# Jake's Ultramarathon Training

Welcome to a typical Pluto notebook! This typically starts with

- Packages needed
- Initial data exploration

```
md"""
# Jake's Ultramarathon Training

Welcome to a typical Pluto notebook! This typically starts with

- Packages needed
- Initial data exploration
"""
```

```
begin
    using DataFrames
    using CSV
    using PyCall
    using Conda
    using Dates
    using Plots
    using Statistics
    using StatsBase
    using StatsPlots
end
```

## Packages Installed

- Pluto maintains it's own package environment per-notebook
- The checkmark means it is installed.
- The little cloud icon means it will be installed when ran

Selection deleted

```
md"""
## Packages Installed

- Pluto maintains it's own package environment per-notebook
- The checkmark means it is installed.
- The little cloud icon means it will be installed when ran
"""
```

| | variable | mean | min | median | max |
|---|---|---|---|---|---|
| 1 | :Timestamp | nothing | "2023-01-01 00:00:00" | nothing | "2023-04-05 00 |
| 2 | :Type | nothing | "Sleep Hours" | nothing | "Weight Pounds |
| 3 | :Value | 7.51523 | 0.0 | 3.05 | 205.3 |
| 4 | :ParsedTimestamp | nothing | 2023-01-01T00:00:00 | 2023-02-19T00:00:00 | 2023-04-05T00: |

```julia
begin
    workouts = CSV.read("workouts.csv", DataFrame);
    metrics = CSV.read("metrics.csv", DataFrame);

    # Start with some average time in each stage. Timestamp should be in a more usable
    # format.
    metrics.ParsedTimestamp = DateTime.(metrics.Timestamp, dateformat"y-m-d H:M:S")
    describe(metrics)
end
```

## First Steps

- Usually a good idea to describe the dataframe that you have
- Peform any cleaning needed

## New Columns

- Defined using the `.` format
- Uses broadcast over the existing Timestamp column that is a string
- Better for plotting and programmatic usage.

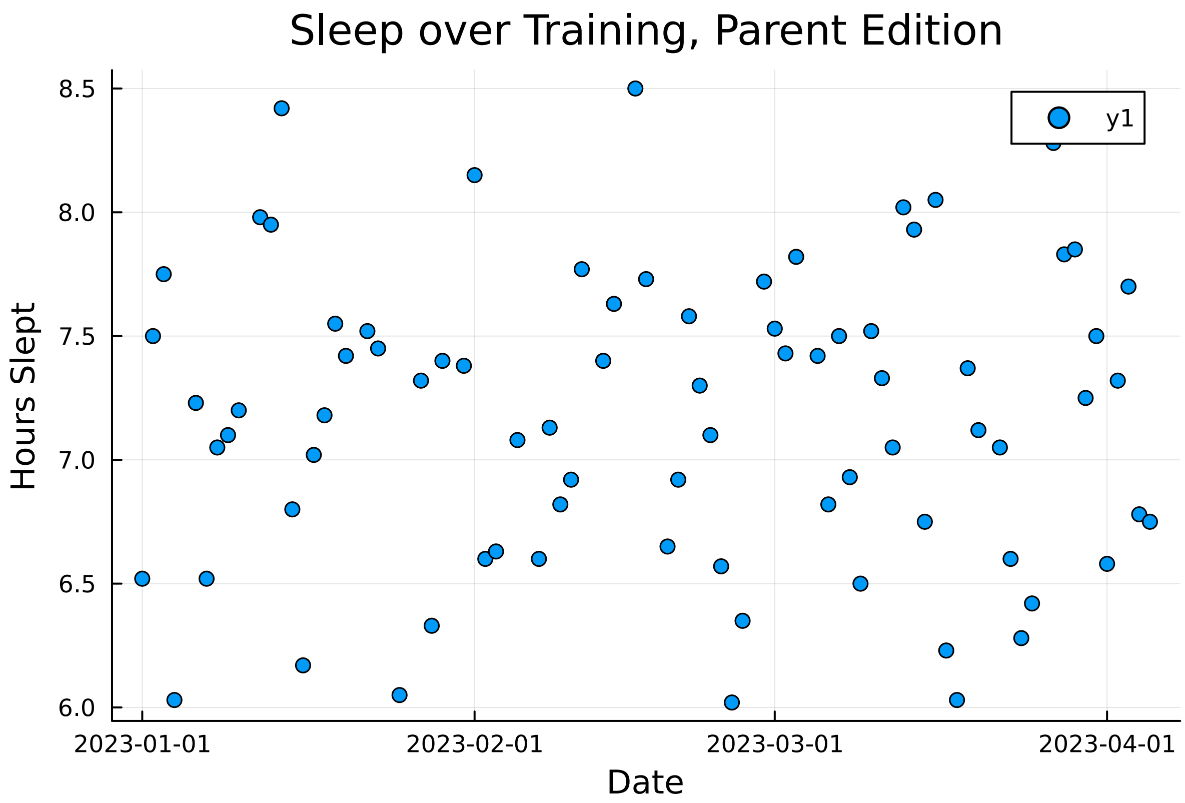
```julia
md"""
## First Steps

- Usually a good idea to describe the dataframe that you have
- Peform any cleaning needed

## New Columns

- Defined using the `.` format
- Uses broadcast over the existing Timestamp column that is a string
- Better for plotting and programmatic usage.
"""
```

Selection deleted

## Sleep During Training

- How well did I sleep during training?
- Was I consistent in my sleep?
- Hypothesis, probably not

```julia
md"""
## Sleep During Training

- How well did I sleep during training?
- Was I consistent in my sleep?
- Hypothesis, probably not
"""
```



```julia
begin
    # Sleep data, with outliers trimmed (didn't wear my watch)
    row_selector = (metrics.Type .== "Sleep Hours") .&& (metrics.Value .< 9.) .&&
    (metrics.Value .> 6)
```

