercise Duration',
    line=dict(color='grey', width=2),
    marker=dict(size=6),
    yaxis='y2'
))

# Shade good and hard days with clear full-day rectangles
for d in hard_days_tmp:
    fig.add_vrect(
        x0=d, x1=d + timedelta(days=1),
        fillcolor="red", opacity=0.2,
        layer="below", line_width=0,
    )

for d in good_days_tmp:
    fig.add_vrect(
        x0=d, x1=d + timedelta(days=1),
        fillcolor="green", opacity=0.2,
        layer="below", line_width=0,
    )


# Layout
fig.update_layout(
    title="HRV Day Before -> How Difficult the Exercise Felt",
    xaxis=dict(title="Day"),
    yaxis=dict(
        title="HRV",
```

```
        side="left",
        showgrid=True
    ),
    yaxis2=dict(
        title="Max HR / Exercise Duration",
        overlaying="y",
        side="right",
        showgrid=False
    ),
    template="plotly_white",
    #legend=dict(x=0.01, y=0.99),
    barmode='overlay'
)

fig.show()
```

Does seem to be a slight trend where the days I felt best, my HRV was high, and the days I struggled, my HRV was low. Not seeing much of a trend from the max HR or workout duration though. Nor am I seeing a significant trend based on how long I slept.

[473]:
```
# Define good and hard days
hard_days = [datetime.strptime(d, "%m/%d/%Y").date() for d in
             ["11/11/2025", "11/02/2025", "10/30/2025", "10/28/2025", "10/9/
 ↪2025", "09/15/2025", "09/04/2025"]]
good_days = [datetime.strptime(d, "%m/%d/%Y").date() for d in
             ["11/10/2025", "11/06/2025", "11/01/2025", "10/16/2025", "10/08/
 ↪2025", "09/16/2025", "09/14/2025", "09/09/2025", "09/05/2025"]]


tmp = daily_df[daily_df['day'].isin(hard_days) | daily_df['day'].
 ↪isin(good_days)]

days_have_data_for = list(tmp['day'])

hard_days_tmp = [day for day in hard_days if day in days_have_data_for]
good_days_tmp = [day for day in good_days if day in days_have_data_for]

fig = go.Figure()

fig.add_trace(go.Scatter(
    x=tmp['day'],
    y=tmp['sleep_breath'],
    mode='markers+lines',
    name='Sleep Breath Night Before',
    yaxis='y2'
))
```

```
fig.add_trace(go.Scatter(
    x=tmp['day'],
    y=tmp['rem_sleep_duration'],
    mode='markers+lines',
    name='REM Duration',
    yaxis='y1'
))

fig.add_trace(go.Scatter(
    x=tmp['day'],
    y=tmp['deep_sleep_duration'],
    mode='markers+lines',
    name='Deep Duration',
    yaxis='y1'
))

fig.add_trace(go.Scatter(
    x=tmp['day'],
    y=tmp['light_sleep_duration'],
    mode='markers+lines',
    name='Light Duration',
    yaxis='y1'
))

# Shade good and hard days with clear full-day rectangles
for d in hard_days_tmp:
    fig.add_vrect(
        x0=d, x1=d + timedelta(days=1),
        fillcolor="red", opacity=0.2,
        layer="below", line_width=0,
    )

for d in good_days_tmp:
    fig.add_vrect(
        x0=d, x1=d + timedelta(days=1),
        fillcolor="green", opacity=0.2,
        layer="below", line_width=0,
    )


# Layout
fig.update_layout(
    title="HRV Day Before -> How Difficult the Exercise Felt",
    xaxis=dict(title="Day"),
    yaxis=dict(
        title="Duration (minutes)",
        side="left",
```

```
        showgrid=True
    ),
    yaxis2=dict(
        title="Sleep Breath Count",
        overlaying="y",
        side="right",
        showgrid=False
    ),
    template="plotly_white",
    barmode='overlay'
)

fig.show()
```

No obvious trends between “good” and “bad” days.

[474]:
```
# using good/bad day labels, see if there is any correlation

# similar to rate of perceived effort
# "good" = 1
# "hard" = -1
# "unremarkable" = 0
daily_df['day_label'] = daily_df.apply(lambda x: "Good" if x['day'] in␣
 ↪good_days else ("Hard" if x['day'] in hard_days else "Unremarkable"),␣
 ↪axis=1) # axis=1 applies function row-wise
daily_df['day_label_num'] = daily_df.apply(lambda x: 1 if x['day'] in good_days␣
 ↪else (-1 if x['day'] in hard_days else 0), axis=1) # axis=1 applies function␣
 ↪row-wise

