My Journey with Sleep: Powered by WHOOP

Code

Hide

```
Sys.setenv(RETICULATE_PYTHON = "/home/tliggett/.pyenv/versions/3.9.4/bin/python3.9")
library(reticulate)
```

Hide

```
reticulate::repl_python()
```

```
Error in `$.python.builtin.object`(x, name) :
  promise already under evaluation: recursive default argument reference or earlier prob
lems?
Python 3.9.4 (/home/tliggett/.pyenv/versions/3.9.4/bin/python3.9)
Reticulate 1.19 REPL -- A Python interpreter in R.
```

Hide

```
import requests  # for getting URL
import json  # for parsing json
from datetime import datetime  # datetime parsing
import pytz  # timezone adjusting
import csv  # for making csv files
import os
import pandas as pd
import numpy as np

save_directory = "~/workspace/coursework/data_visualization__COSC322/final_project" # ke
ep trailing slash
```

This is where the API requests from WHOOP are made.

Hide

```
def get_user_data_raw(username: str, password: str, start_date='2000-01-01T00:00:00.000
Z', end_date='2030-01-01T00:00:00.000Z', url='https://api-7.whoop.com/users/{}/cycles'):

  ################################################################
  # GET ACCESS TOKEN
  # Post credentials
  r = requests.post("https://api-7.whoop.com/oauth/token", json={
  "grant_type": "password",
  "issueRefresh": False,
  "password": password,
  "username": username })

  # Exit if fail
  if r.status_code != 200:
    print("Fail - Credentials rejected.")
    return 1

  print("Success - Credentials accepted")

  # Set userid/token variables
  userid = r.json()['user']['id']
  access_token = r.json()['access_token']

  # GET DATA
  # Download data
  url = url.format(userid)

  params = {
  'start': start_date,
  'end': end_date
  }

  headers = {
  'Authorization': 'bearer {}'.format(access_token)
  }
  r = requests.get(url, params=params, headers=headers)

  # Check if user/auth are accepted
  if r.status_code != 200:
    print("Fail - User ID / auth token rejected.")
    return 1

  print("Success - User ID / auth token accepted")
  # print(json.dumps(r.json()))
  data_raw = r.json()
  return data_raw


def get_user_data_df(username: str,
                     password: str,
                     start_date='2000-01-01T00:00:00.000Z',
                     end_date='2030-01-01T00:00:00.000Z',
                     url='https://api-7.whoop.com/users/{}/cycles'):
```

```
  # get raw whoop data
  data_raw = get_user_data_raw(username, password, start_date, end_date, url)
  # convert json to pandas df

  # df = pd.DataFrame.from_dict(data_raw)
  df = pd.json_normalize(data_raw)
  return df
```

Pull data from both me and Izzy (password are redacted)

Hide

```
tj_data = get_user_data_df("trevor.liggett@gmail.com", "StrongestHalls43")
```

```
Success - Credentials accepted
Success - User ID / auth token accepted
```

Hide

```
izzy_data = get_user_data_df("sommersizzy@gmail.com", "ilovetj")
```

```
Success - Credentials accepted
Success - User ID / auth token accepted
```

This data wrangling is necessary to get both the dates and many of the sleep statistics formatted correctly.

Hide

```
def first_date(row):
  return row['days'][0]

tj_data['date'] = tj_data.apply (lambda row: first_date(row), axis=1)
```

Hide

```
def get_sleep_stat(row, sleep_stat: str):
  if row['sleep.sleeps'] == []:
    return np.nan
  return row['sleep.sleeps'][0][sleep_stat]

tj_data['sleep.sws.duration'] = tj_data.apply (lambda row: get_sleep_stat(row, 'slowWave
SleepDuration'), axis=1)
tj_data['sleep.quality.duration'] = tj_data.apply (lambda row: get_sleep_stat(row, 'qual
ityDuration'), axis=1)
tj_data['sleep.light.duration'] = tj_data.apply (lambda row: get_sleep_stat(row, 'lightS
leepDuration'), axis=1)
tj_data['sleep.rem.duration'] = tj_data.apply (lambda row: get_sleep_stat(row, 'remSleep
Duration'), axis=1)
tj_data['sleep.wake.duration'] = tj_data.apply (lambda row: get_sleep_stat(row, 'wakeDur
ation'), axis=1)
tj_data['respiratoryRate'] = tj_data.apply (lambda row: get_sleep_stat(row, 'respiratory
Rate'), axis=1)
tj_data['sleep.efficiency'] = tj_data.apply (lambda row: get_sleep_stat(row, 'sleepEffic
iency'), axis=1)
tj_data['sleep.consistency'] = tj_data.apply (lambda row: get_sleep_stat(row, 'sleepCons
istency'), axis=1)
```

Hide

```
quit
library(ggplot2)
library(tidyverse)
```

```
Registered S3 methods overwritten by 'dbplyr':
  method         from
  print.tbl_lazy
  print.tbl_sql
── Attaching packages ─────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────── tidyverse 1.3.0 ──
✓ tibble  3.0.6     ✓ dplyr   1.0.5
✓ tidyr   1.1.2     ✓ stringr 1.4.0
✓ readr   1.4.0     ✓ forcats 0.5.1
✓ purrr   0.3.4
── Conflicts ─────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────── tidyverse_conflicts() ──
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Hide

```
library(lubridate)
```

```
Attaching package: ‘lubridate’

The following objects are masked from ‘package:base’:

    date, intersect, setdiff, union
```

Hide

```
library(patchwork) # To display 2 charts together
```

Hide

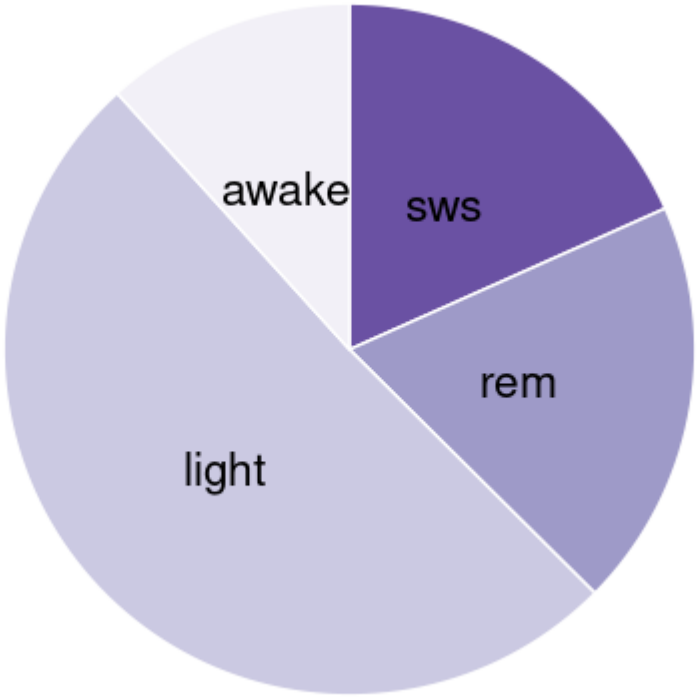
```
tj_data <- py$tj_data
tj_data$date <- as.Date(tj_data$date, format = "%Y-%m-%d")
```

SLEEP

These graphs display summaries of how much time I spent in the different stages of sleep.

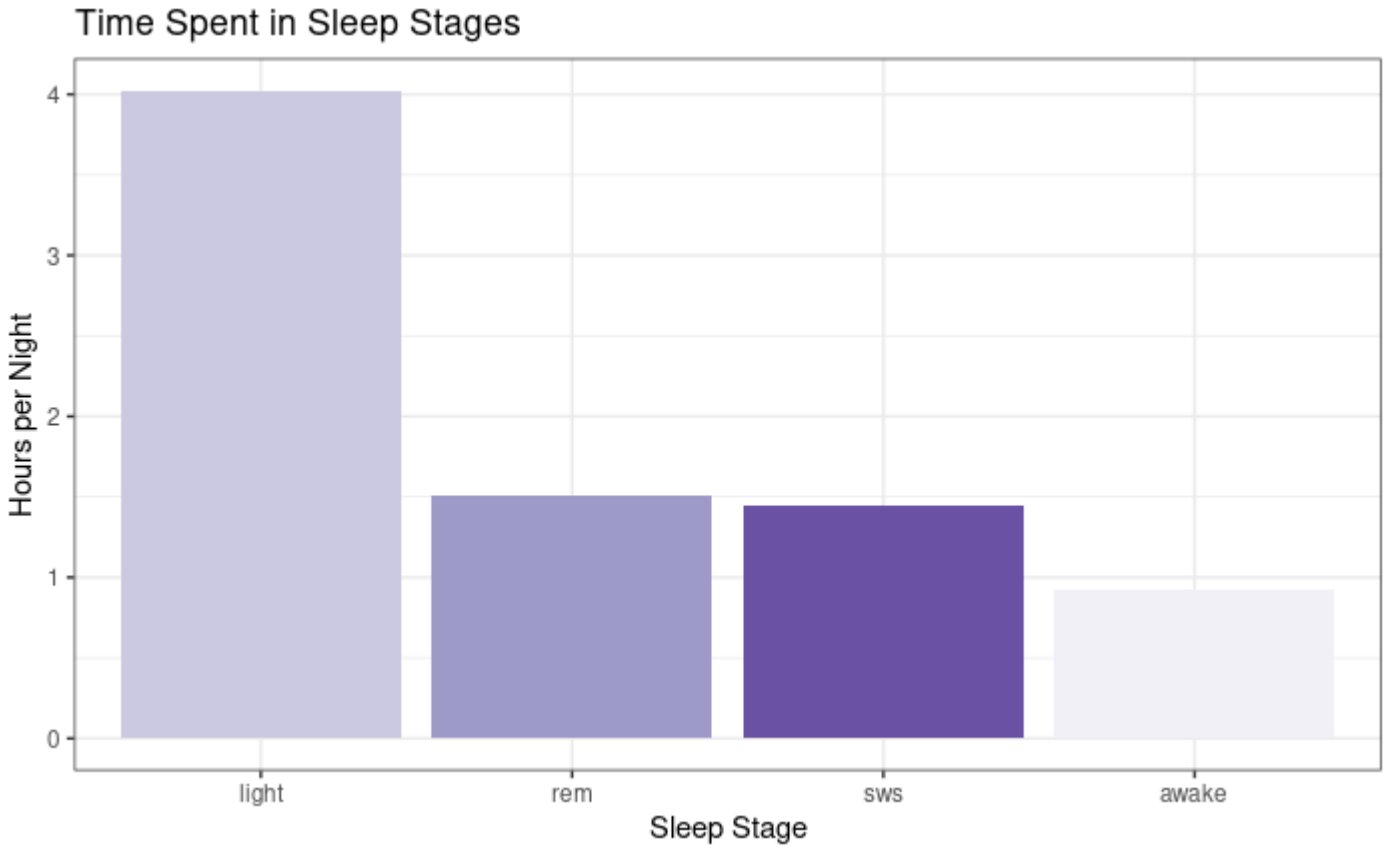
Hide