Selection deleted

```julia
    sleep_hours = metrics[row_selector, [:Value, :ParsedTimestamp]]

    # Plot it
    scatter(sleep_hours.ParsedTimestamp, sleep_hours.Value, xlabel="Date",
    ylabel="Hours Slept", title="Sleep over Training, Parent Edition")
end
```

## Answer

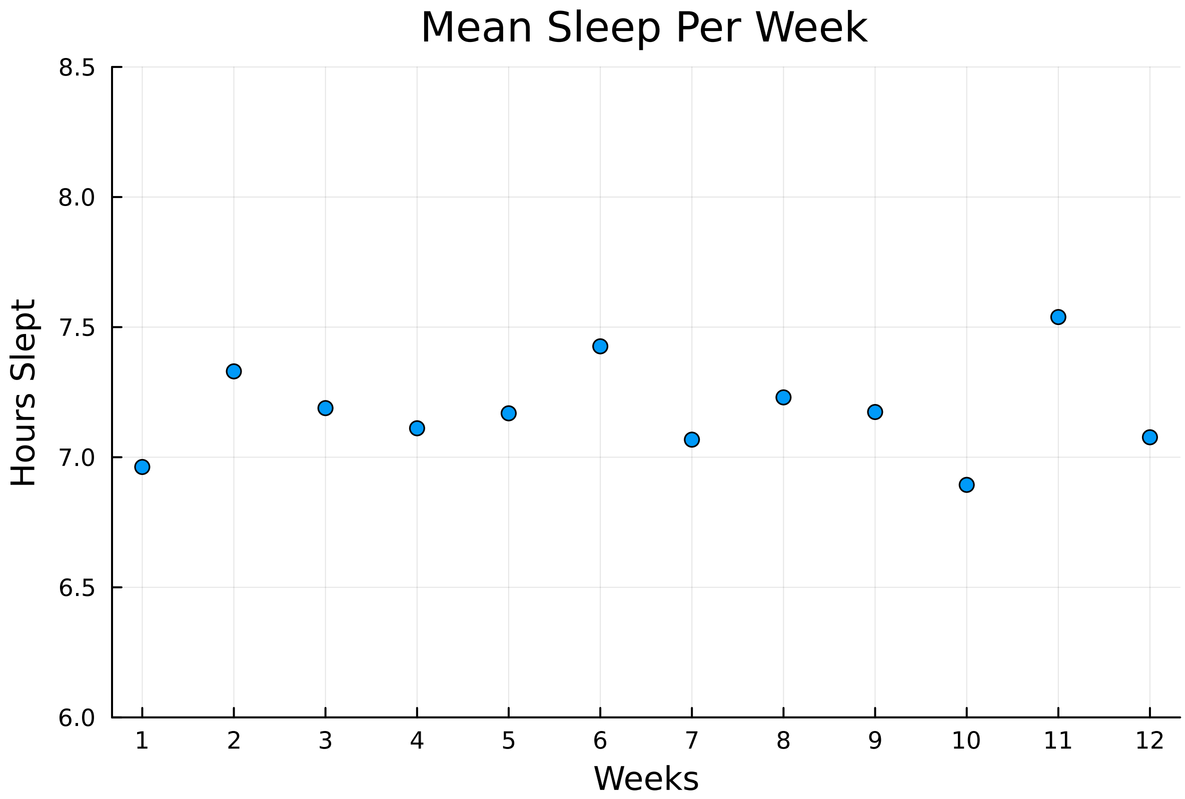
- No, I was not consistent in my sleep at all.
- Probably due to toddler
- Or potty training
- Or having aching legs...
- I digress.

### Follow up

Were my averages week to week consistent?

```
md"""
## Answer

- No, I was not consistent in my sleep at all.
- Probably due to toddler
- Or potty training
- Or having aching legs...
- I digress.

### Follow up

Were my averages week to week consistent?
"""
```

Selection deleted



```julia
begin
    weeks = length(sleep_hours.Value) ÷ 7
    week_sleeps = [sleep_hours.Value[i:min(i+7, length(sleep_hours.Value))] for i ∈ 1:7:length(sleep_hours.Value)]
    mean_week_sleeps = mean.(week_sleeps)

    scatter(1:length(mean_week_sleeps), mean_week_sleeps, ylims=(6.0, 8.5), xticks=1:1:length(mean_week_sleeps), ylabel="Hours Slept", xlabel="Weeks", title="Mean Sleep Per Week", legend=false)
end
```

Selection deleted

## Answer

- Yes, my averages week to week were consistent.
- Between 7-7.5 hours of sleep usually.

## Methodology

- Creates segments of max length 7 sleep observations
- Takes the mean of each segment
- Plot those means

## Next Up

- Reading my workout data

```
md"""
## Answer

- Yes, my averages week to week were consistent.
- Between 7-7.5 hours of sleep usually.

## Methodology

- Creates segments of max length 7 sleep observations
- Takes the mean of each segment
- Plot those means

## Next Up

- Reading my workout data
"""
```