daily_df.to_csv()
```

[474]:
```
',day,total_max_hr,total_duration,deep_sleep_duration,rem_sleep_duration,light_s
leep_duration,sleep_hrv,sleep_breath,total_sleep_duration,awake_time,day_label,d
ay_label_num\n0,2025-09-19,159.0,182.23095,81.5,114.5,359.0,41.0,15.0,555.0,100.
5,Unremarkable,0\n1,2025-09-21,103.0,11.0,85.5,95.0,280.0,40.0,15.5,460.5,51.766
666666666666,Unremarkable,0\n2,2025-09-22,118.0,28.308316666666666,51.5,89.0,277
.0,37.0,15.5,417.5,68.06666666666666,Unremarkable,0\n3,2025-09-23,181.0,27.48213
3333333334,59.5,102.5,353.0,37.0,15.25,515.0,52.766666666666666,Unremarkable,0\n
4,2025-09-24,148.0,110.38288333333334,56.0,95.0,404.0,43.0,15.875,555.0,58.26666
6666666666,Unremarkable,0\n5,2025-09-25,110.0,27.684666666666665,34.0,76.5,363.0
,40.0,15.125,473.5,92.76666666666667,Unremarkable,0\n6,2025-09-27,182.0,45.61018
333333333,68.0,163.5,296.0,43.0,15.0,527.5,51.18333333333333,Unremarkable,0\n7,2
025-09-28,188.0,19.34745,77.5,136.5,232.0,42.0,15.5,446.0,46.93333333333333,Unre
markable,0\n8,2025-09-29,124.0,65.0,66.0,193.5,390.0,53.0,16.25,649.5,250.5,Unre
markable,0\n9,2025-09-30,155.0,23.479233333333333,27.5,88.5,420.0,44.0,15.5,536.
0,343.1,Unremarkable,0\n10,2025-10-01,181.0,52.285583333333335,69.5,72.0,296.5,4
9.0,15.75,438.0,69.0,Unremarkable,0\n11,2025-10-02,126.0,22.2371,77.5,80.0,408.0
```

,42.0,14.625,565.5,118.66666666666667,Unremarkable,0\n12,2025-10-06,113.0,58.148
56666666667,63.5,92.0,173.0,39.0,15.0,328.5,42.13333333333333,Unremarkable,0\n13
,2025-10-07,161.0,67.49791666666667,68.5,126.0,352.0,40.0,15.375,546.5,116.23333
333333333,Unremarkable,0\n14,2025-10-08,184.0,50.620066666666666,75.0,90.0,382.0
,41.0,15.625,547.0,145.06666666666666,Good,1\n15,2025-10-09,180.0,62.33815,78.5,
117.5,286.0,44.0,15.75,482.0,66.75,Hard,-1\n16,2025-10-10,96.0,8.977599999999999
,113.0,101.5,333.0,39.0,15.25,547.5,77.76666666666667,Unremarkable,0\n17,2025-10
-11,156.0,24.169449999999998,43.0,98.0,307.0,46.0,15.625,448.0,57.46666666666667
,Unremarkable,0\n18,2025-10-12,142.0,113.09331666666667,95.0,129.5,238.5,47.0,15
.625,463.0,38.61666666666667,Unremarkable,0\n19,2025-10-14,171.0,36.669183333333
336,94.0,87.5,396.5,40.0,15.625,578.0,141.08333333333334,Unremarkable,0\n20,2025
-10-15,113.0,40.0,44.0,71.0,188.0,46.0,15.625,303.0,33.666666666666664,Unremarka
ble,0\n21,2025-10-16,176.0,78.39488333333334,68.0,28.5,184.5,50.0,15.125,281.0,6
7.13333333333334,Good,1\n22,2025-10-17,150.0,104.20898333333334,88.0,170.0,483.5
,53.0,15.25,741.5,158.5,Unremarkable,0\n23,2025-10-18,157.0,45.995383333333336,7
1.5,71.5,447.0,40.0,15.5,590.0,290.81666666666666,Unremarkable,0\n24,2025-10-19,
107.0,64.0,52.0,115.0,375.5,39.0,15.0,542.5,20.633333333333333,Unremarkable,0\n2
5,2025-10-24,171.0,21.9205,80.5,111.0,312.5,28.0,16.75,504.0,66.93333333333334,U
nremarkable,0\n26,2025-10-25,176.0,28.485233333333333,100.0,81.0,226.0,38.0,15.0
,407.0,138.98333333333332,Unremarkable,0\n27,2025-10-27,184.0,57.943533333333335
,60.0,100.0,331.5,41.0,15.75,491.5,92.5,Unremarkable,0\n28,2025-10-28,98.0,32.04
11,104.5,90.0,290.0,46.0,15.125,484.5,85.08333333333333,Hard,-1\n29,2025-10-29,1
66.0,22.0,50.0,89.5,293.0,39.0,15.125,432.5,71.4,Unremarkable,0\n30,2025-10-30,1
48.0,42.467650000000006,49.5,118.5,355.5,44.0,14.875,523.5,69.45,Hard,-1\n31,202
5-11-01,171.0,35.60543333333333,68.5,87.0,283.0,39.0,14.375,438.5,84.91666666666
667,Good,1\n32,2025-11-02,169.0,21.35183333333333,51.0,114.5,411.0,46.0,15.0,576
.5,84.68333333333334,Hard,-1\n33,2025-11-03,131.0,36.28451666666667,65.5,91.0,34
2.5,35.0,14.625,499.0,62.4,Unremarkable,0\n34,2025-11-04,166.0,40.87361666666666
,64.5,85.0,369.0,42.0,15.25,518.5,117.53333333333333,Unremarkable,0\n35,2025-11-
05,166.0,20.053816666666666,53.5,78.5,405.5,48.0,15.0,537.5,214.3,Unremarkable,0
\n36,2025-11-06,174.0,77.58715000000001,65.5,107.5,332.5,45.0,15.375,505.5,66.31
666666666666,Good,1\n37,2025-11-07,117.0,113.0,84.0,96.0,338.0,48.0,14.875,518.0
,47.18333333333333,Unremarkable,0\n38,2025-11-08,127.0,48.05501666666667,54.0,11
0.5,343.5,42.0,15.375,508.0,111.36666666666666,Unremarkable,0\n39,2025-11-09,112
.0,19.309933333333333,56.5,82.0,355.5,44.0,15.5,494.0,96.28333333333333,Unremark
able,0\n40,2025-11-10,183.0,49.989016666666664,47.0,106.0,368.0,57.0,15.5,521.0,
99.55,Good,1\n41,2025-11-11,167.0,16.712,64.5,122.0,310.5,47.0,15.5,497.0,39.9,H
ard,-1\n42,2025-11-12,,76.65963333333335,71.5,123.5,351.0,44.0,15.5,546.0,90.866
66666666666,Unremarkable,0\n43,2025-11-13,181.0,27.914783333333332,89.5,69.0,312
.0,40.0,15.875,470.5,65.58333333333333,Unremarkable,0\n44,2025-11-16,178.0,82.51
608333333334,38.0,91.5,429.0,53.0,15.125,558.5,51.38333333333333,Unremarkable,0\
n45,2025-11-18,106.0,22.0,71.5,99.0,369.5,43.0,14.875,540.0,82.68333333333334,Un
remarkable,0\n46,2025-11-22,132.0,151.0,91.5,151.0,297.5,43.0,15.25,540.0,73.083
33333333333,Unremarkable,0\n'

[475]:
```
corr = daily_df.drop('day', axis=1).corr('spearman', numeric_only=True)
```

```
fig = px.imshow(
        corr,
        text_auto=True,  # Display correlation values on the heatmap
        color_continuous_scale='RdBu',  # Choose a diverging color scale
        zmin=-1, zmax=1,  # Set the color scale range for correlations
        title="Correlation Heatmap",
        width=700,
        height=600
)

fig.show()
```

The day label may be correlated with rem_sleep_duration (negative), total_duration (positive), and amount awake (positive)

[476]:
```
## wonder if we think about it as a range, less than 7 hours of sleep, more␣
 ↪than 9 hours of sleep, HRV < X, HRV > Y -- binning

variables = [
    "rem_sleep_duration",
    "light_sleep_duration",
    "deep_sleep_duration",
    "total_sleep_duration",
    "sleep_hrv",
    "sleep_breath",
    "awake_time"
]

#labels = ["Very Low", "Low", "High", "Very High"]
labels = [0, 1, 2, 3]
for var in variables:
    daily_df[f"{var}_qbin"], ranges = pd.qcut(
        daily_df[var],
        q=4,
        labels=labels,
        duplicates="drop",
        retbins=True
    )
    print(f"{var} Ranges:")
    for i in range(len(ranges)-1):
        print(f"Bin {i+1}: {ranges[i]:.1f} to {ranges[i+1]:.1f} minutes")
```

```
rem_sleep_duration Ranges:
Bin 1: 28.5 to 87.2 minutes
Bin 2: 87.2 to 96.0 minutes
Bin 3: 96.0 to 114.8 minutes
Bin 4: 114.8 to 193.5 minutes
light_sleep_duration Ranges:
```