```
labels <- c("sws", "rem", "light", "awake")

sleep_stats <- tj_data %>%
  # filter(!is.na(as.numeric(sleep.sws.duration))) %>%
  summarize(sws = mean(sleep.sws.duration, na.rm = TRUE) / 3600000,
          rem = mean(sleep.rem.duration, na.rm = TRUE) / 3600000,
          light = mean(sleep.light.duration, na.rm = TRUE) / 3600000,
          wake = mean(sleep.wake.duration, na.rm = TRUE) / 3600000)
# Create Data
data <- data.frame(
  group=c("sws", "rem", "light", "awake"),
  value=c(sleep_stats$sws, sleep_stats$rem, sleep_stats$light, sleep_stats$wake)
)


# Compute the position of labels
data <- data %>%
  arrange(desc(group)) %>%
  mutate(prop = value / sum(data$value) *100) %>%
  mutate(ypos = cumsum(prop)- 0.5*prop )

# Basic piechart
ggplot(data, aes(x="", y=prop, fill=group)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  theme_void() +
  theme(legend.position="none") +
  geom_text(aes(y = ypos, label = group), color = "black", size=6) +
  scale_fill_brewer(palette="Purples")
```



Hide

```
# Bar chart version
ggplot(data = data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(group, -value), y = value, fill=
group)) +
    labs(title = "Time Spent in Sleep Stages", x="Sleep Stage", y="Hours per Night") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_brewer(palette="Purples")
```



STRAIN

Here I wrangled the workout data from both me and Izzy in python.

Hide

```
reticulate::repl_python()
```

```
Python 3.9.4 (/home/tliggett/.pyenv/versions/3.9.4/bin/python3.9)
Reticulate 1.19 REPL -- A Python interpreter in R.
```

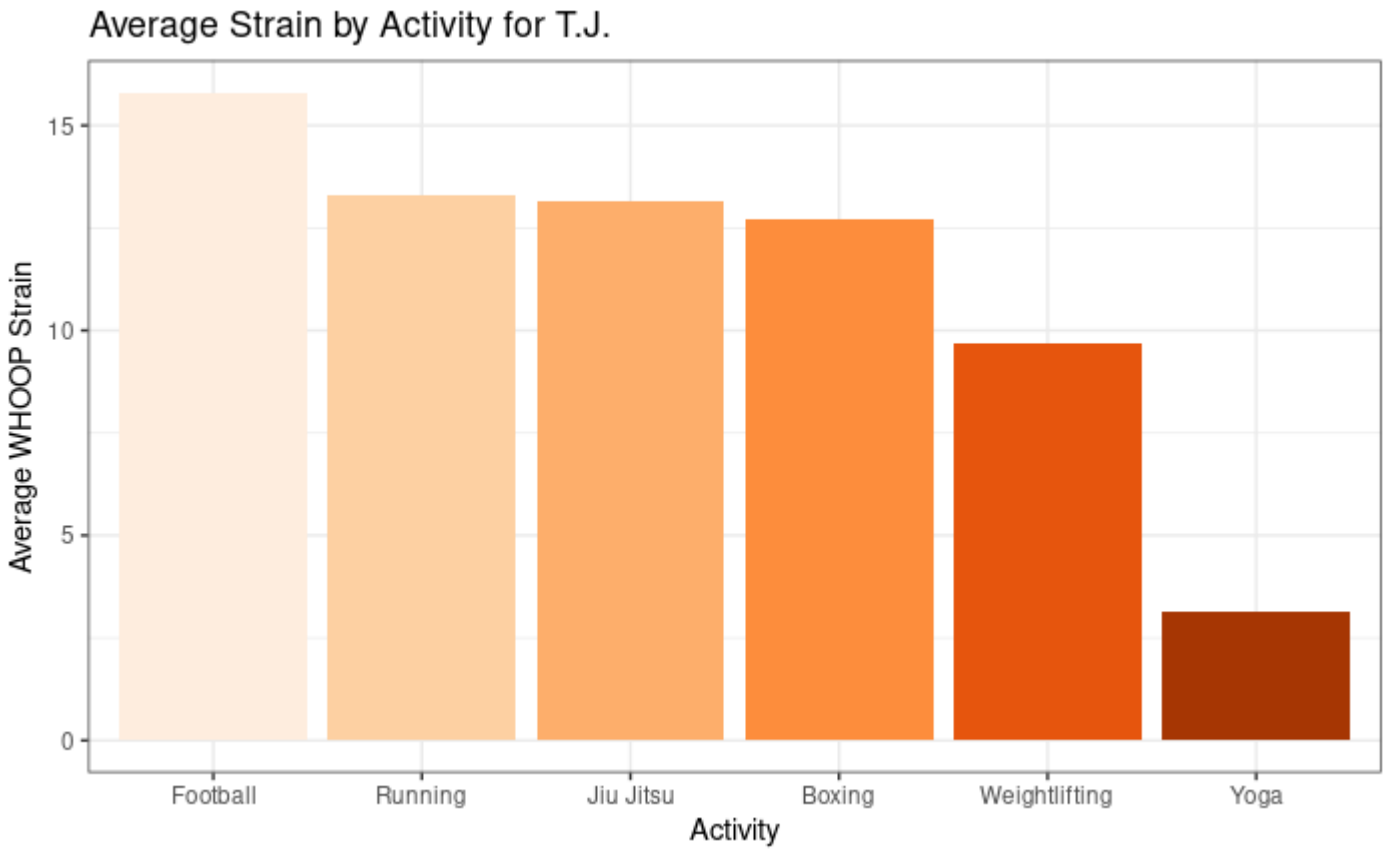
Hide

```
workouts = pd.DataFrame(columns=['sportId', 'cumulativeWorkoutStrain', 'score'])

for row in tj_data['strain.workouts']:
  for workout in row:
    workouts = workouts.append({'sportId': workout['sportId'],
                                'cumulativeWorkoutStrain': workout['cumulativeWorkoutStr
ain'],
                                'score': workout['score']
                                }, ignore_index=True)


izzy_workouts = pd.DataFrame(columns=['sportId', 'cumulativeWorkoutStrain', 'score'])

for row in izzy_data['strain.workouts']:
  for workout in row:
    izzy_workouts = izzy_workouts.append({'sportId': workout['sportId'],
                                'cumulativeWorkoutStrain': workout['cumulativeWorkoutStr
ain'],
                                'score': workout['score']
                                }, ignore_index=True)
```

Now to convert and pipe the data into meaningful summaries, and then graph the findings. Done in R

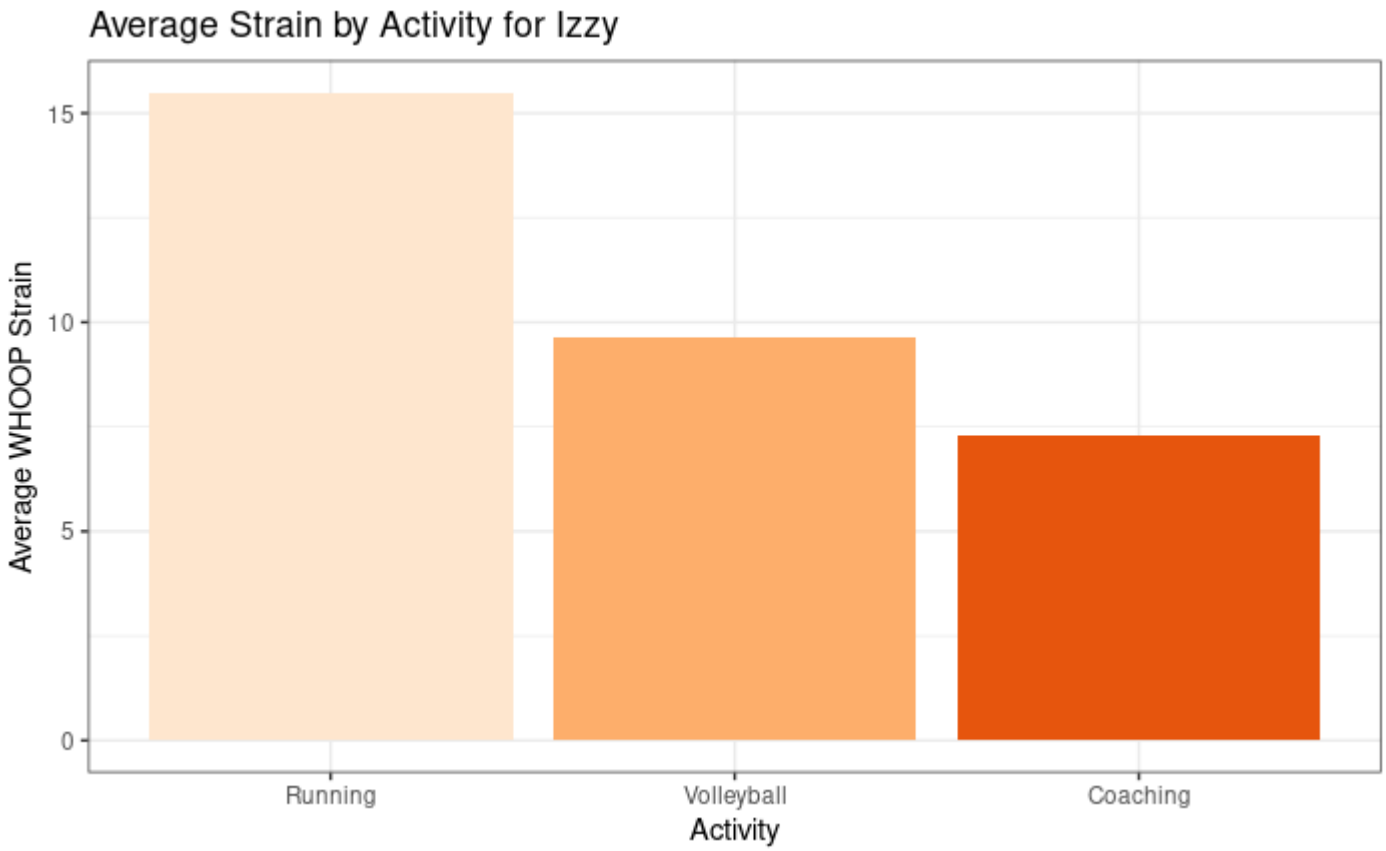
Hide

```
workouts <- py$workouts  %>%
  filter(sportId %in% c(21, 0, 98, 39, 45, 44)) %>%
  group_by(sportId) %>%
  summarize(strain = mean(score, na.rm = TRUE)) %>%
  arrange(desc(strain))

izzy_workouts <- py$izzy_workouts  %>%
  filter(sportId %in% c(0, 36, 87)) %>%
  group_by(sportId) %>%
  summarize(strain = mean(score, na.rm = TRUE)) %>%
  arrange(desc(strain))

workouts$sport <- c("Football", "Running", "Jiu Jitsu", "Boxing", "Weightlifting", "Yog
a")

izzy_workouts$sport <- c("Running",  "Volleyball", "Coaching")

ggplot(data = workouts) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(sport, -strain), y = strain, fil
l=reorder(sport, -strain))) +
    labs(title = "Average Strain by Activity for T.J.", x="Activity", y="Average WHOOP S
train") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_brewer(palette="Oranges")
```



Hide