Selection deleted

## Reading Workout Data

- Data is stored in .FIT files
- I failed to write the Garmin FIT SDK in time for Julia
- Use PyCall and Garmin's Python SDK!

```
md"""
## Reading Workout Data

- Data is stored in .FIT files
- I failed to write the Garmin FIT SDK in time for Julia
- Use PyCall and Garmin's Python SDK!
"""
```

```
begin
    # Install the Garmin SDK into our Notebook environment.
    Conda.pip_interop(true)
    Conda.pip("install", "/Users/jacobwindle/Downloads/FitSDKRelease_21.105.00/py")
end
```

```
Running `conda config --set pip_interop_enabled true --file /Users/jacobwindle/.j
ulia/conda/3/aarch64/condarc-julia.yml` in root environment

Running `pip install /Users/jacobwindle/Downloads/FitSDKRelease_21.105.00/py` in
root environment
```

```
Processing /Users/jacobwindle/Downloads/FitSDKRelease_21.105.00/py
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Building wheels for collected packages: garmin-fit-sdk
  Building wheel for garmin-fit-sdk (pyproject.toml): started
  Building wheel for garmin-fit-sdk (pyproject.toml): finished with status 'don
e'
  Created wheel for garmin-fit-sdk: filename=garmin_fit_sdk-21.105.0-py2.py3-no
ne-any.whl size=134176 sha256=10bc4195228a000d8e2458265e5583043a13df91dfe5e8e3d
4b0047a1ca529e6
  Stored in directory: /private/var/folders/mr/vq80v4cn4rgdtl1pbfrcfvvr0000gp/
T/pip-ephem-wheel-cache-06p5moqu/wheels/ce/6d/b8/b728a6064b9404f496915e7dec78a1
d6ce0927e9e2d73bbe8a
Successfully built garmin-fit-sdk
Installing collected packages: garmin-fit-sdk
  Attempting uninstall: garmin-fit-sdk
    Found existing installation: garmin-fit-sdk 21.105.0
    Uninstalling garmin-fit-sdk-21.105.0:
      Successfully uninstalled garmin-fit-sdk-21.105.0
Successfully installed garmin-fit-sdk-21.105.0
```

Selection deleted

# Python Interop

- Useful because I failed at writing the FIT SDK
- Use Pip interop in this case, `install` with local filepath
- Now time to read all the FIT files

```
md"""
## Python Interop

- Useful because I failed at writing the FIT SDK
- Use Pip interop in this case, `install` with local filepath
- Now time to read all the FIT files
"""
```

```
[Dict("79" => [Dict(   more)], "lap_mesgs" => [Dict(   more), Dict(   more), Dict(   more)
```

```
begin
    @pyimport garmin_fit_sdk

    fit_files = readdir("./fit_files"; join=true)

    function decode_fit_file(fp::AbstractString)::Tuple{Any,Any}
        stream = garmin_fit_sdk.Stream.from_file(fp)
        decoder = garmin_fit_sdk.Decoder(stream)
        try
            decoder.read()
        finally
            stream.close()
        end
    end

    decoded = []
    for file ∈ fit_files
        push!(decoded, decode_fit_file(file))
    end

    decoded_fit_files = [dc[1] for dc in decoded]
end
```

Selection deleted

## Reading the FIT files

- Function decode*fit*file will use the Garmin Python SDK to read fit file
- Use combo listcomp and broadcast to decode all files

## Getting my Heartrate Data

- From clicking through data, all heartrate information is in `record_mesgs`
- Convert `record_mesgs` into a dataframe for each date

```
md"""
## Reading the FIT files

- Function decode_fit_file will use the Garmin Python SDK to read fit file
- Use combo listcomp and broadcast to decode all files

## Getting my Heartrate Data

- From clicking through data, all heartrate information is in `record_mesgs`
- Convert `record_mesgs` into a dataframe for each date
"""
```

Selection deleted

[(2023-01-02T16:53:51,

| | altitude | distance | position_lat | heart_rate | enhanced_spee |
|---|---|---|---|---|---|
| 1 | 542.0 | 0.0 | 433472571 | 90 | 1.344 |
| 2 | 542.0 | 1.42 | 433472469 | 90 | 1.325 |
| 3 | 543.8 | 19.84 | 433471179 | 86 | 2.911 |
| 4 | 545.8 | 32.86 | 433470431 | 90 | 2.799 |
| 5 | 546.2 | 35.65 | 433470266 | 94 | 2.799 |
| 6 | 546.8 | 38.42 | 433470114 | 98 | 2.79 |
| 7 | 547.2 | 41.19 | 433469959 | 103 | 2.762 |
| 8 | 547.8 | 44.2 | 433469806 | 109 | 2.762 |
| 9 | 548.0 | 47.25 | 433469642 | 114 | 2.781 |
| 10 | 548.4 | 50.44 | 433469481 | 119 | 2.837 |
| more | | | | | |
| 418 | 505.4 | 7356.13 | 433260335 | 150 | 2.687 |

```julia
begin
    function string_keys(d::Dict{Any,Any})::Dict{String,Any}
        Dict([k => v for (k, v) in d if k isa AbstractString])
    end

    function extract_record_and_timestamp(fit_file::Dict{Any, Any})
        ts = DateTime(fit_file["activity_mesgs"][1]["timestamp"])
        messages = string_keys.(fit_file["record_mesgs"])

        try
            ts, DataFrame(messages)
        catch
            nothing
        end
    end

    decoded_files = extract_record_and_timestamp.(decoded_fit_files)
end

```

Selection deleted

## Mapping my data to DataFrame

- Extract the data needed
- Write function that pulls data out of PyDict

```
md"""

## Mapping my data to DataFrame

- Extract the data needed
- Write function that pulls data out of PyDict

"""
```