```
Bin 1: 173.0 to 294.5 minutes
Bin 2: 294.5 to 342.5 minutes
Bin 3: 342.5 to 372.5 minutes
Bin 4: 372.5 to 483.5 minutes
deep_sleep_duration Ranges:
Bin 1: 27.5 to 53.8 minutes
Bin 2: 53.8 to 68.0 minutes
Bin 3: 68.0 to 79.5 minutes
Bin 4: 79.5 to 113.0 minutes
total_sleep_duration Ranges:
Bin 1: 281.0 to 466.8 minutes
Bin 2: 466.8 to 515.0 minutes
Bin 3: 515.0 to 546.2 minutes
Bin 4: 546.2 to 741.5 minutes
sleep_hrv Ranges:
Bin 1: 28.0 to 40.0 minutes
Bin 2: 40.0 to 43.0 minutes
Bin 3: 43.0 to 46.0 minutes
Bin 4: 46.0 to 57.0 minutes
sleep_breath Ranges:
Bin 1: 14.4 to 15.0 minutes
Bin 2: 15.0 to 15.4 minutes
Bin 3: 15.4 to 15.6 minutes
Bin 4: 15.6 to 16.8 minutes
awake_time Ranges:
Bin 1: 20.6 to 57.9 minutes
Bin 2: 57.9 to 73.1 minutes
Bin 3: 73.1 to 105.9 minutes
Bin 4: 105.9 to 343.1 minutes
```

[477]:
```
corr = daily_df[['total_sleep_duration_qbin', 'rem_sleep_duration_qbin',␣
 ↪'deep_sleep_duration', 'light_sleep_duration','sleep_hrv_qbin',␣
 ↪'sleep_breath_qbin', 'awake_time_qbin', 'day_label_num']].corr('spearman')

fig = px.imshow(
        corr,
        text_auto=True,  # Display correlation values on the heatmap
        color_continuous_scale='RdBu',  # Choose a diverging color scale
        zmin=-1, zmax=1,  # Set the color scale range for correlations
        title="Correlation Heatmap",
        width=700,
        height=600
)

fig.show()
```

REM sleep (negative) is the most prominent correlation field for the “day label”. Meaning the days with less REM mean I have worse days for exercise

[478]:
```
daily_df.groupby("rem_sleep_duration_qbin")["day_label_num"].mean()
```

```
/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_94335/2578610582.py:1
: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
```

[478]:
```
rem_sleep_duration_qbin
0    0.166667
1    0.000000
2    0.090909
3   -0.250000
Name: day_label_num, dtype: float64
```

[479]:
```
daily_df[daily_df['day_label']=="Good"].describe()
```

[479]:

	total_duration	deep_sleep_duration	rem_sleep_duration	\
count	5.000000	5.000000	5.00000	
mean	58.439310	64.800000	83.80000	
std	18.833472	10.551066	32.25407	
min	35.605433	47.000000	28.50000	
25%	49.989017	65.500000	87.00000	
50%	50.620067	68.000000	90.00000	
75%	77.587150	68.500000	106.00000	
max	78.394883	75.000000	107.50000	

	light_sleep_duration	sleep_hrv	sleep_breath	total_sleep_duration	\
count	5.000000	5.000000	5.000000	5.000000	
mean	310.000000	46.400000	15.200000	458.600000	
std	79.896026	7.266361	0.496865	107.067385	
min	184.500000	39.000000	14.375000	281.000000	
25%	283.000000	41.000000	15.125000	438.500000	
50%	332.500000	45.000000	15.375000	505.500000	
75%	368.000000	50.000000	15.500000	521.000000	
max	382.000000	57.000000	15.625000	547.000000	

	awake_time	day_label_num
count	5.000000	5.0
mean	92.596667	1.0
std	32.401589	0.0
min	66.316667	1.0
25%	67.133333	1.0
50%	84.916667	1.0
75%	99.550000	1.0
max	145.066667	1.0

[480]:
```
daily_df[daily_df['day_label']!="Good"].describe()
```

[480]:

	total_duration	deep_sleep_duration	rem_sleep_duration	\
count	42.000000	42.000000	42.000000	
mean	50.493721	68.214286	104.511905	
std	38.507132	19.558587	27.717980	
min	8.977600	27.500000	69.000000	
25%	22.547633	53.625000	87.750000	
50%	38.334592	67.000000	97.000000	
75%	63.584537	81.250000	116.875000	
max	182.230950	113.000000	193.500000	

	light_sleep_duration	sleep_hrv	sleep_breath	total_sleep_duration	\
count	42.000000	42.000000	42.000000	42.000000	
mean	335.750000	42.738095	15.354167	508.476190	
std	67.091151	4.869314	0.416794	75.037791	
min	173.000000	28.000000	14.625000	303.000000	
25%	296.125000	40.000000	15.000000	471.250000	
50%	343.000000	42.500000	15.312500	516.500000	
75%	374.000000	46.000000	15.593750	546.375000	
max	483.500000	53.000000	16.750000	741.500000	

	awake_time	day_label_num
count	42.000000	42.000000
mean	95.447222	-0.119048
std	67.828769	0.327770
min	20.633333	-1.000000
25%	53.941667	0.000000
50%	72.241667	0.000000
75%	108.650000	0.000000
max	343.100000	0.000000

Interestingly, when I have less REM I have better days…

Bin	REM Range (min)	Corr (with day_label_num)	Interpretation
1	**28.5 – 87.2**	**+0.1667**	Moderate positive impact — these lower REM nights tend to correlate with *better* day labels.
2	**87.2 – 96.0**	**0.0000**	No relationship — neutral.
3	**96.0 – 114.8**	**+0.0909**	Slight positive impact.
4	**114.8 – 193.5**	**–0.2500**	**Strong negative impact — high REM correlates with worse day labels**.

[481]:
```
# define what "ideal" looks like
ideal_total_duration = [7*60, 9*60]  # 7-9 hours
ideal_sleep_distribution = {
    'deep_sleep_duration': 0.25,
    'light_sleep_duration': 0.50,
    'rem_sleep_duration': 0.25
}

percent_epsilon = 0.02

for field_name, ideal_percent in ideal_sleep_distribution.items():

    # compute percentage
    pct_col = f"{field_name}_percentage"
    daily_df[pct_col] = daily_df[field_name] / daily_df['total_sleep_duration']