```
ggplot(data = izzy_workouts) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(sport, -strain), y = strain, fil
l=reorder(sport, -strain))) +
    labs(title = "Average Strain by Activity for Izzy", x="Activity", y="Average WHOOP S
train") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_brewer(palette="Oranges")
```

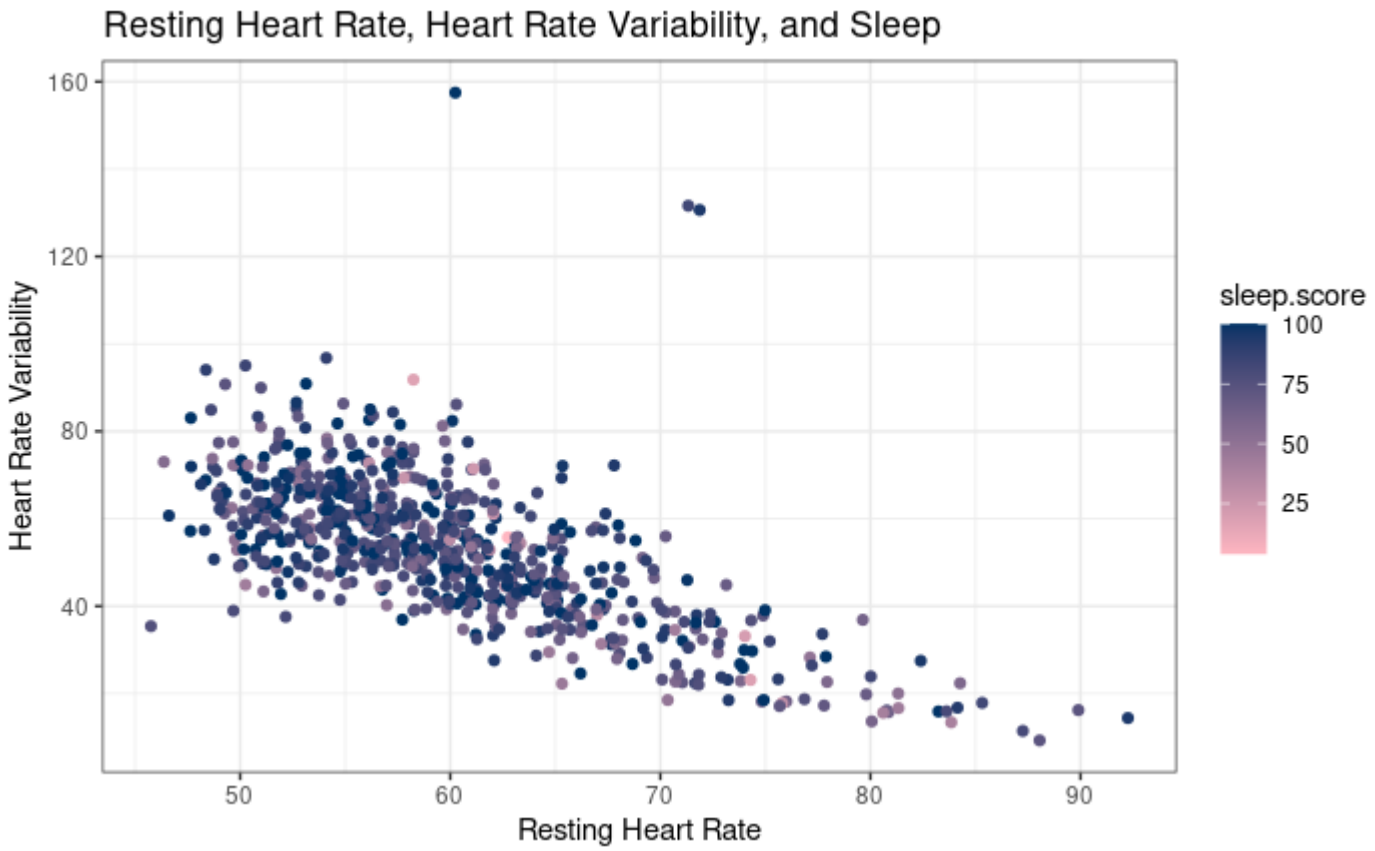


RECOVERY

Charts to show the relationships between recovery statistics.

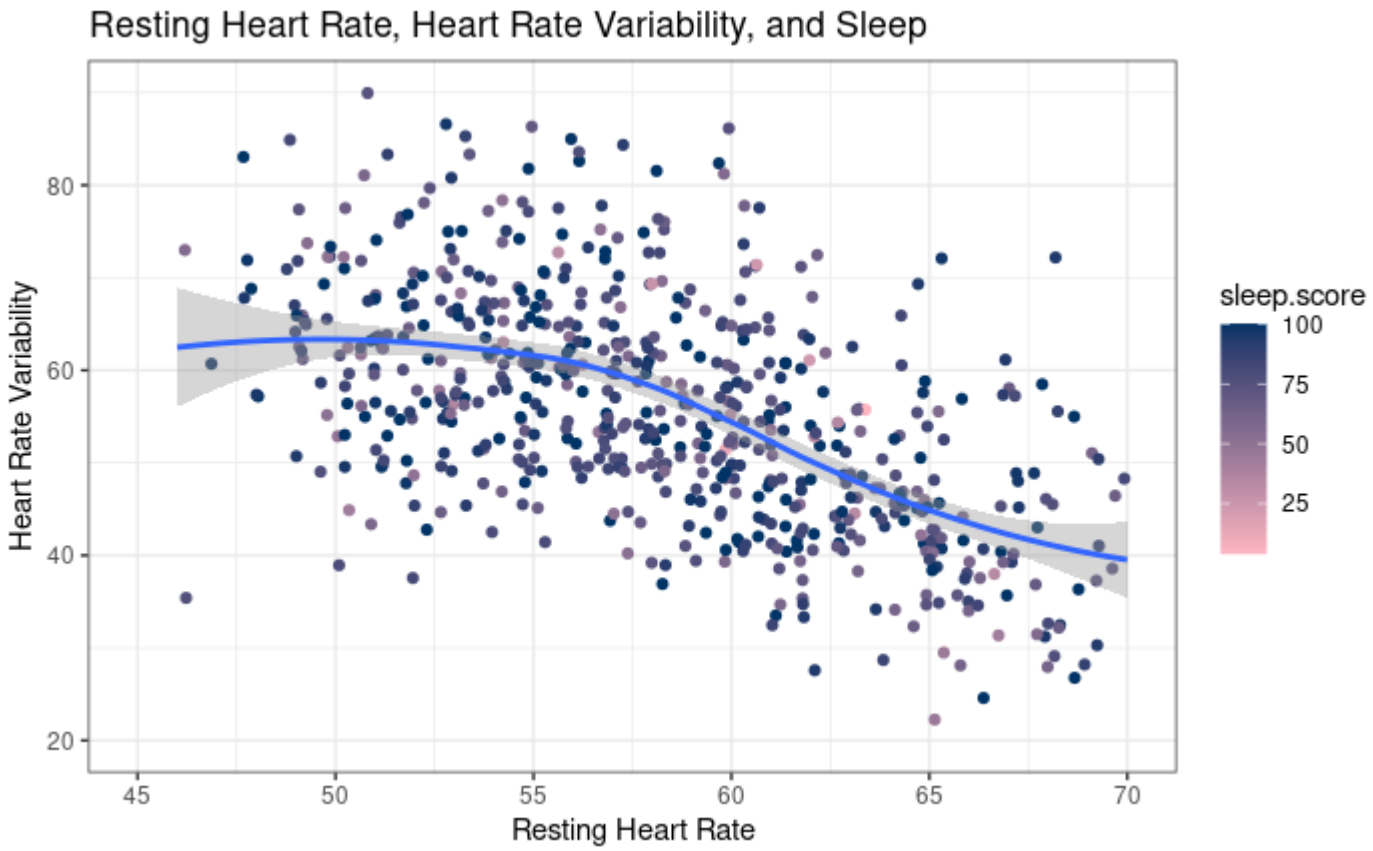
Hide

```
tj_data$recovery.hrv <- tj_data$recovery.heartRateVariabilityRmssd * 1000
tj_data$sleep.hours <- tj_data$sleep.qualityDuration / 3600000

ggplot(data = tj_data, aes(x = recovery.restingHeartRate, y = recovery.hrv, color=sleep.
score)) +
  geom_jitter() +
  scale_color_gradient(low="lightpink", high="#003366") +
  labs(title = "Resting Heart Rate, Heart Rate Variability, and Sleep", x="Resting Heart
Rate", y="Heart Rate Variability") +
  theme_bw()
```



Hide

```
ggplot(data = tj_data, aes(x = recovery.restingHeartRate, y = recovery.hrv, color=sleep.score)) +
  geom_jitter() +
  geom_smooth() +
  scale_color_gradient(low="lightpink", high="#003366") +
  labs(title = "Resting Heart Rate, Heart Rate Variability, and Sleep", x="Resting Heart Rate", y="Heart Rate Variability") +
  theme_bw() +
  xlim(45, 70) + ylim(20, 90)
```



Hide

```
NA
```

DAY OF WEEK SUMMARIES

The data wrangling for the dow summaries

Hide