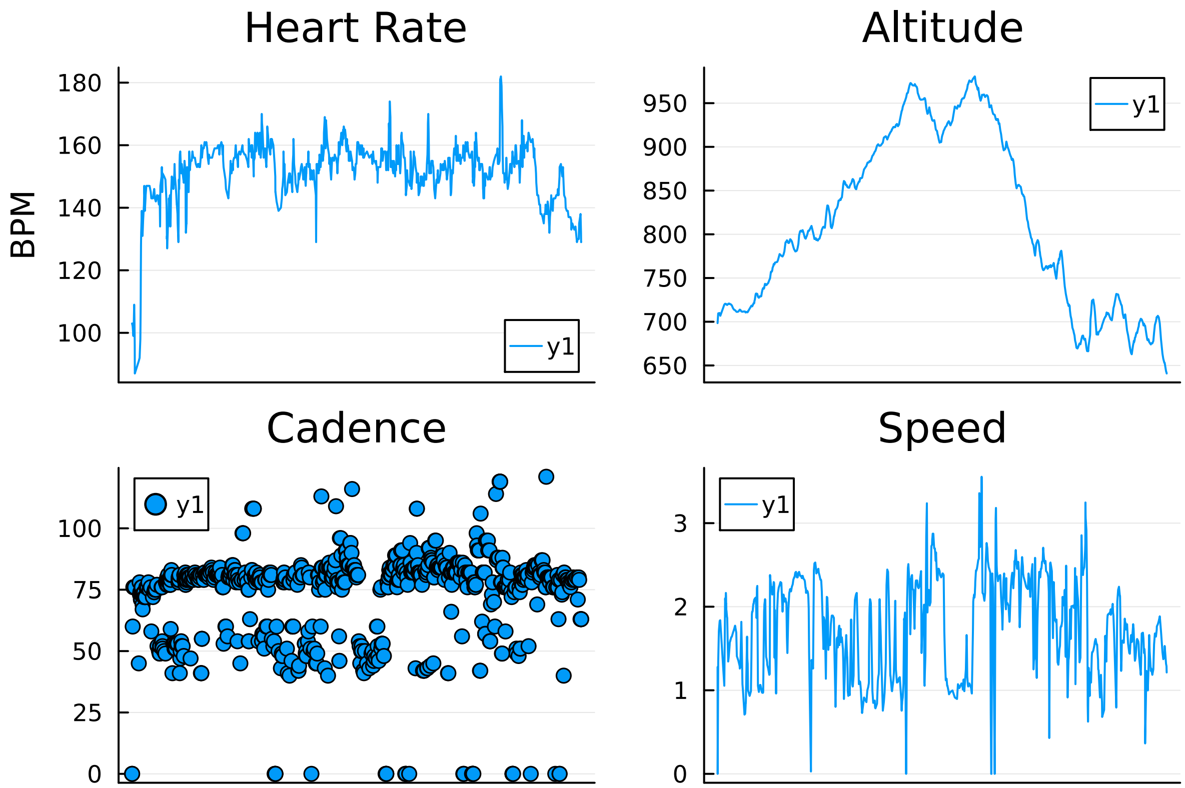
| | variable | mean | min | median | |
|---|---|---|---|---|---|
| 1 | :altitude | 506.938 | 471.8 | 500.6 | 558.4 |
| 2 | :distance | 3535.39 | 0.0 | 3474.36 | 7356. |
| 3 | :position_lat | 4.33296e8 | 433204250 | 4.3326e8 | 43347 |
| 4 | :heart_rate | 145.935 | 86 | 146.0 | 166 |
| 5 | :enhanced_speed | 2.74143 | 0.0 | 2.827 | 3.863 |
| 6 | :fractional_cadence | 0.239234 | 0.0 | 0.0 | 0.5 |
| 7 | :speed | 2.74143 | 0.0 | 2.827 | 3.863 |
| 8 | :position_long | -9.82415e8 | -982694531 | -9.82404e8 | -9821 |
| 9 | :cadence | 80.9234 | 0 | 82.0 | 89 |
| 10 | :enhanced_altitude | 506.938 | 471.8 | 500.6 | 558.4 |
| 11 | :timestamp | nothing | 2023-01-02T16:08:19 | 2023-01-02T16:28:42.500 | 2023- |

```
describe(decoded_files[1][2])
```

Selection deleted

```
graph_run (generic function with 1 method)
```

```julia
begin
    # A generic graph_run function
    function graph_run(decoded_file::Tuple{DateTime, DataFrame})
        label = "Run on $(decoded_file[1])"
        df = decoded_file[2]
        l = @layout [a b; c d]
        p = plot(df."timestamp", df."heart_rate"; title="Heart Rate", ylabel="BPM", xticks=nothing)
        p2 = plot(df.timestamp, df.altitude; title="Altitude", xticks=nothing)
        p3 = scatter(df.timestamp, df.cadence, title="Cadence", xticks=nothing)
        p4 = plot(df.timestamp, df.speed; title="Speed", xticks=nothing)
        plot(p, p2, p3, p4, layout = l)
    end
end
```