    # thresholds
    lower = ideal_percent - percent_epsilon
    upper = ideal_percent + percent_epsilon

    # label: -1 = too little, 0 = ideal, 1 = too much
    daily_df[f"{field_name}_is_ideal"] = (
        daily_df[pct_col].apply(
            lambda v: -1 if v < lower else (0 if lower <= v <= upper else 1)
        )
    )
daily_df['total_sleep_duration_hr'] = daily_df['total_sleep_duration'] / 60
daily_df[f"total_sleep_duration_is_ideal"] = (
        daily_df['total_sleep_duration'].apply(
            lambda v: -1 if v < ideal_total_duration[0] else (0 if␣
 ↪ideal_total_duration[0] <= v <= ideal_total_duration[1] else 1)
        )
    )
```

[482]:
```
daily_df
```

[482]:

	day	total_max_hr	total_duration	deep_sleep_duration	\
0	2025-09-19	159.0	182.230950	81.5	
1	2025-09-21	103.0	11.000000	85.5	
2	2025-09-22	118.0	28.308317	51.5	
3	2025-09-23	181.0	27.482133	59.5	
4	2025-09-24	148.0	110.382883	56.0	
5	2025-09-25	110.0	27.684667	34.0	
6	2025-09-27	182.0	45.610183	68.0	
7	2025-09-28	188.0	19.347450	77.5	
8	2025-09-29	124.0	65.000000	66.0	
9	2025-09-30	155.0	23.479233	27.5	

10	2025-10-01	181.0	52.285583	69.5
11	2025-10-02	126.0	22.237100	77.5
12	2025-10-06	113.0	58.148567	63.5
13	2025-10-07	161.0	67.497917	68.5
14	2025-10-08	184.0	50.620067	75.0
15	2025-10-09	180.0	62.338150	78.5
16	2025-10-10	96.0	8.977600	113.0
17	2025-10-11	156.0	24.169450	43.0
18	2025-10-12	142.0	113.093317	95.0
19	2025-10-14	171.0	36.669183	94.0
20	2025-10-15	113.0	40.000000	44.0
21	2025-10-16	176.0	78.394883	68.0
22	2025-10-17	150.0	104.208983	88.0
23	2025-10-18	157.0	45.995383	71.5
24	2025-10-19	107.0	64.000000	52.0
25	2025-10-24	171.0	21.920500	80.5
26	2025-10-25	176.0	28.485233	100.0
27	2025-10-27	184.0	57.943533	60.0
28	2025-10-28	98.0	32.041100	104.5
29	2025-10-29	166.0	22.000000	50.0
30	2025-10-30	148.0	42.467650	49.5
31	2025-11-01	171.0	35.605433	68.5
32	2025-11-02	169.0	21.351833	51.0
33	2025-11-03	131.0	36.284517	65.5
34	2025-11-04	166.0	40.873617	64.5
35	2025-11-05	166.0	20.053817	53.5
36	2025-11-06	174.0	77.587150	65.5
37	2025-11-07	117.0	113.000000	84.0
38	2025-11-08	127.0	48.055017	54.0
39	2025-11-09	112.0	19.309933	56.5
40	2025-11-10	183.0	49.989017	47.0
41	2025-11-11	167.0	16.712000	64.5
42	2025-11-12	NaN	76.659633	71.5
43	2025-11-13	181.0	27.914783	89.5
44	2025-11-16	178.0	82.516083	38.0
45	2025-11-18	106.0	22.000000	71.5
46	2025-11-22	132.0	151.000000	91.5

	rem_sleep_duration	light_sleep_duration	sleep_hrv	sleep_breath	\
0	114.5	359.0	41.0	15.000	
1	95.0	280.0	40.0	15.500	
2	89.0	277.0	37.0	15.500	
3	102.5	353.0	37.0	15.250	
4	95.0	404.0	43.0	15.875	
5	76.5	363.0	40.0	15.125	
6	163.5	296.0	43.0	15.000	
7	136.5	232.0	42.0	15.500	

```
8                193.5                390.0       53.0        16.250
9                 88.5                420.0       44.0        15.500
10                72.0                296.5       49.0        15.750
11                80.0                408.0       42.0        14.625
12                92.0                173.0       39.0        15.000
13               126.0                352.0       40.0        15.375
14                90.0                382.0       41.0        15.625
15               117.5                286.0       44.0        15.750
16               101.5                333.0       39.0        15.250
17                98.0                307.0       46.0        15.625
18               129.5                238.5       47.0        15.625
19                87.5                396.5       40.0        15.625
20                71.0                188.0       46.0        15.625
21                28.5                184.5       50.0        15.125
22               170.0                483.5       53.0        15.250
23                71.5                447.0       40.0        15.500
24               115.0                375.5       39.0        15.000
25               111.0                312.5       28.0        16.750
26                81.0                226.0       38.0        15.000
27               100.0                331.5       41.0        15.750
28                90.0                290.0       46.0        15.125
29                89.5                293.0       39.0        15.125
30               118.5                355.5       44.0        14.875
31                87.0                283.0       39.0        14.375
32               114.5                411.0       46.0        15.000
33                91.0                342.5       35.0        14.625
34                85.0                369.0       42.0        15.250
35                78.5                405.5       48.0        15.000
36               107.5                332.5       45.0        15.375
37                96.0                338.0       48.0        14.875
38               110.5                343.5       42.0        15.375
39                82.0                355.5       44.0        15.500
40               106.0                368.0       57.0        15.500
41               122.0                310.5       47.0        15.500
42               123.5                351.0       44.0        15.500
43                69.0                312.0       40.0        15.875
44                91.5                429.0       53.0        15.125
45                99.0                369.5       43.0        14.875
46               151.0                297.5       43.0        15.250

    total_sleep_duration  awake_time  … sleep_breath_qbin  awake_time_qbin  \
0                  555.0  100.500000  …                 0                2
1                  460.5   51.766667  …                 2                0
2                  417.5   68.066667  …                 2                1
3                  515.0   52.766667  …                 1                0
4                  555.0   58.266667  …                 3                1
5                  473.5   92.766667  …                 1                2
```

```
6                  527.5   51.183333  …                  0                  0
7                  446.0   46.933333  …                  2                  0
8                  649.5  250.500000  …                  3                  3
9                  536.0  343.100000  …                  2                  3
10                 438.0   69.000000  …                  3                  1
11                 565.5  118.666667  …                  0                  3
12                 328.5   42.133333  …                  0                  0
13                 546.5  116.233333  …                  1                  3
14                 547.0  145.066667  …                  3                  3
15                 482.0   66.750000  …                  3                  1
16                 547.5   77.766667  …                  1                  2
17                 448.0   57.466667  …                  3                  0
18                 463.0   38.616667  …                  3                  0
19                 578.0  141.083333  …                  3                  3
20                 303.0   33.666667  …                  3                  0
21                 281.0   67.133333  …                  1                  1
22                 741.5  158.500000  …                  1                  3
23                 590.0  290.816667  …                  2                  3
24                 542.5   20.633333  …                  0                  0
25                 504.0   66.933333  …                  3                  1
26                 407.0  138.983333  …                  0                  3
27                 491.5   92.500000  …                  3                  2
28                 484.5   85.083333  …                  1                  2
29                 432.5   71.400000  …                  1                  1
30                 523.5   69.450000  …                  0                  1
31                 438.5   84.916667  …                  0                  2
32                 576.5   84.683333  …                  0                  2
33                 499.0   62.400000  …                  0                  1
34                 518.5  117.533333  …                  1                  3
35                 537.5  214.300000  …                  0                  3
36                 505.5   66.316667  …                  1                  1
37                 518.0   47.183333  …                  0                  0
38                 508.0  111.366667  …                  1                  3
39                 494.0   96.283333  …                  2                  2
40                 521.0   99.550000  …                  2                  2
41                 497.0   39.900000  …                  2                  0
42                 546.0   90.866667  …                  2                  2
43                 470.5   65.583333  …                  3                  1
44                 558.5   51.383333  …                  1                  0
45                 540.0   82.683333  …                  0                  2
46                 540.0   73.083333  …                  1                  1

   deep_sleep_duration_percentage deep_sleep_duration_is_ideal  \
0                        0.146847                           -1
1                        0.185668                           -1
2                        0.123353                           -1
3                        0.115534                           -1
```