```
tj_data$dayOfWeek <- weekdays(as.Date(tj_data$date))
tj_data$dayOfWeekNum <- wday(as.Date(tj_data$date))

weekdays <- c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturda
y")

weekday_data <- tj_data %>%
  group_by(dayOfWeekNum) %>%
  summarize(recovery = median(recovery.score, na.rm = TRUE), strain = median(strain.scor
e, na.rm = TRUE), sleep = mean(sleep.qualityDuration, na.rm = TRUE) / 3600000, sleep_nee
d = mean(sleep.needBreakdown.total, na.rm = TRUE) /3600000)

weekday_data$dow <- weekdays

weekday_data
```

dayOfWeekNum	recovery	strain	sleep	sleep_need	dow
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>

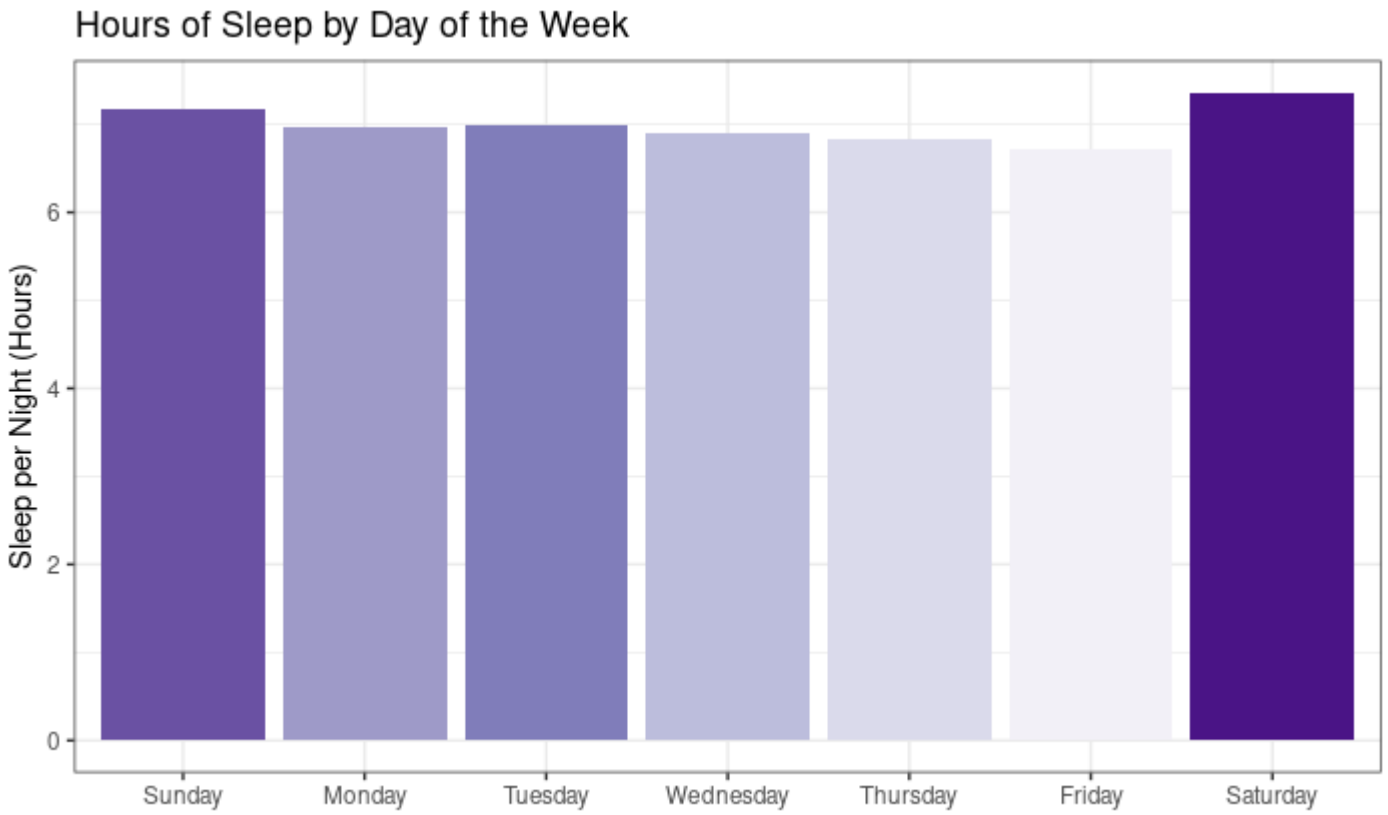
dayOfWeekNum <dbl>	recovery <dbl>	strain <dbl>	sleep <dbl>	sleep_need <dbl>	dow <chr>
1	55	6.152754	7.171962	8.085589	Sunday
2	75	12.958233	6.960574	8.235998	Monday
3	61	13.097235	6.984664	8.546159	Tuesday
4	56	12.552008	6.901612	8.559127	Wednesday
5	58	12.930203	6.826156	8.789615	Thursday
6	57	11.712450	6.726550	8.619979	Friday
7	52	8.908119	7.360054	8.515442	Saturday

7 rows

Pretty graphs to show trends in dow

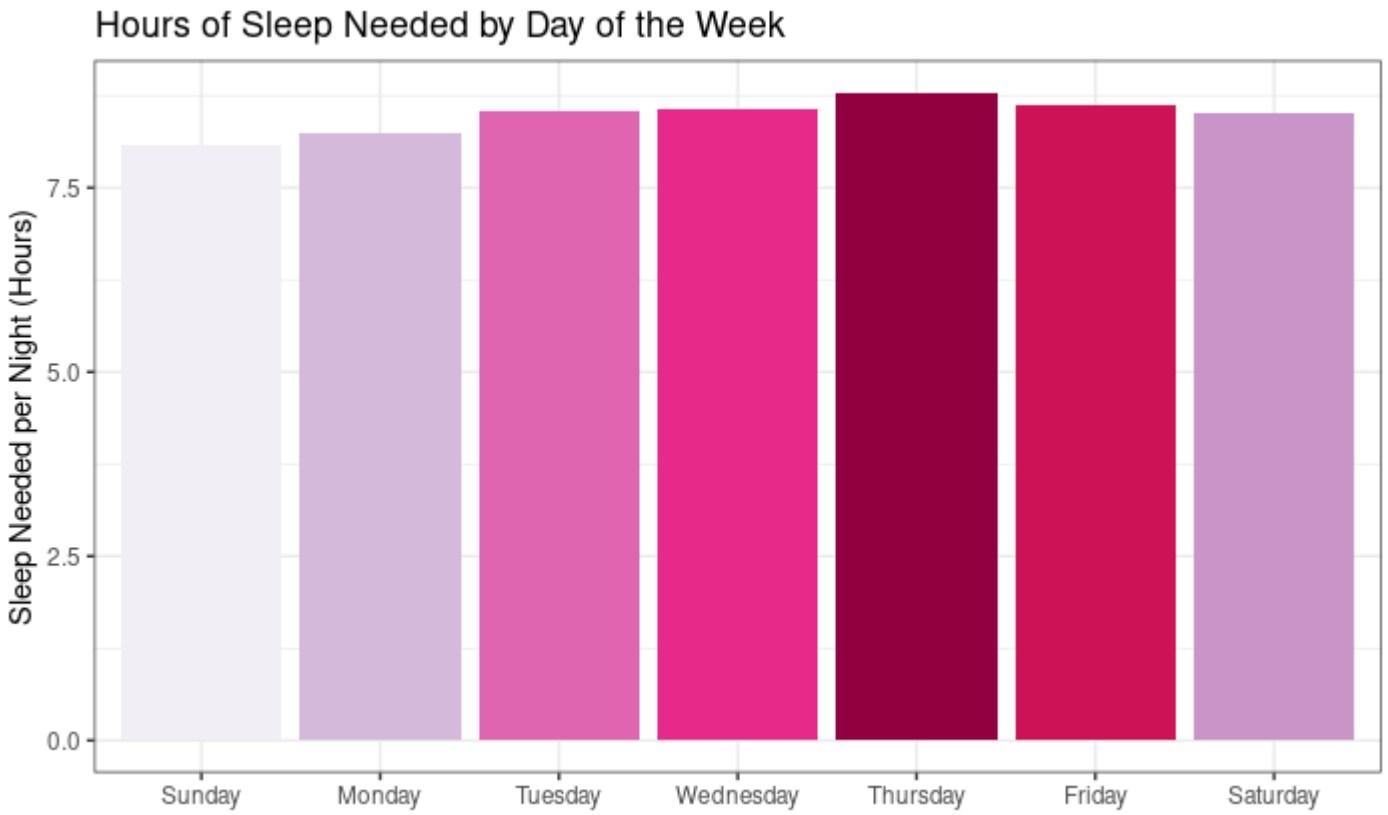
Hide

```
ggplot(data = weekday_data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(dow,dayOfWeekNum), y = sleep, fi
ll=reorder(dow, sleep))) +
    labs(title = "Hours of Sleep by Day of the Week",x = "", y="Sleep per Night (Hours)"
) +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_brewer(palette="Purples")
```



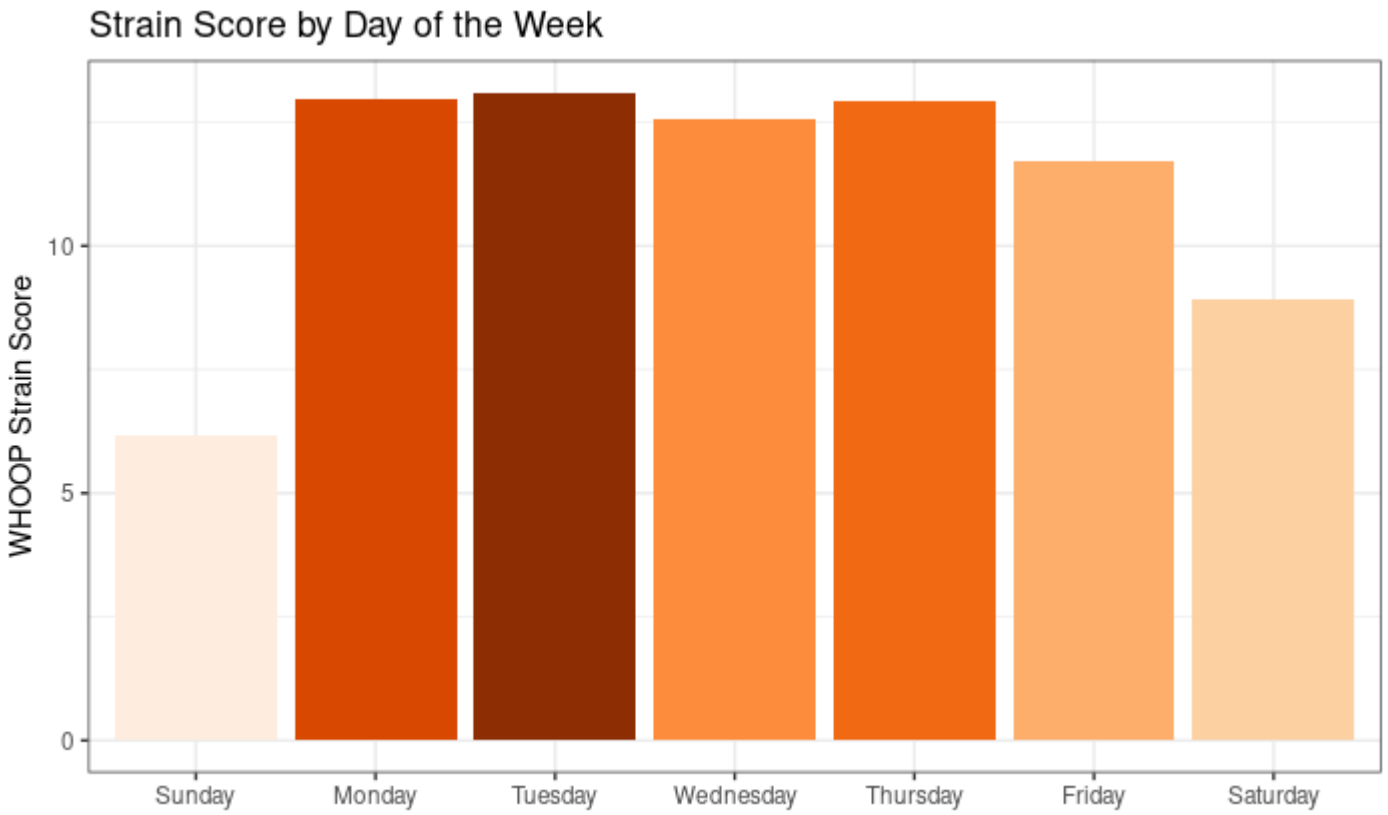
Hide