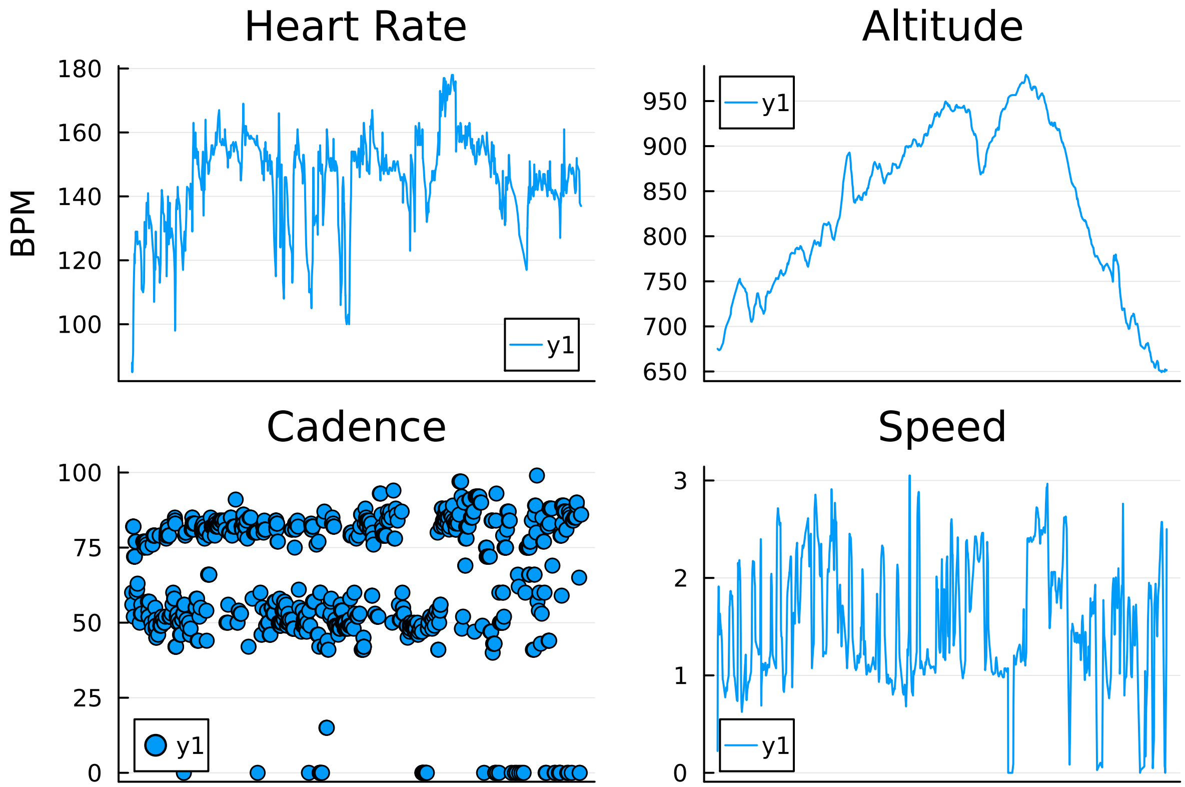


```julia
graph_run(decoded_files[14])
```

## Graphing Runs

- Show speed, cadence, heartrate, and altitude
- Trail runs appear to have varied cadence, road runs are more stable

```
md"""
## Graphing Runs

- Show speed, cadence, heartrate, and altitude
- Trail runs appear to have varied cadence, road runs are more stable
"""
```

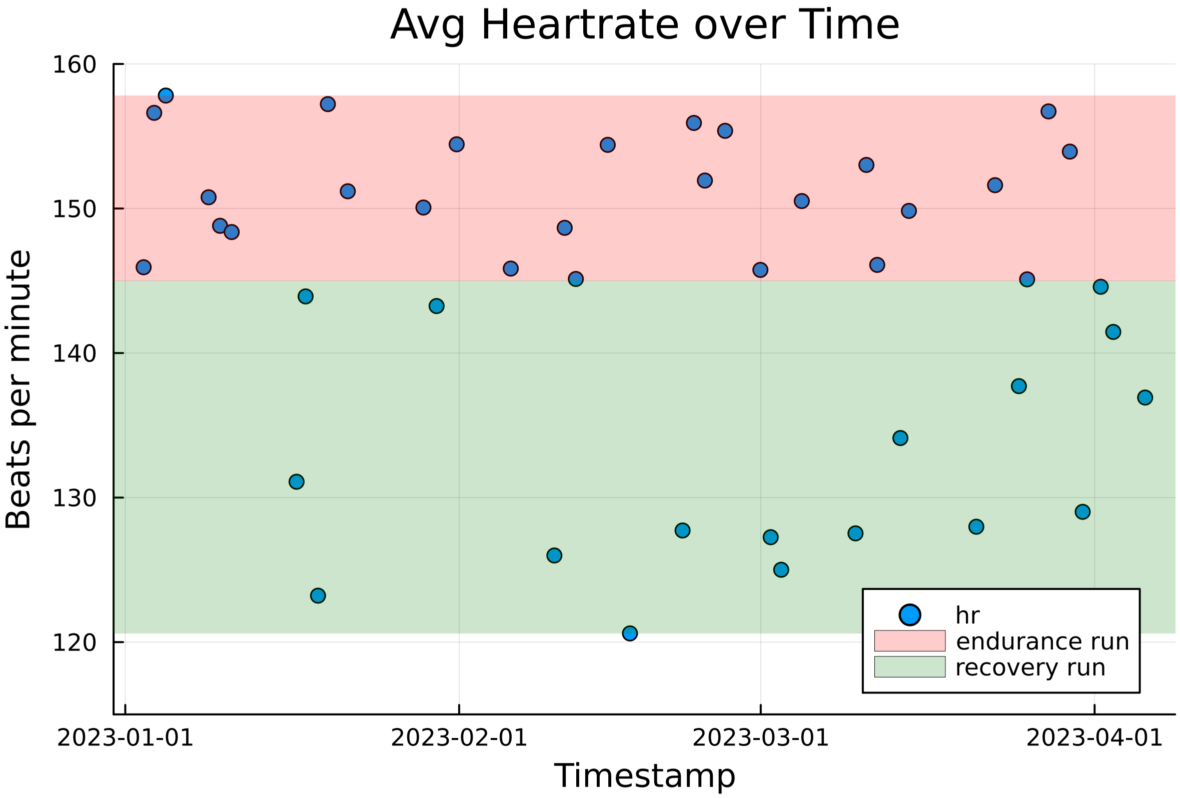


```
graph_run(decoded_files[19])
```

## Trends across Training

- What was my average heartrate for each run?
- Was my average heartrate effected by sleep?

```
md"""
## Trends across Training

- What was my average heartrate for each run?
- Was my average heartrate effected by sleep?
"""
```