```
4                          0.100901                            -1
5                          0.071806                            -1
6                          0.128910                            -1
7                          0.173767                            -1
8                          0.101617                            -1
9                          0.051306                            -1
10                         0.158676                            -1
11                         0.137047                            -1
12                         0.193303                            -1
13                         0.125343                            -1
14                         0.137112                            -1
15                         0.162863                            -1
16                         0.206393                            -1
17                         0.095982                            -1
18                         0.205184                            -1
19                         0.162630                            -1
20                         0.145215                            -1
21                         0.241993                             0
22                         0.118678                            -1
23                         0.121186                            -1
24                         0.095853                            -1
25                         0.159722                            -1
26                         0.245700                             0
27                         0.122075                            -1
28                         0.215686                            -1
29                         0.115607                            -1
30                         0.094556                            -1
31                         0.156214                            -1
32                         0.088465                            -1
33                         0.131263                            -1
34                         0.124397                            -1
35                         0.099535                            -1
36                         0.129575                            -1
37                         0.162162                            -1
38                         0.106299                            -1
39                         0.114372                            -1
40                         0.090211                            -1
41                         0.129779                            -1
42                         0.130952                            -1
43                         0.190223                            -1
44                         0.068039                            -1
45                         0.132407                            -1
46                         0.169444                            -1

   light_sleep_duration_percentage light_sleep_duration_is_ideal  \
0                         0.646847                             1
1                         0.608035                             1
```

```
2                          0.663473                              1
3                          0.685437                              1
4                          0.727928                              1
5                          0.766631                              1
6                          0.561137                              1
7                          0.520179                              1
8                          0.600462                              1
9                          0.783582                              1
10                         0.676941                              1
11                         0.721485                              1
12                         0.526636                              1
13                         0.644099                              1
14                         0.698355                              1
15                         0.593361                              1
16                         0.608219                              1
17                         0.685268                              1
18                         0.515119                              0
19                         0.685986                              1
20                         0.620462                              1
21                         0.656584                              1
22                         0.652057                              1
23                         0.757627                              1
24                         0.692166                              1
25                         0.620040                              1
26                         0.555283                              1
27                         0.674466                              1
28                         0.598555                              1
29                         0.677457                              1
30                         0.679083                              1
31                         0.645382                              1
32                         0.712923                              1
33                         0.686373                              1
34                         0.711668                              1
35                         0.754419                              1
36                         0.657765                              1
37                         0.652510                              1
38                         0.676181                              1
39                         0.719636                              1
40                         0.706334                              1
41                         0.624748                              1
42                         0.642857                              1
43                         0.663124                              1
44                         0.768129                              1
45                         0.684259                              1
46                         0.550926                              1

   rem_sleep_duration_percentage rem_sleep_duration_is_ideal  \
```

0	0.206306	-1
1	0.206298	-1
2	0.213174	-1
3	0.199029	-1
4	0.171171	-1
5	0.161563	-1
6	0.309953	1
7	0.306054	1
8	0.297921	1
9	0.165112	-1
10	0.164384	-1
11	0.141468	-1
12	0.280061	1
13	0.230558	0
14	0.164534	-1
15	0.243776	0
16	0.185388	-1
17	0.218750	-1
18	0.279698	1
19	0.151384	-1
20	0.234323	0
21	0.101423	-1
22	0.229265	-1
23	0.121186	-1
24	0.211982	-1
25	0.220238	-1
26	0.199017	-1
27	0.203459	-1
28	0.185759	-1
29	0.206936	-1
30	0.226361	-1
31	0.198404	-1
32	0.198612	-1
33	0.182365	-1
34	0.163934	-1
35	0.146047	-1
36	0.212661	-1
37	0.185328	-1
38	0.217520	-1
39	0.165992	-1
40	0.203455	-1
41	0.245473	0
42	0.226190	-1
43	0.146652	-1
44	0.163832	-1
45	0.183333	-1
46	0.279630	1

	total_sleep_duration_hr	total_sleep_duration_is_ideal
0	9.250000	1
1	7.675000	0
2	6.958333	-1
3	8.583333	0
4	9.250000	1
5	7.891667	0
6	8.791667	0
7	7.433333	0
8	10.825000	1
9	8.933333	0
10	7.300000	0
11	9.425000	1
12	5.475000	-1
13	9.108333	1
14	9.116667	1
15	8.033333	0
16	9.125000	1
17	7.466667	0
18	7.716667	0
19	9.633333	1
20	5.050000	-1
21	4.683333	-1
22	12.358333	1
23	9.833333	1
24	9.041667	1
25	8.400000	0
26	6.783333	-1
27	8.191667	0
28	8.075000	0
29	7.208333	0
30	8.725000	0
31	7.308333	0
32	9.608333	1
33	8.316667	0
34	8.641667	0
35	8.958333	0
36	8.425000	0
37	8.633333	0
38	8.466667	0
39	8.233333	0
40	8.683333	0
41	8.283333	0
42	9.100000	1
43	7.841667	0
44	9.308333	1

```
45                 9.000000                                0
46                 9.000000                                0

[47 rows x 27 columns]
```

[483]:
```
corr = daily_df[['deep_sleep_duration_is_ideal','rem_sleep_duration_is_ideal',␣
 ↪'light_sleep_duration_is_ideal',␣
 ↪'total_sleep_duration_is_ideal','day_label_num']].corr('spearman')

fig = px.imshow(
        corr,
        text_auto=True,  # Display correlation values on the heatmap
        color_continuous_scale='RdBu',  # Choose a diverging color scale
        zmin=-1, zmax=1,  # Set the color scale range for correlations
        title="Correlation Heatmap",
        width=700,
        height=600
)