```
ggplot(data = weekday_data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(dow,dayOfWeekNum), y = sleep_nee
d, fill=reorder(dow,sleep_need))) +
    labs(title = "Hours of Sleep Needed by Day of the Week",x="", y="Sleep Needed per Ni
ght (Hours)") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_brewer(palette="PuRd")
```



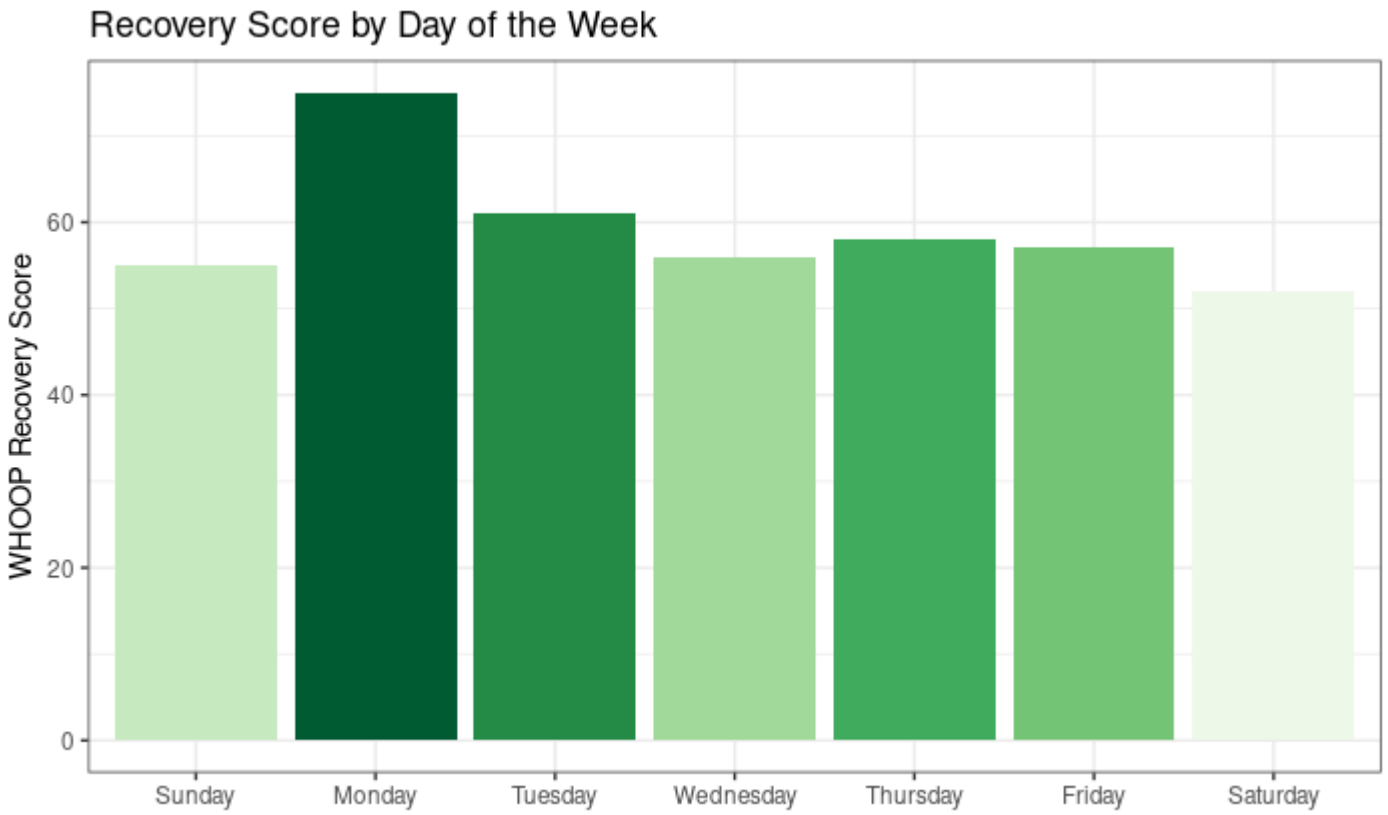
Hide

```
ggplot(data = weekday_data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(dow,dayOfWeekNum), y = strain, f
ill=reorder(dow,strain))) +
    labs(title = "Strain Score by Day of the Week", x="", y="WHOOP Strain Score") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_brewer(palette="Oranges")
```



Hide

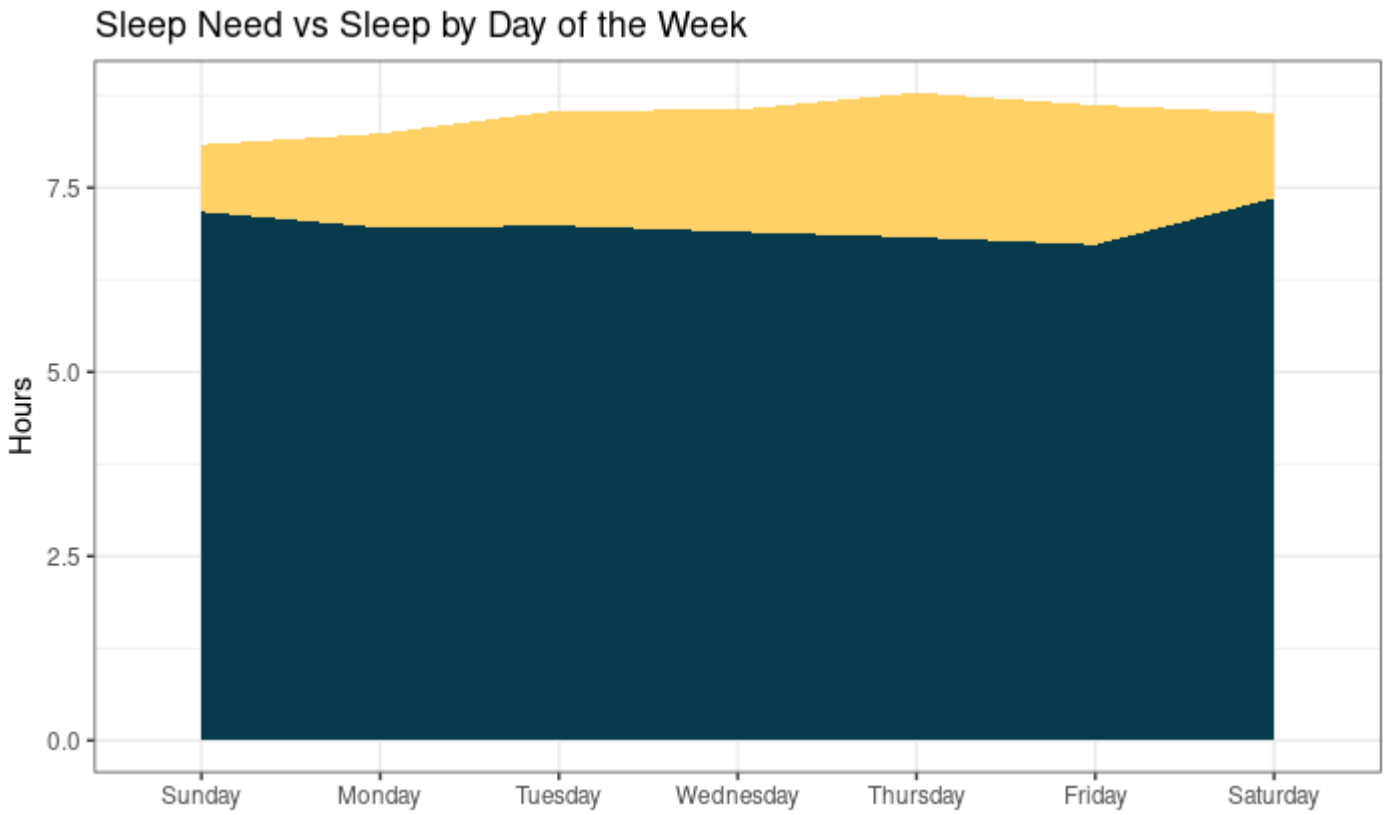
```
ggplot(data = weekday_data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(dow,dayOfWeekNum), y = recovery,
fill = reorder(dow,recovery))) +
    labs(title = "Recovery Score by Day of the Week", x="", y="WHOOP Recovery Score") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_brewer(palette="Greens")
```



Didn't end up using this but shows the sleep debt incurred in a typical week

Hide

```
ggplot(data = weekday_data, aes(x=reorder(dow,dayOfWeekNum), group = 1)) +
  geom_area(aes(y=sleep_need), fill="#FFD166") +
  geom_area(aes(y=sleep), fill = "#073B4C") +
  labs(title = "Sleep Need vs Sleep by Day of the Week",x="", y="Hours") +
    theme_bw() +
    theme(legend.position = "none")
```



MONTHLY SUMMARIES

Summarized data by month. Used in presentation but not the paper.

Hide

```
# The wrangling of the data
months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",
"Dec")

tj_data$month <- month(as.Date(tj_data$date))

month_data <- tj_data %>%
  filter(date >= "2020-01-01" & date <"2021-01-01") %>%
  group_by(month) %>%
  summarize(recovery = mean(recovery.score, na.rm = TRUE), strain = mean(strain.score, n
a.rm = TRUE), sleep = mean(sleep.qualityDuration, na.rm = TRUE) / 3600000, sleep_need =
 mean(sleep.needBreakdown.total, na.rm = TRUE) /3600000)