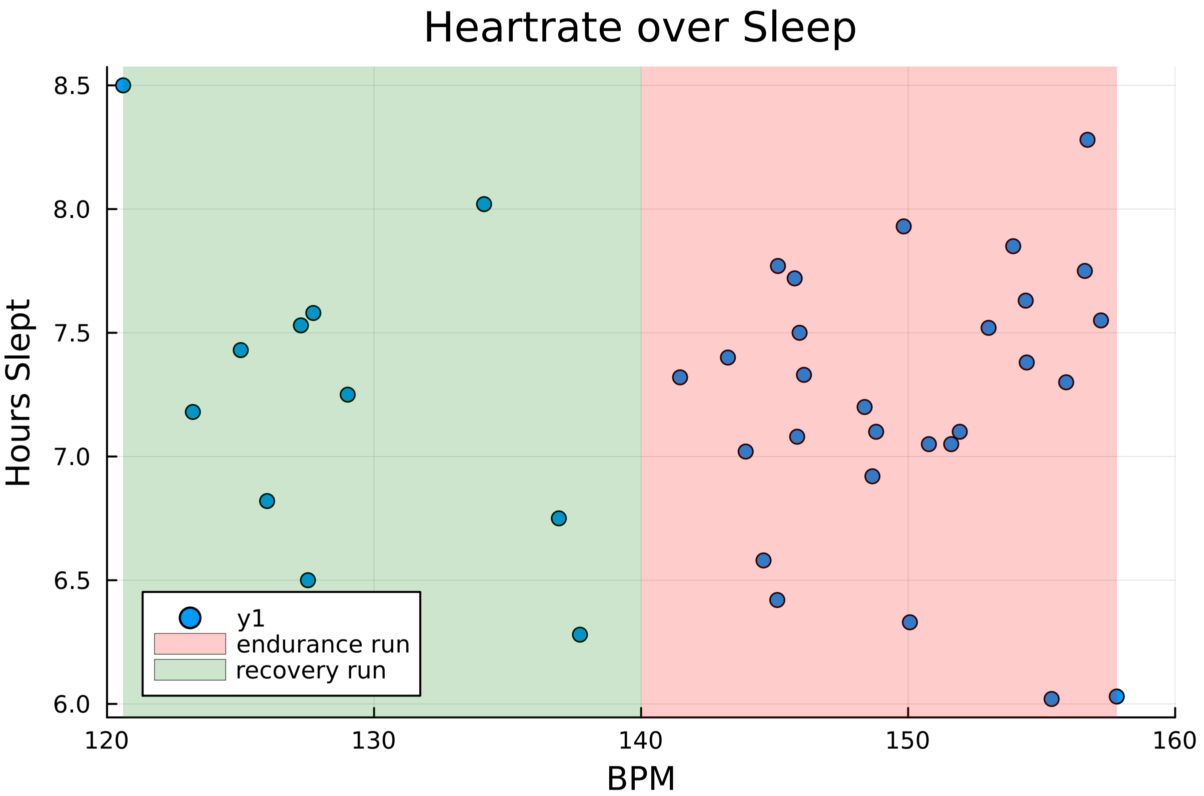


```julia
begin
    function calc_avgs(data::Union{Nothing, Tuple{DateTime, DataFrame}})
        if isnothing(data)
            return nothing
        end
        try
            return (timestamp=data[1], mean_hr=mean(data[2].heart_rate), mean_cadence=mean(data[2].cadence), mean_alt=mean(data[2].altitude))
        catch
            return nothing
        end
    end

    avg_heartrates = DataFrame(filter(f -> !isnothing(f), calc_avgs.(decoded_files)))

    ps = scatter(avg_heartrates.timestamp, avg_heartrates.mean_hr, title="Avg Heartrate over Time", label="hr", xlabel="Timestamp", ylabel="Beats per minute", ylims=(115, 160))
    hspan!(ps, [145, maximum(avg_heartrates.mean_hr)]; color=:red, alpha=0.2, label="endurance run")
    hspan!(ps, [minimum(avg_heartrates.mean_hr), 145]; color=:green, alpha=0.2, label="recovery run")
end
```

# Determining if Heartrate was affected by sleep

- Will need to use the workouts to determine which days were endurance runs
- Find sleep data for same day
- Stitch together to see results

```
md"""
## Determining if Heartrate was affected by sleep

- Will need to use the workouts to determine which days were endurance runs
- Find sleep data for same day
- Stitch together to see results
"""
```



```julia
begin
    # Copy the existing dataframe.
    hr_sleep_df = deepcopy(avg_heartrates)

    # Get the days that we have an endurance run
    my_endurance_days = workouts[:, :WorkoutDay]

    my_endurance_selector = [d in my_endurance_days for d in Date.(hr_sleep_df.timestamp)]

    my_endurance_days_df = hr_sleep_df[my_endurance_selector, :]
    my_endurance_days_df.Date = Date.(my_endurance_days_df.timestamp)

    sleep_on_endurance_days = [d in my_endurance_days for d in Date.(sleep_hours.ParsedTimestamp)]
    sleep_on_endurance_days_df = sleep_hours[sleep_on_endurance_days, :]
    sleep_on_endurance_days_df.Date = Date.(sleep_on_endurance_days_df.ParsedTimestamp)

    sleep_hr_df = leftjoin(my_endurance_days_df, sleep_on_endurance_days_df; on = :Date)
    ns = scatter(sleep_hr_df.mean_hr, sleep_hr_df.Value; xlims=(120, 160), title="Heartrate over Sleep", xlabel="BPM", ylabel="Hours Slept")
    vspan!(ns, [140, maximum(sleep_hr_df.mean_hr)]; color=:red, alpha=0.2, label="endurance run")
    vspan!(ns, [minimum(sleep_hr_df.mean_hr), 140]; color=:green, alpha=0.2, label="recovery run")
end
```