fig.show()
```

[484]:
```
daily_df[['deep_sleep_duration_is_ideal','rem_sleep_duration_is_ideal',␣
 ↪'light_sleep_duration_is_ideal',␣
 ↪'total_sleep_duration_is_ideal','day_label_num']].
 ↪corr('spearman')['day_label_num']
```

[484]:
```
deep_sleep_duration_is_ideal     0.228522
rem_sleep_duration_is_ideal     -0.195293
light_sleep_duration_is_ideal    0.000000
total_sleep_duration_is_ideal   -0.064307
day_label_num                    1.000000
Name: day_label_num, dtype: float64
```

Variable	Correlation with day_label_num	Interpretation
deep_sleep_duration_is_ideal	0.2285	Positive correlation → higher "ideal deep sleep" values tend to occur on better days. Since your labels are -1 (too little), 0 (ideal), 1 (too much), this means **having more deep sleep than the lower bound** tends to coincide with better days.
rem_sleep_duration_is_ideal	-0.1953	Negative correlation → higher rem_sleep_is_ideal values are associated with worse days. Since -1 = too little, 0 = ideal, 1 = too much, this suggests that **having "too much" REM sleep** might occur on worse days, or conversely, days with "too little" REM tend to be better.
light_sleep_duration_is_ideal	0.0	No correlation → light sleep percentage doesn't relate to your day labels.
total_sleep_duration_is_ideal	-0.0643	Slight negative correlation → being outside the ideal total sleep range slightly associates with better days, but it's very weak.

[485]:
```
all_days = sorted(sleep_df["day"].unique())
tmp = []

day_to_hrv = {
        d: hrv for d, hrv in zip(sleep_df["day"], sleep_df["average_hrv"])
}

day_to_max_hr = {
        pd.Timestamp(d): hr for d, hr in zip(daily_df["day"],␣
 ↪daily_df["total_max_hr"])
}

day_to_workout_duration = {
        pd.Timestamp(d): duration for d, duration in zip(daily_df["day"],␣
 ↪daily_df["total_duration"])
}

for day in all_days:

    day_before = day - timedelta(days=1)
    day_after  = day + timedelta(days=1)

    tmp.append({
        "day": day,
        "hrv_day_before": day_to_hrv.get(day_before),
        "hrv_day_of": day_to_hrv.get(day),
        "hrv_day_after": day_to_hrv.get(day_after),
        "total_exercise_duration_minutes": day_to_workout_duration.get(day, 0),
        "max_exercise_hr": day_to_max_hr.get(day, 0),
    })

hrv_ex_df = pd.DataFrame(tmp)
hrv_ex_df.to_markdown('tb.md')
```

[486]:
```
fig = go.Figure()


fig.add_trace(go.Scatter(
    x=hrv_ex_df['day'],
    y=hrv_ex_df['hrv_day_of'],
    mode='markers+lines',
    name='HRV',
    line=dict(color='green', width=2),
    yaxis='y1'
))

fig.add_trace(go.Scatter(
```

```
    x=hrv_ex_df['day'],
    y=hrv_ex_df['total_exercise_duration_minutes'],
    mode='markers+lines',
    name='Exercise Duration',
    line=dict(color='grey', width=2),
    marker=dict(size=6),
    yaxis='y2'
))

fig.add_trace(go.Scatter(
    x=hrv_ex_df['day'],
    y=hrv_ex_df['max_exercise_hr'],
    mode='markers+lines',
    name='Max HR',
    line=dict(color='black', width=2),
    marker=dict(size=6),
    yaxis='y2'
))

# Shade good and hard days with clear full-day rectangles
for d in hard_days:
    fig.add_vrect(
        x0=d, x1=d + timedelta(days=1),
        fillcolor="red", opacity=0.2,
        layer="below", line_width=0,
    )

for d in good_days:
    fig.add_vrect(
        x0=d, x1=d + timedelta(days=1),
        fillcolor="green", opacity=0.2,
        layer="below", line_width=0,
    )

# Layout
fig.update_layout(
    title="HRV vs Exercise 'Difficulty'",
    xaxis=dict(title="Day"),
    yaxis=dict(
        title="HRV",
        side="left",
        showgrid=True
    ),
    yaxis2=dict(
        title="Max HR / Exercise Duration",
        overlaying="y",
        side="right",
```

```
        showgrid=False
    ),
    template="plotly_white",
    barmode='overlay'
)

fig.show()
```

[487]:
```
corr = hrv_ex_df.drop('day', axis=1).corr('spearman')


fig = px.imshow(
        corr,
        text_auto=True,  # Display correlation values on the heatmap
        color_continuous_scale='RdBu',  # Choose a diverging color scale
        zmin=-1, zmax=1,  # Set the color scale range for correlations
        title="Correlation Heatmap",
        width=700,
        height=600
)

fig.show()
```

[488]:
```
hrv_ex_df
```

[488]:
```
          day  hrv_day_before  hrv_day_of  hrv_day_after  \
0  2025-09-14             NaN        49.0           39.0
1  2025-09-15            49.0        39.0            NaN
2  2025-09-18             NaN        41.0           40.0
3  2025-09-19            41.0        40.0           40.0
4  2025-09-20            40.0        40.0           37.0
..        …              …          …              …
62 2025-11-17            41.0        43.0            NaN
63 2025-11-19             NaN        40.0           36.0
64 2025-11-20            40.0        36.0           43.0
65 2025-11-21            36.0        43.0           35.0
66 2025-11-22            43.0        35.0            NaN

    total_exercise_duration_minutes  max_exercise_hr
0                           0.00000             0.0
1                           0.00000             0.0
2                           0.00000             0.0
3                         182.23095           159.0
4                           0.00000             0.0
..                              …              …
62                          0.00000             0.0
63                          0.00000             0.0
```

```
64                          0.00000             0.0
65                          0.00000             0.0
66                        151.00000           132.0

[67 rows x 6 columns]
```

[489]:
```
fig = px.bar(workout_sessions_df,
                              x='day_x', y='workout_duration',␣
 ↪color='workout_type',
                              title='Workout Duration by Type')
fig.add_trace(go.Scatter(
    x=workout_sessions_df['day_x'],
    y=workout_sessions_df['max_workout_hr'],
    mode='markers+lines',
    name='Max Heart Rate'
))
fig.update_layout(xaxis_title='Day', yaxis_title='Duration (minutes)')
fig.show()
```

[490]:
```
all_days = sorted(sleep_df["day"].unique())
tmp = []

day_to_hrv = {
        d: hrv for d, hrv in zip(sleep_df["day"], sleep_df["average_hrv"])
}

day_to_max_hr = {
        pd.Timestamp(d): hr for d, hr in zip(daily_df["day"],␣
 ↪daily_df["total_max_hr"])
}

day_to_workout_duration = {
        pd.Timestamp(d): duration for d, duration in zip(daily_df["day"],␣
 ↪daily_df["total_duration"])
}


for day in all_days:

    day_before = day - timedelta(days=1)
    day_after  = day + timedelta(days=1)

    tmp.append({
        "day": day,
        "hrv_day_before": day_to_hrv.get(day_before),
        "hrv_day_of": day_to_hrv.get(day),
        "hrv_day_after": day_to_hrv.get(day_after),
```