month_data$monthtext <- months

month_data
```

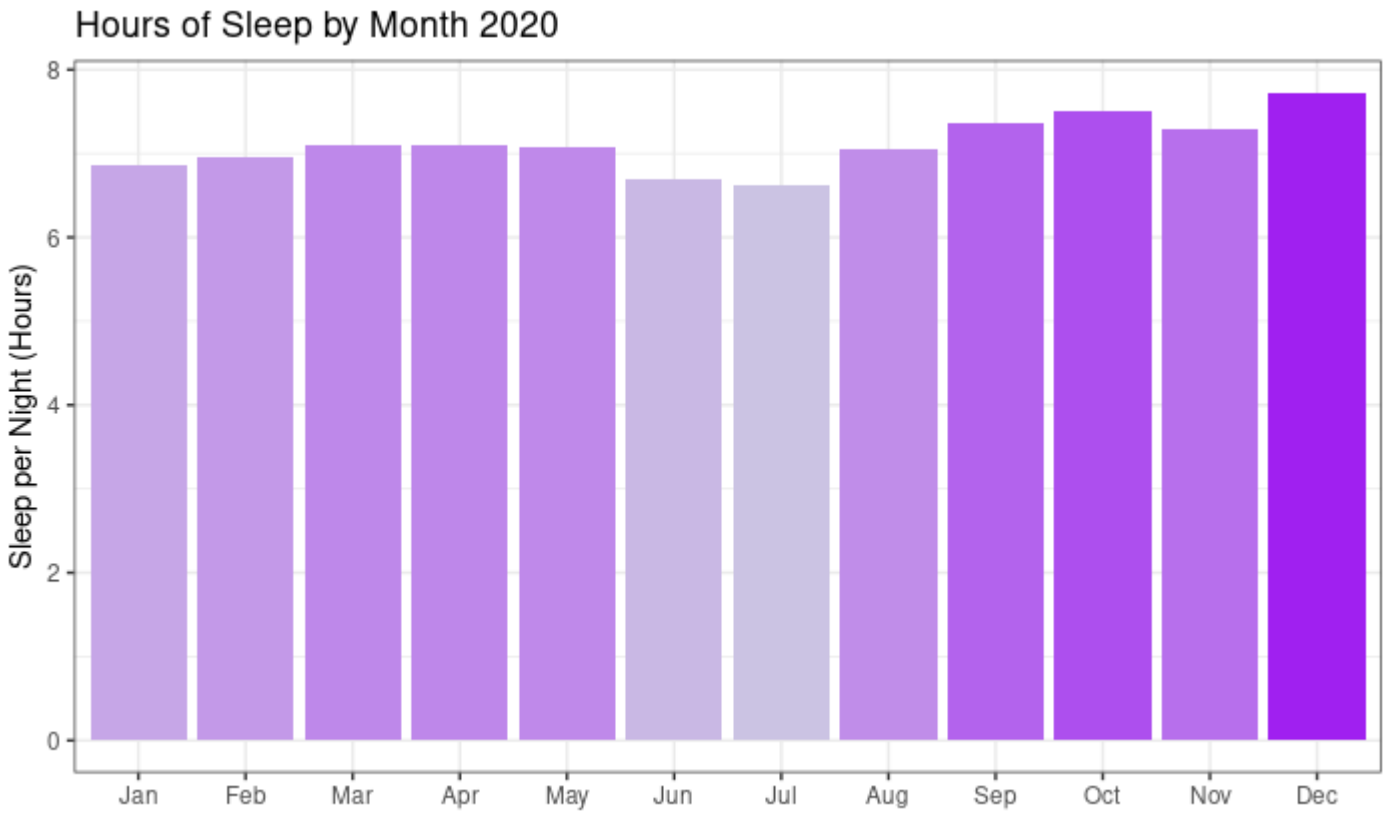
month	recovery	strain	sleep	sleep_need	monthtext
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1	57.89655	10.311614	6.853161	8.623407	Jan

month <dbl>	recovery <dbl>	strain <dbl>	sleep <dbl>	sleep_need <dbl>	monthtext <chr>
2	60.79310	11.564513	6.957157	9.060672	Feb
3	64.43478	9.287099	7.089860	8.715013	Mar
4	63.46667	10.555871	7.099221	8.544709	Apr
5	56.50000	9.943054	7.078087	8.599903	May
6	57.16667	12.440945	6.703877	8.430310	Jun
7	57.51613	11.075592	6.613951	8.609364	Jul
8	56.65517	11.536605	7.051709	8.700162	Aug
9	58.90000	10.154535	7.367824	7.831096	Sep
10	59.10000	11.038353	7.497224	8.665091	Oct

1-10 of 12 rows Previous **1** 2 Next

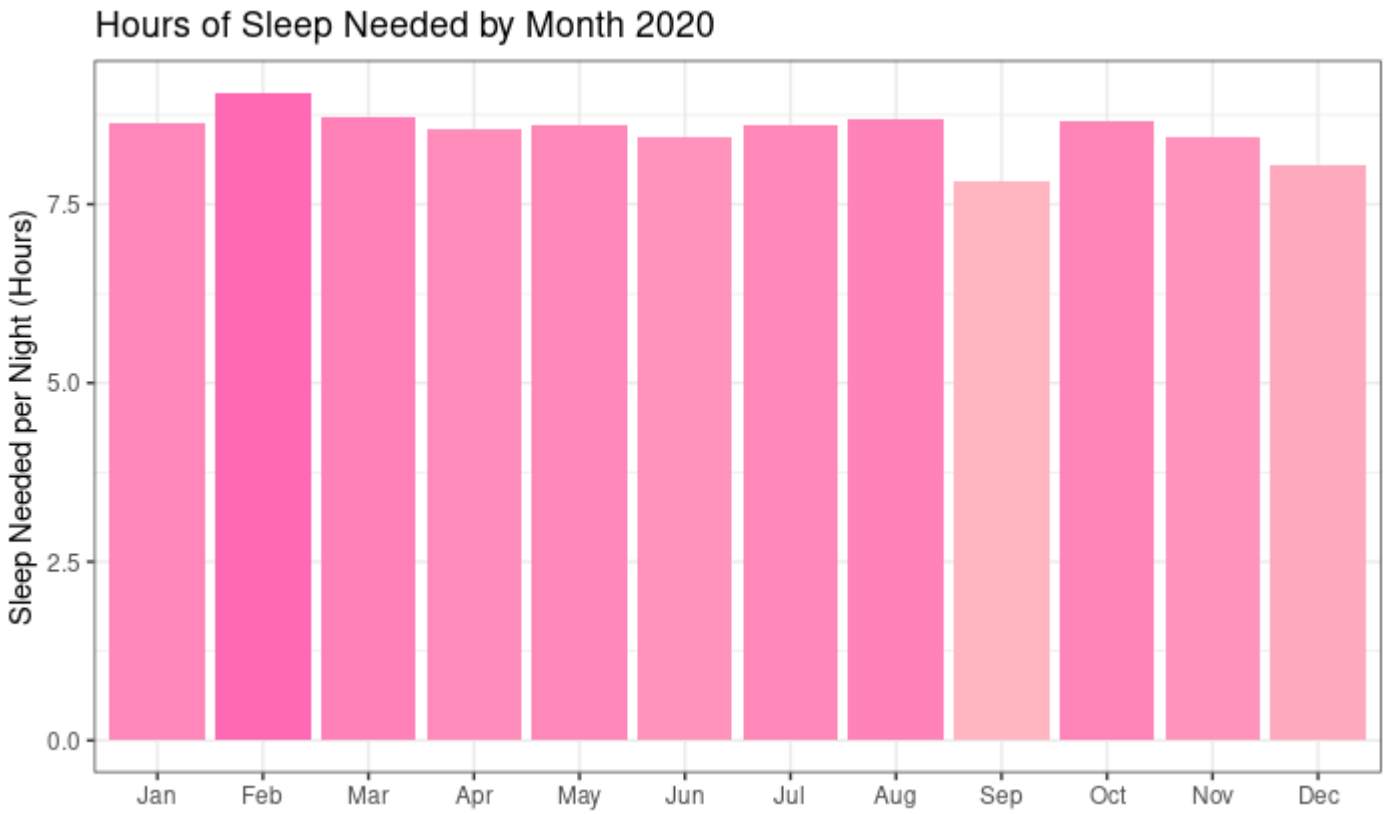
Hide

```
# Pretty graphs

ggplot(data = month_data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(monthtext, month), y = sleep, fi
ll=sleep)) +
    labs(title = "Hours of Sleep by Month 2020",x = "", y="Sleep per Night (Hours)") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_continuous(low="#CBC3E3", high="purple")
```



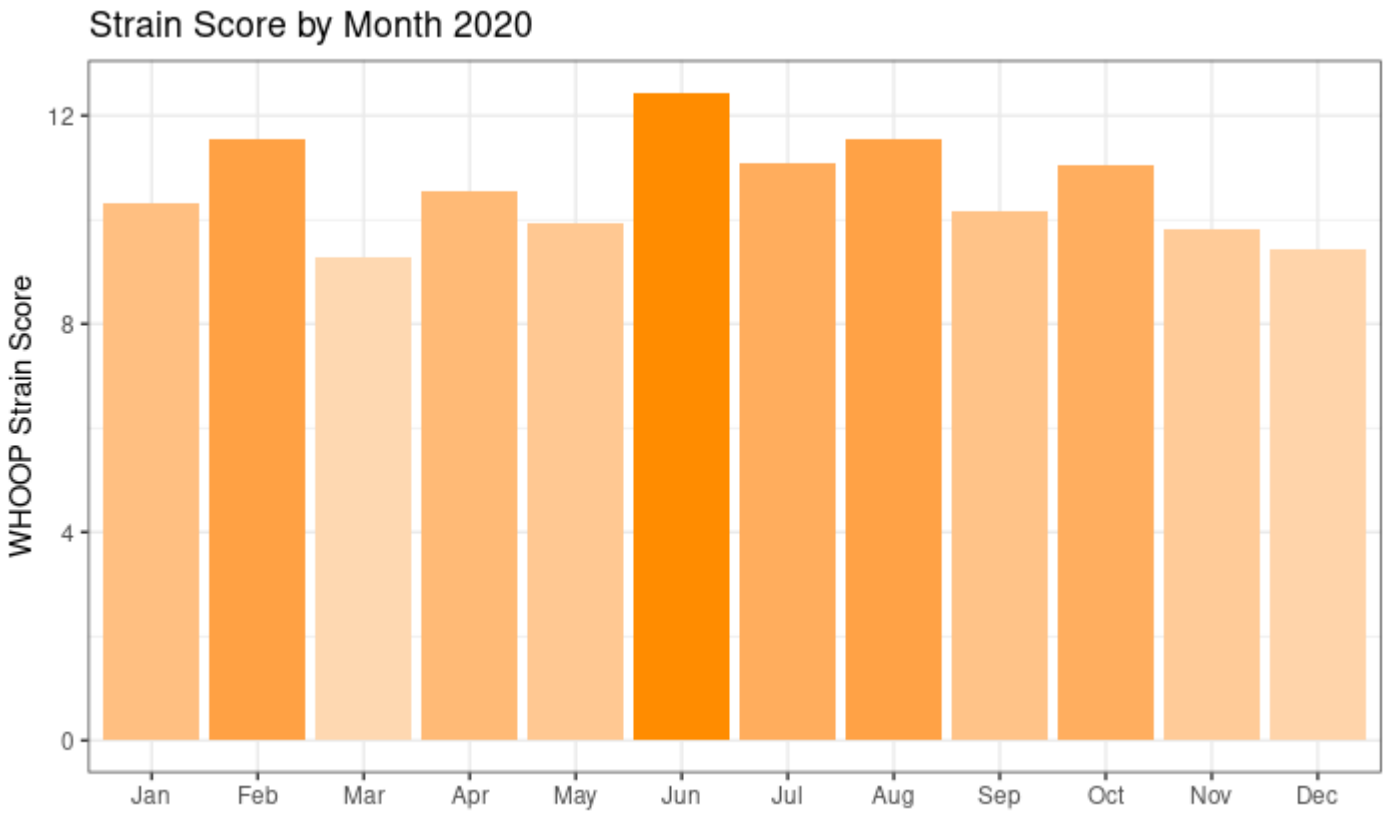
Hide