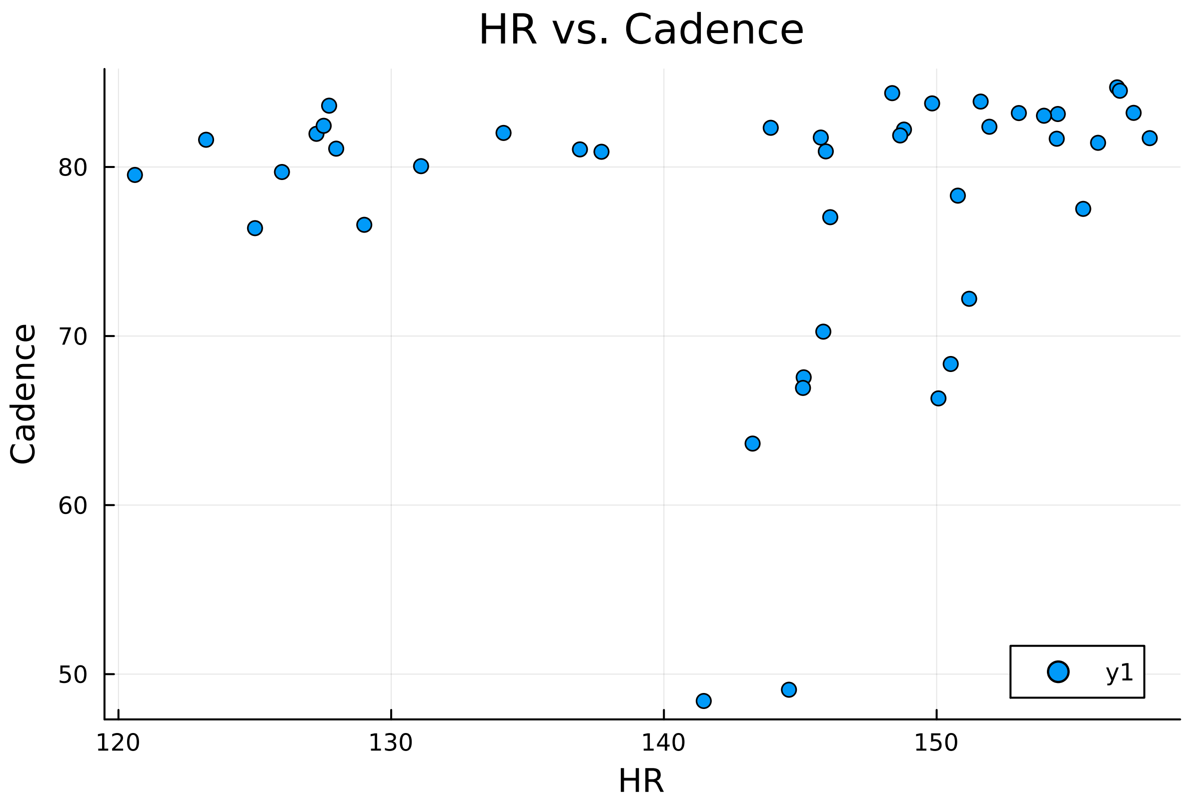
## Findings

- No discernible relationship
- Appears to have clusters, left half is recovery, right half is endurance

## Next Question

- Does Cadence Affect HR?

```
md"""
## Findings

- No discernible relationship
- Appears to have clusters, left half is recovery, right half is endurance

## Next Question

- Does Cadence Affect HR?
"""
```



```
begin
    cadence_hr_df = deepcopy(avg_heartrates)
    # cadence_hr_df.Date = Date.(cadence_hr_df.timestamp)
    scatter(cadence_hr_df.mean_hr, cadence_hr_df.mean_cadence; xlabel="HR", ylabel="Cadence", title="HR vs. Cadence")
end
```