```
        "total_exercise_duration_minutes": day_to_workout_duration.get(day, 0),
        "max_exercise_hr": day_to_max_hr.get(day, 0),
    })

hrv_ex_df = pd.DataFrame(tmp)
hrv_ex_df.to_markdown('tb.md')
```

[491]:
```
exercise_type = {
    'low_intensity': ['yoga', 'walking', 'houseWork'],
    'medium_intensity': ['strengthTraining', 'hiking', 'cycling',␣
 ↪'stairExercise'],
    'high_intensity': ['running', 'other']  # other = rucking
}

exercise_type_to_intensity = {
    ex: intensity
    for intensity, exercises in exercise_type.items()
    for ex in exercises
}
```

[492]:
```
# --- Aggregate workout sessions per day_before (sleep day) ---
agg_workouts = workout_sessions_df.groupby("day_x", as_index=False).agg({
    "workout_type": lambda x: list(x),
})

tmp_daily = daily_df.copy()
tmp_daily["day"] = tmp_daily["day"].apply(lambda x: pd.Timestamp(x))

tmp_daily = tmp_daily.merge(
    agg_workouts,
    left_on="day",
    right_on="day_x",
    how="left"
)

def dedup(exercises):
    exercise_type = {
        0: ['yoga', 'walking', 'houseWork'],
        1: ['strengthTraining', 'hiking', 'cycling'],
        2: ['running', 'other']  # other = rucking
    }
    if len(set(exercises)) == 1: # all the same exercise:
        return exercises[0]

    exercise_to_intensity = {ex: intensity for intensity, lst in exercise_type.
 ↪items() for ex in lst}
```

```
    # Return the exercise with the highest intensity
    return max(exercises, key=lambda x: exercise_to_intensity.get(x, -1))


# If multiple workouts, pick most intense one
tmp_daily["workout_type"] = tmp_daily["workout_type"].apply(
    lambda x: dedup(x)
)

tmp_daily["intensity_label"] = tmp_daily["workout_type"].apply(lambda x:␣
 ↪exercise_type_to_intensity.get(x, -1), 1)

# --- Select only the 3 main sleep stages ---
ideal_cols = [
    "deep_sleep_duration_is_ideal",
    "light_sleep_duration_is_ideal",
    "rem_sleep_duration_is_ideal"
]

# --- Keep only rows where ANY stage is non-ideal ---
nonideal_df = tmp_daily[
    tmp_daily[ideal_cols].isin([-1, 0, 1]).any(axis=1)
]

# --- Melt to long format ---
long_df = nonideal_df.melt(
    id_vars=["day", "intensity_label"],
    value_vars=ideal_cols,
    var_name="sleep_category",
    value_name="ideal_value"
)
# Map readable labels
long_df["ideal_value"] = long_df["ideal_value"].map({
    -1: "Too Little",
    0: "Ideal",
    1: "Too Much"
})

# Clean names
long_df["sleep_category"] = (
    long_df["sleep_category"]
    .str.replace("_duration_is_ideal", "", regex=False)
    .str.replace("_", " ")
    .str.title()
)

# --- Count by workout × sleep category × deviation direction ---
```

```
counts = (
    long_df.groupby(["intensity_label", "sleep_category", "ideal_value"])
    .size()
    .reset_index(name="count")
)

# --- PLOT ---
fig = px.bar(
    counts,
    x="intensity_label",
    y="count",
    color="ideal_value",
    facet_col="sleep_category",   # <<< THIS KEEPS REM / LIGHT / DEEP SEPARATE
    category_orders={"sleep_category": ["Deep Sleep", "Light Sleep", "Rem␣
 ↪Sleep"]},
    title="Non-Ideal Sleep Trends by Workout Type and Sleep Stage",
    labels={
        "intensity_label": "Workout Intensity",
        "count": "Count of Non-Ideal Nights",
        "ideal_value": "Deviation",
        "sleep_category": "Sleep Stage"
    }
)

fig.update_layout(
    height=600,
    barmode="stack"
)

fig.show()
```

```
/var/folders/y6/nzcpctnx7m5_79zgd1sv_ctr0000gn/T/ipykernel_94335/3548586916.py:3
6: FutureWarning:

the convert_dtype parameter is deprecated and will be removed in a future
version.  Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.
```

[493]:
```
counts = (
    long_df.groupby(["intensity_label", "sleep_category", "ideal_value"])
    .size()
    .reset_index(name="count")
)

#counts[counts['sleep_category'] =='Deep Sleep']#['count'].sum() # should be 75
```

```
# 75 exercises in workout_agg, so should have 75 intensity labels (regardless␣
 ↪of sleep_category)
counts
```

[493]:

	intensity_label	sleep_category	ideal_value	count
0	high_intensity	Deep Sleep	Ideal	2
1	high_intensity	Deep Sleep	Too Little	29
2	high_intensity	Light Sleep	Ideal	1
3	high_intensity	Light Sleep	Too Much	30
4	high_intensity	Rem Sleep	Ideal	3
5	high_intensity	Rem Sleep	Too Little	25
6	high_intensity	Rem Sleep	Too Much	3
7	low_intensity	Deep Sleep	Too Little	5
8	low_intensity	Light Sleep	Too Much	5
9	low_intensity	Rem Sleep	Ideal	1
10	low_intensity	Rem Sleep	Too Little	4
11	medium_intensity	Deep Sleep	Too Little	11
12	medium_intensity	Light Sleep	Too Much	11
13	medium_intensity	Rem Sleep	Too Little	8
14	medium_intensity	Rem Sleep	Too Much	3

[494]:
```
intensity_counts = {
    'high_intensity': 0,
    'medium_intensity': 0,
    'low_intensity': 0
}
for workout in agg_workouts['workout_type']:
    for w in workout:
        #print(w, exercise_type_to_intensity.get(w))
        intensity_counts[exercise_type_to_intensity.get(w)] += 1

intensity_counts
```

[494]:
```
{'high_intensity': 33, 'medium_intensity': 16, 'low_intensity': 26}
```

[504]:
```
counts.to_csv('counts.csv')
```

[495]:
```
# P( intensity_label | deep_sleep == Too Little )

bayes_all = []

for combo in counts[['sleep_category', 'ideal_value']].drop_duplicates().
 ↪to_dict(orient='records'):

    subset = counts[
        (counts.sleep_category == combo['sleep_category']) &
        (counts.ideal_value == combo['ideal_value'])
```

```
    ]

    rows = []

    for intensity in subset.itertuples():
        bayes = {
            'intensity_label': intensity.intensity_label,
            'sleep_category': intensity.sleep_category,
            'sleep_quality': intensity.ideal_value
        }
        # Prior: global intensity distribution
        bayes['prior'] = intensity_counts[intensity.intensity_label] /␣
 ↪sum(intensity_counts.values())

        # Likelihood: P(sleep_quality | intensity)
        bayes['likelihood'] = intensity.count / intensity_counts[intensity.
 ↪intensity_label]

        # Unnormalized posterior
        bayes['unnormalized_posterior'] = bayes['prior'] * bayes['likelihood']

        rows.append(bayes)