```
ggplot(data = month_data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(monthtext, month), y = sleep_nee
d, fill=sleep_need)) +
    labs(title = "Hours of Sleep Needed by Month 2020",x="", y="Sleep Needed per Night
 (Hours)") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_continuous(low="lightpink", high="hotpink")
```



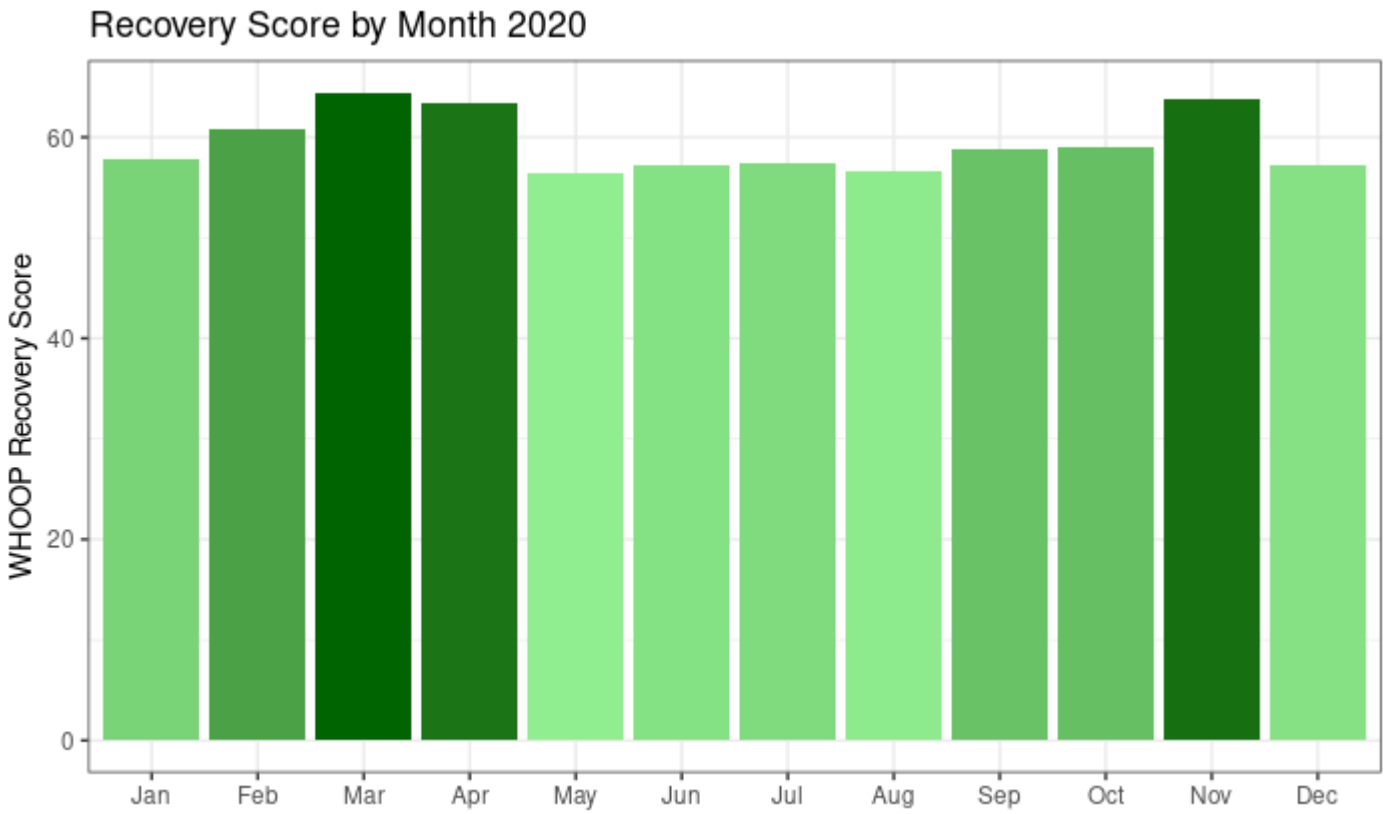
Hide

```
ggplot(data = month_data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(monthtext, month), y = strain, f
ill=strain)) +
    labs(title = "Strain Score by Month 2020", x="", y="WHOOP Strain Score") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_continuous(low="#fed8b1", high="darkorange")
```



Hide

```
ggplot(data = month_data) +
    geom_bar(stat = "identity", mapping = aes(x=reorder(monthtext, month), y = recovery,
fill=recovery)) +
    labs(title = "Recovery Score by Month 2020", x="", y="WHOOP Recovery Score") +
    theme_bw() +
    theme(legend.position = "none") +
    scale_fill_continuous(low="lightgreen", high="darkgreen")
```

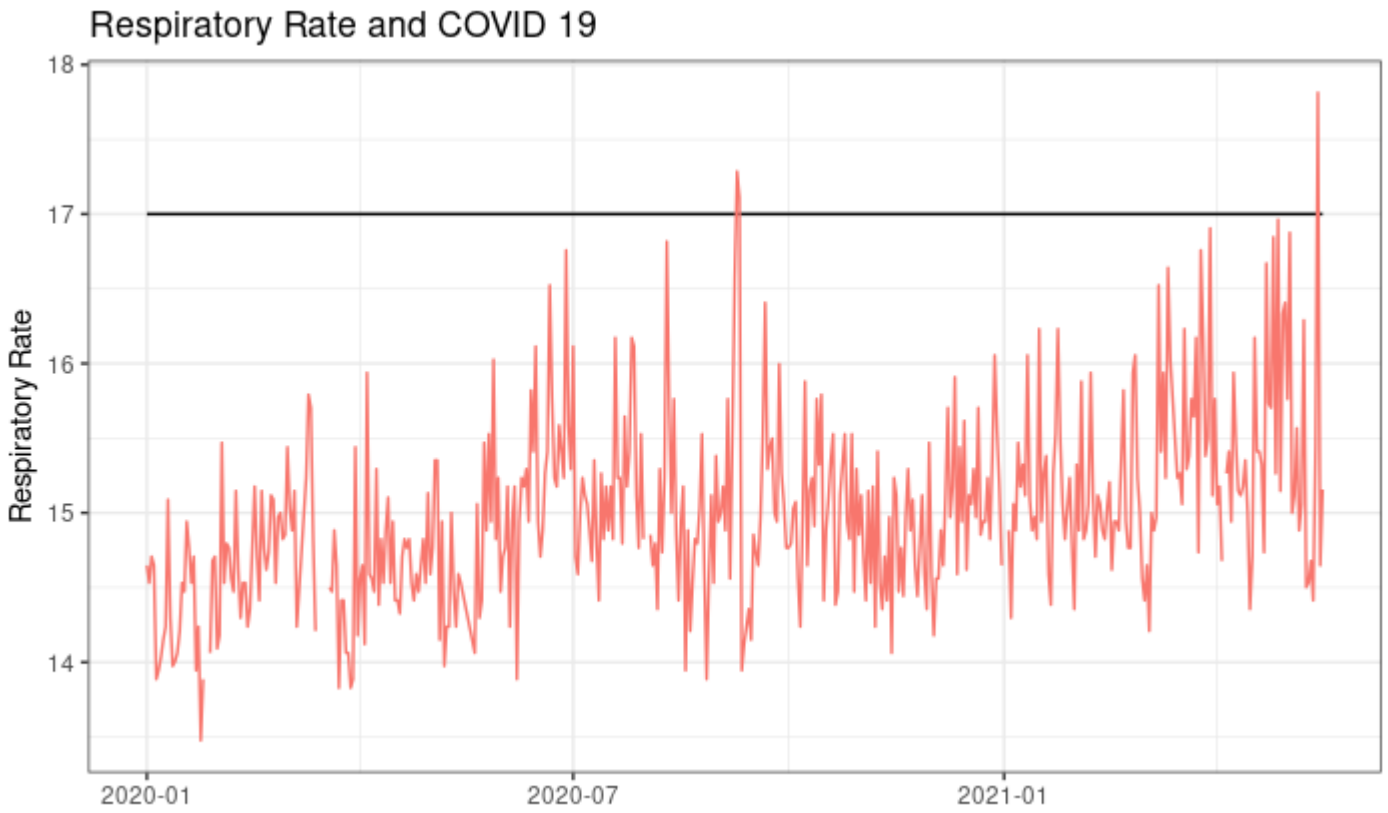


COVID 19 Stuff

For Appendix A. This shows how WHOOP was able to detect my COVID-19 Case as well as my vaccination… Crazy

Hide

```
rr_data <- tj_data %>%
  filter(date >= "2020-01-01")

ggplot(data = rr_data, aes(x = date)) +
  geom_line(aes(y=17)) +
  geom_line(aes(y= respiratoryRate, colour='#271033')) +
  labs(title = "Respiratory Rate and COVID 19", x="", y="Respiratory Rate") +
  theme_bw() +
  theme(legend.position = "none")
```



Weekly Average Summaries

Summarizing some of my major statistics with weekly averages over the years.

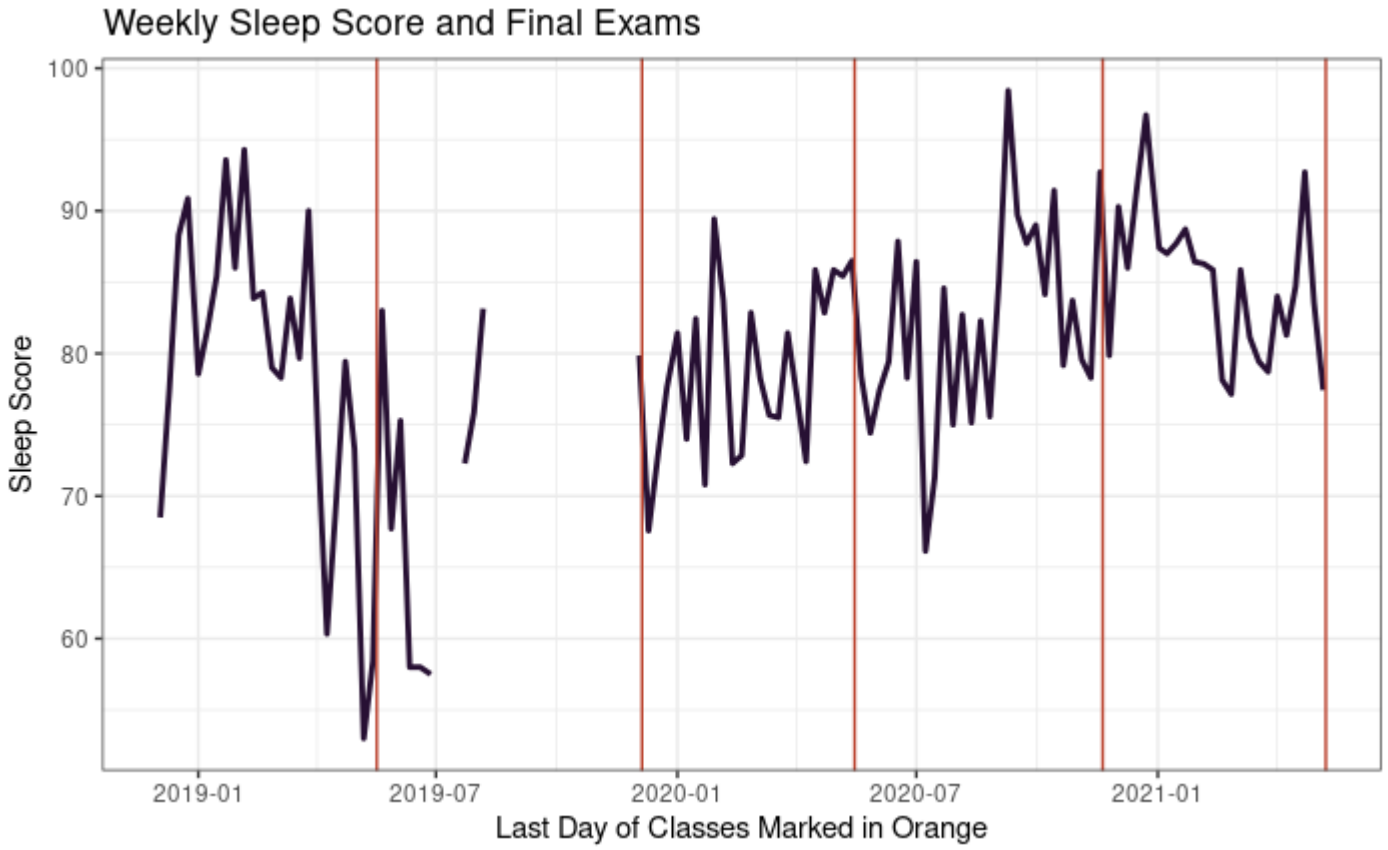
Hide