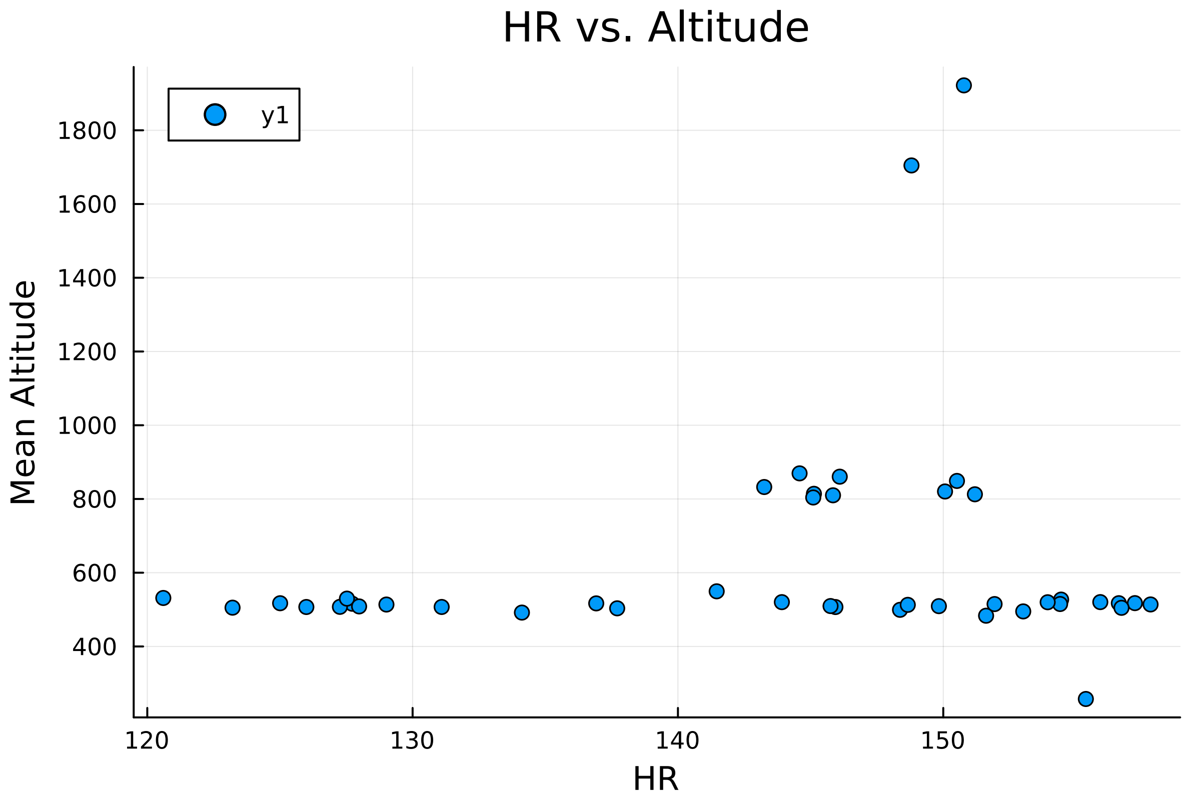
## Findings

- No relationship, because I always ran with the same cadence!

## Next Question

- Does altitude affect HR?

```
md"""
## Findings

- No relationship, because I always ran with the same cadence!

## Next Question

- Does altitude affect HR?
"""
```



```
begin
    altitude_hr_df = deepcopy(avg_heartrates)
    scatter(altitude_hr_df.mean_hr, altitude_hr_df.mean_alt; xlabel="HR", ylabel="Mean Altitude", title="HR vs. Altitude")
end
```

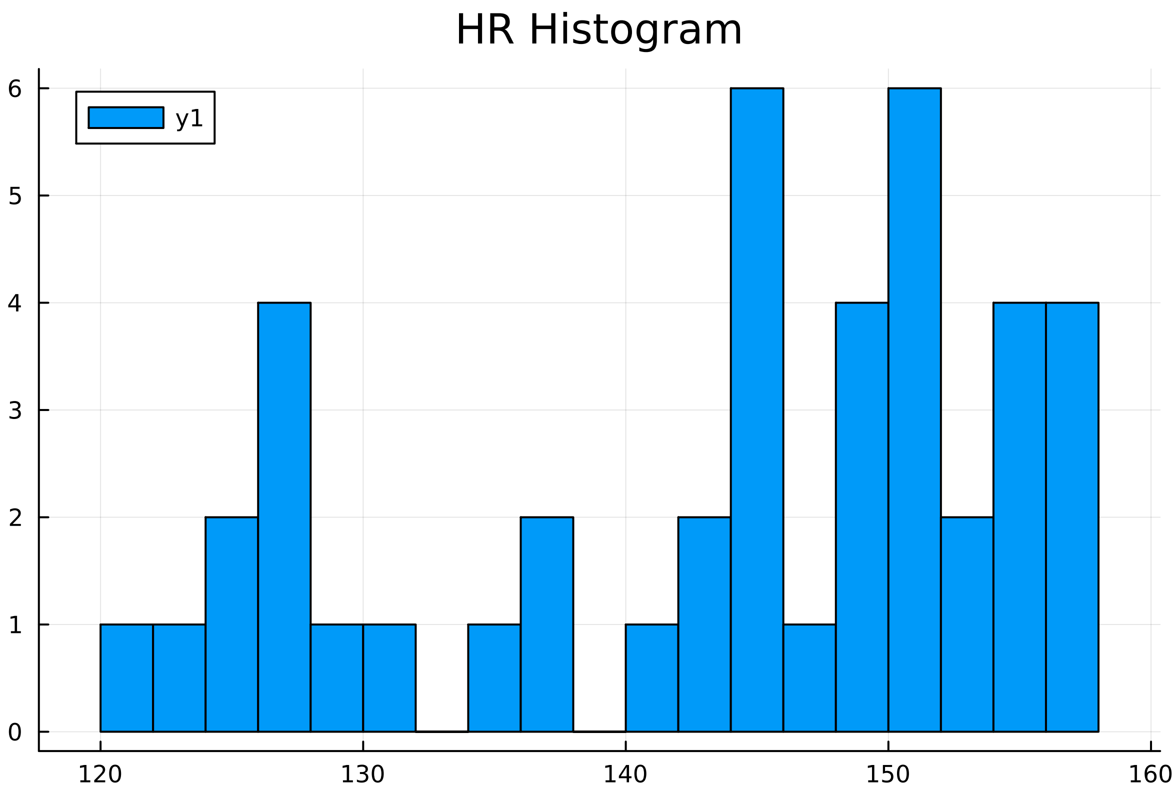
# Findings

- Altitude does appear to affect heart rate.
- Majority of runs were not at altitude, show normal distribution
- Runs at higher altitudes appear to cluster at higher HR

# Next Questions

- Do my data follow the normal distribution?

```
md"""
# Findings

- Altitude does appear to affect heart rate.
- Majority of runs were not at altitude, show normal distribution
- Runs at higher altitudes appear to cluster at higher HR

# Next Questions

- Do my data follow the normal distribution?
"""
```



```
begin
    d = fit(UnitRangeTransform, avg_heartrates.mean_hr)
    hr_normalized = StatsBase.transform(d, avg_heartrates.mean_hr)

    avg_heartrates.hr_normalized = hr_normalized
    @df avg_heartrates histogram(:mean_hr; bins=25, title="HR Histogram")
end
```

## Findings

- Kind of normally distributed, but have two clusters
- Recovery vs. Endurance again!

```
md"""
## Findings

- Kind of normally distributed, but have two clusters
- Recovery vs. Endurance again!
"""
```