    # Convert rows for this sleep category to a DataFrame
    df_tmp = pd.DataFrame(rows)

    # Normalize within THIS sleep category/value combo
    df_tmp['normalized_posterior'] = (
        df_tmp['unnormalized_posterior'] /
        df_tmp['unnormalized_posterior'].sum()
    )

    bayes_all.append(df_tmp)

# Concatenate all results
bayes_simple = pd.concat(bayes_all, ignore_index=True)
bayes_simple
```

[495]:

	intensity_label	sleep_category	sleep_quality	prior	likelihood	\
0	high_intensity	Deep Sleep	Ideal	0.440000	0.060606	
1	high_intensity	Deep Sleep	Too Little	0.440000	0.878788	
2	low_intensity	Deep Sleep	Too Little	0.346667	0.192308	
3	medium_intensity	Deep Sleep	Too Little	0.213333	0.687500	
4	high_intensity	Light Sleep	Ideal	0.440000	0.030303	
5	high_intensity	Light Sleep	Too Much	0.440000	0.909091	
6	low_intensity	Light Sleep	Too Much	0.346667	0.192308	
7	medium_intensity	Light Sleep	Too Much	0.213333	0.687500	

```
8     high_intensity      Rem Sleep         Ideal  0.440000    0.090909
9      low_intensity      Rem Sleep         Ideal  0.346667    0.038462
10    high_intensity      Rem Sleep    Too Little  0.440000    0.757576
11     low_intensity      Rem Sleep    Too Little  0.346667    0.153846
12  medium_intensity      Rem Sleep    Too Little  0.213333    0.500000
13    high_intensity      Rem Sleep      Too Much  0.440000    0.090909
14  medium_intensity      Rem Sleep      Too Much  0.213333    0.187500

    unnormalized_posterior  normalized_posterior
0                 0.026667              1.000000
1                 0.386667              0.644444
2                 0.066667              0.111111
3                 0.146667              0.244444
4                 0.013333              1.000000
5                 0.400000              0.652174
6                 0.066667              0.108696
7                 0.146667              0.239130
8                 0.040000              0.750000
9                 0.013333              0.250000
10                0.333333              0.675676
11                0.053333              0.108108
12                0.106667              0.216216
13                0.040000              0.500000
14                0.040000              0.500000
```

[501]:
```
bayes_simple.to_csv('bayes_simple.csv')
```

If I had too little REM sleep, I have a 67% chance of doing high intensity exercise, 11% chance of doing low intensity, and 22% of medium intensity.

[496]:
```
intensity_counts
```

[496]:
```
{'high_intensity': 33, 'medium_intensity': 16, 'low_intensity': 26}
```

[497]:
```
quality_map = {
    -1: "Too Little",
     0: "Ideal",
     1: "Too Much"
}

df_counts = (
    tmp_daily
    .groupby(
        ["intensity_label",
         "rem_sleep_duration_is_ideal",
         "light_sleep_duration_is_ideal",
         "deep_sleep_duration_is_ideal"]
    )
```

```
    .size()
    .reset_index(name="count")
    .rename(columns={
        "rem_sleep_duration_is_ideal": "Rem Sleep Quality",
        "light_sleep_duration_is_ideal": "Light Sleep Quality",
        "deep_sleep_duration_is_ideal": "Deep Sleep Quality"
    })
)

# Apply mapping to the three sleep-quality columns
for col in ["Rem Sleep Quality", "Light Sleep Quality", "Deep Sleep Quality"]:
    df_counts[col] = df_counts[col].map(quality_map)

assert df_counts['count'].sum() == 47 # verify this adds to 47
df_counts
```

[497]:

	intensity_label	Rem Sleep Quality	Light Sleep Quality	Deep Sleep Quality	count
0	high_intensity	Too Little	Too Much	Too Little	23
1	high_intensity	Too Little	Too Much	Ideal	2
2	high_intensity	Ideal	Too Much	Too Little	3
3	high_intensity	Too Much	Ideal	Too Little	1
4	high_intensity	Too Much	Too Much	Too Little	2
5	low_intensity	Too Little	Too Much	Too Little	4
6	low_intensity	Ideal	Too Much	Too Little	1
7	medium_intensity	Too Little	Too Much	Too Little	8
8	medium_intensity	Too Much	Too Much	Too Little	3

[500]:
```
# Bayes, P( intensity level | Rem = Ideal, Light = Ideal, Deep Sleep = Too␣
 ↪Little)

prior_strength = 5  # adjust based on confidence in prior

combinations = df_counts[["Rem Sleep Quality", "Light Sleep Quality", "Deep␣
 ↪Sleep Quality"]].drop_duplicates().to_dict(orient='records')

n_all_workouts = df_counts['count'].sum()
```

```
# prior_strength * p of each intensity happening (regardless of sleep qualities)
alpha_priors = {
    'high_intensity': prior_strength * intensity_counts['high_intensity']/
 ↪n_all_workouts,
    'medium_intensity': prior_strength * intensity_counts['medium_intensity']/
 ↪n_all_workouts,
    'low_intensity': prior_strength * intensity_counts['low_intensity']/
 ↪n_all_workouts
}

# Posterior
bayes_all = []
for combo in combinations:
    df = df_counts[ # will get all intensity labels with this sleep quality␣
 ↪spread
        (df_counts['Rem Sleep Quality'] == combo['Rem Sleep Quality']) &
        (df_counts['Light Sleep Quality'] == combo['Light Sleep Quality']) &
        (df_counts['Deep Sleep Quality'] == combo['Deep Sleep Quality'])
    ][['intensity_label', 'count']].groupby('intensity_label').sum().
 ↪reset_index()

    bayes = {
        'total_alpha_post': 0
    }

    bayes.update(combo)

    for intensity_label in intensity_counts.keys():
        num_intensity_seen_in_df = df[df['intensity_label'] ==␣
 ↪intensity_label]['count'].item() if not df[df['intensity_label'] ==␣
 ↪intensity_label].empty else 0
        bayes[f'alpha_posteriors_parameter_{intensity_label}'] =␣
 ↪alpha_priors[intensity_label] + num_intensity_seen_in_df
        bayes['total_alpha_post'] +=␣
 ↪bayes[f'alpha_posteriors_parameter_{intensity_label}']

    # have to run seperately so we have the total alpha post
    for intensity_label in intensity_counts.keys():
        bayes[f'posterior_p_{intensity_label}'] =␣
 ↪bayes[f'alpha_posteriors_parameter_{intensity_label}'] /␣
 ↪bayes['total_alpha_post']

    bayes_all.append(bayes)
```

```
pd.DataFrame(bayes_all)[['Rem Sleep Quality', 'Light Sleep Quality', 'Deep␣
↪Sleep Quality', 'posterior_p_low_intensity', 'posterior_p_medium_intensity',␣
↪'posterior_p_high_intensity']].set_index(['Rem Sleep Quality', 'Light Sleep␣
↪Quality', 'Deep Sleep Quality']).to_csv('bayes_all.csv')
```

[503]:
```
df_counts.to_csv('df_counts.csv')
```

[ ]:
```

```