```
tj_data$week <- week(tj_data$date)
tj_data$year <- year(tj_data$date)

weekly_data <- tj_data %>%
  filter((!(week %in% c(53))) & (date <= '2021-05-13')) %>% # Take out week 53 because i
t's not full so crazy outliers
  group_by(year, week) %>%
  summarize(sleep = mean(sleep.score, na.rm = TRUE),
            date = min(date, na.rm = TRUE),
            efficiency = mean(sleep.efficiency, na.rm = TRUE),
            consistency = mean(sleep.consistency, na.rm = TRUE),
            duration = mean(sleep.quality.duration, na.rm = TRUE) / 3600000
            )
```

```
`summarise()` has grouped output by 'year'. You can override using the `.groups` argumen
t.
```

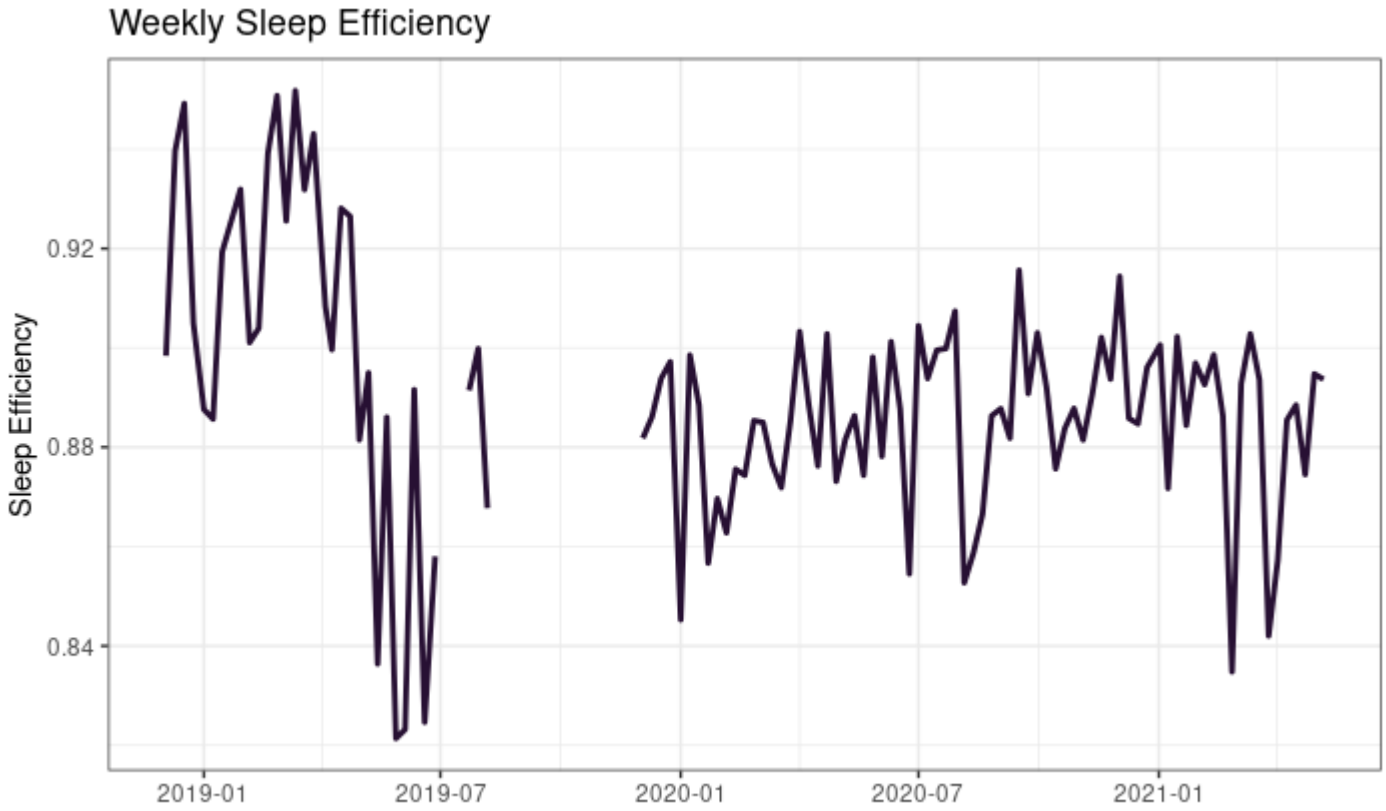
Hide

```
# These were the last day of classes each of these semesters
finals_dates <- c("2019-05-17","2019-12-05", "2020-05-15","2020-11-20", "2021-05-09")

ggplot(weekly_data, aes(x = date, y = sleep)) +
  geom_line(color="#271033", size = 1) +
  geom_vline(xintercept = as.numeric(as.Date(finals_dates, format = "%Y-%m-%d")), color=
"#BB4430") +
  labs(title = "Weekly Sleep Score and Final Exams", x="Last Day of Classes Marked in Or
ange", y="Sleep Score") +
  theme_bw() +
  theme(legend.position = "none")
```



Hide

```
ggplot(weekly_data, aes(x = date, y = efficiency)) +
  geom_line(color="#271033", size = 1) +
  labs(title = "Weekly Sleep Efficiency", x="", y="Sleep Efficiency") +
  theme_bw() +
  theme(legend.position = "none")
```

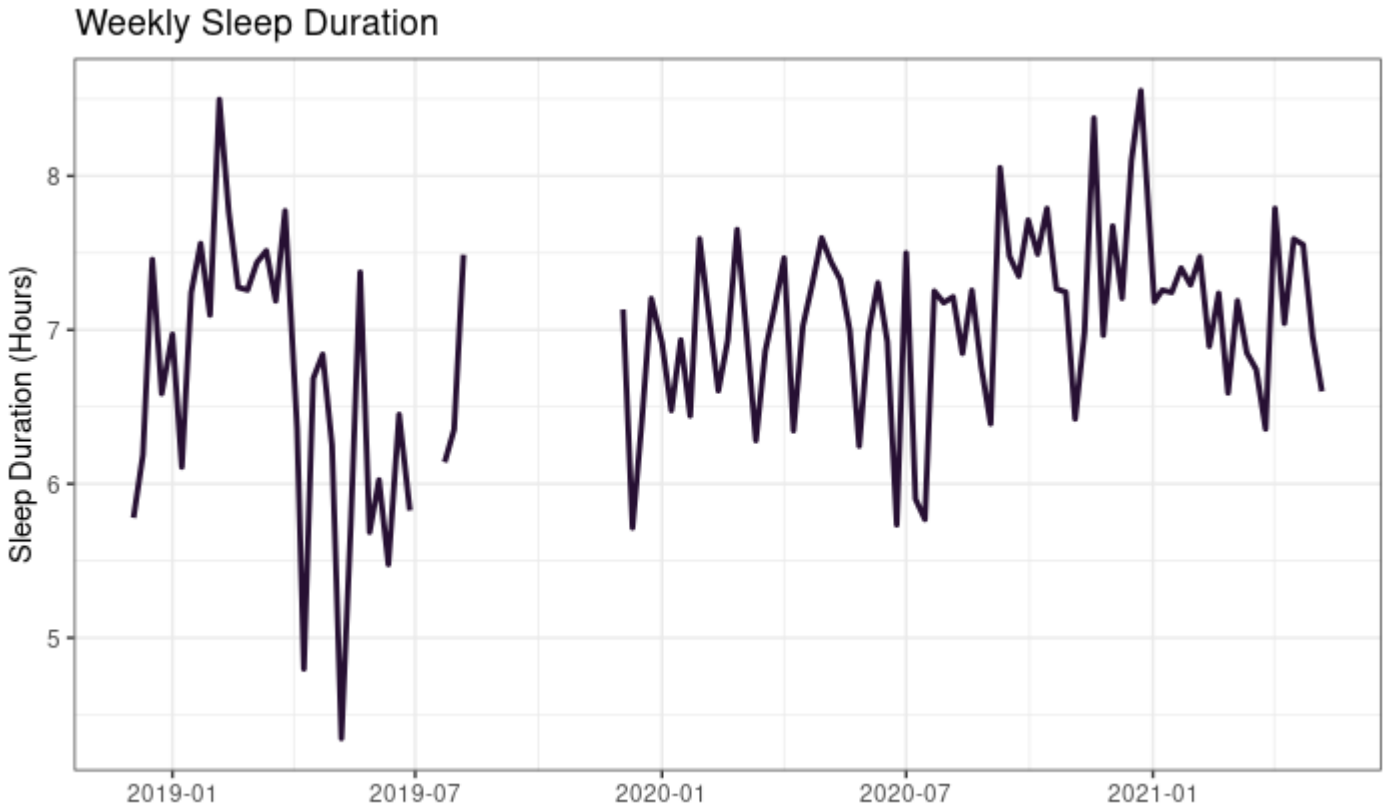


Hide

```
ggplot(weekly_data, aes(x = date, y = consistency)) +
  geom_line(color="#271033", size = 1) +
  labs(title = "Weekly Sleep Consistency", x="", y="Sleep Consistency") +
  theme_bw() +
  theme(legend.position = "none")
```

Hide

```
ggplot(weekly_data, aes(x = date, y = duration)) +
  geom_line(color="#271033", size = 1) +
  labs(title = "Weekly Sleep Duration", x="", y="Sleep Duration (Hours)") +
  theme_bw() +
  theme(legend.position = "none")
```


Weekly Sleep Duration
Sleep Duration (Hours)
8
7
6
5
2019-01
2019-07
2020-01
2020-07
2021-01