MADA Project

Joe Martin

10/8/2021

Part 1 - Background

In marathon culture, the crowning achievement is qualifying for the Boston Marathon. Having some of the strictest qualifying time requirements, athletes take great care to choose qualifying races to achieve their desired time. Beyond this single event, time matters to all competitive athletes, who will work to ensure every variable for a race is perfect. When researching races, athletes will search for net down-hill courses, flat courses, or cold courses in an effort to maximize their performance.

In recent years, fitness tracking watches have become increasingly accurate and affordable, where just about every endurance athlete owns one. In fact, certain smart watch models can predict accurate values for tests like VO2 Max and lactate threshold. These heart rate-based tests are typically expensive, invasive, and reserved for professional athletes. Along with other advanced data, smart watches can provide a complete picture of a user's fitness and accurately predict race paces.

The goal of this project, therefore, is to model individual performance in an attempt to create a model that can predict average pace. The application for this model would be for choosing an eventual race. A user could theoretically use variables like distance, weather, and elevation from the race course, as well as performance variables like heart rate, stride length, and cadence to make a general prediction of how well they could perform in a race.

Methodology

Data

The data used for this model comes from my Garmin Forerunner 245 and run journal. The final data set contains 417 observations and more than 20 variables. Variables include date, distance, weather (temperature and conditions), total ascent, total descent, min elevation, max elevation, cadence, average stride length, and average pace. Much of this data is replicated in my running journal. While this isn't included in the model, it was used in my data exploration and was joined with my final model set.

```
glimpse(run_journal)
```

Running Journal

```
## Rows: 1,070
## Columns: 18
## $ weekday      <chr> "Sunday", "Thursday", "Saturday", "Monday", "Tuesday", "~
## $ date         <date> 2013-01-20, 2013-02-21, 2013-04-06, 2013-04-08, 2013-04~
```

```
## $ start_time    <chr> "10:06 AM", "3:34 PM", "10:33 AM", "4:40 PM", "4:37 PM",~
## $ run_type      <fct> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ distance_type <fct> Short Distance, Short Distance, Short Distance, Short Di~
## $ temperature   <dbl> 55, NA, 59, 75, 79, 70, NA, 72, NA, 57, 55, 70, 52, NA, ~
## $ weather       <fct> clear, NA, clear, cloudy, partly cloudy, cloudy, NA, sun~
## $ location      <chr> "Bonterra Village", "Bonterra Village", "Bonterra Villag~
## $ shoes         <fct> Nike Air Relentless, Nike Air Relentless, Nike Air Relen~
## $ distance      <dbl> 3.26, 1.32, 2.19, 2.51, NA, 2.58, 2.45, 2.55, 2.05, 2.59~
## $ total_time    <Period> 27M 21S, 10M 47S, 18M 48S, 21M 20S, 21M 36S, 22M 13S,~
## $ min           <dbl> 480, 480, 480, 480, 480, 480, 480, 480, 480, 480, 480, 4~
## $ sec           <dbl> 22, 12, 34, 30, 25, 36, 58, 42, 36, 11, 18, 59, 5, 2, 12~
## $ splits        <chr> "8:04,8:27,8:43", "7:38", "7:54,9:08", "8:06,8:53", "7:5~
## $ notes         <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ datetime      <dttm> 2013-01-20 10:06:00, 2013-02-21 15:34:00, 2013-04-06 10~
## $ avg_pace_sec  <dbl> 502, 492, 514, 510, 505, 516, 538, 522, 516, 491, 498, 4~
## $ week          <dbl> 3, 8, 14, 15, 15, 15, 15, 16, 16, 16, 17, 17, 18, 19, 19~
```

```
head(run_journal)
```

```
## # A tibble: 6 x 18
##   weekday  date       start_time run_type distance_type  temperature weather
##   <chr>    <date>     <chr>      <fct>    <fct>                <dbl> <fct>
## 1 Sunday   2013-01-20 10:06 AM   <NA>     Short Distance          55 clear
## 2 Thursday 2013-02-21 3:34 PM    <NA>     Short Distance          NA <NA>
## 3 Saturday 2013-04-06 10:33 AM   <NA>     Short Distance          59 clear
## 4 Monday   2013-04-08 4:40 PM    <NA>     Short Distance          75 cloudy
## 5 Tuesday  2013-04-09 4:37 PM    <NA>     Short Distance          79 partly clo~
## 6 Thursday 2013-04-11 10:11 AM   <NA>     Short Distance          70 cloudy
## # ... with 11 more variables: location <chr>, shoes <fct>, distance <dbl>,
## #   total_time <Period>, min <dbl>, sec <dbl>, splits <chr>, notes <chr>,
## #   datetime <dttm>, avg_pace_sec <dbl>, week <dbl>
```

```
glimpse(garmin)
```

Garmin Data

```
## Rows: 428
## Columns: 25
## $ id              <chr> "2020-04-16-4", "2020-04-28-1.2", "2020-04-28-1.19", ~
## $ distance        <dbl> 4.00, 1.20, 1.19, 3.11, 5.01, 3.70, 1.42, 8.01, 3.92,~
## $ calories        <dbl> 445, 136, 134, 350, 641, 426, 146, 962, 485, 593, 396~
## $ avg_hr          <dbl> 162, 169, 166, 178, 182, 163, 160, 169, 173, 171, 165~
## $ max_hr          <dbl> 175, 186, 181, 192, 194, 181, 171, 181, 191, 186, 183~
## $ avg_run_cadence <dbl> 156, 152, 152, 154, 151, 151, 153, 148, 145, 153, 155~
## $ max_run_cadence <dbl> 174, 224, 167, 167, 172, 181, 201, 158, 178, 170, 184~
## $ total_ascent    <dbl> 177, 134, 139, 114, 347, 167, 49, 506, 88, 313, 89, 1~
## $ total_decent    <dbl> 147, 73, 75, 21, 255, 99, 45, 463, 90, 290, 102, 75, ~
## $ avg_stride      <dbl> 1.38, 1.52, 1.48, 1.58, 1.47, 1.43, 1.52, 1.42, 1.42,~
## $ min_elevation   <dbl> 634, 633, 631, 633, 625, 633, 640, 630, 719, 634, 719~
## $ max_elevation   <dbl> 741, 735, 736, 741, 741, 741, 663, 741, 737, 741, 735~
```

```
## $ avg_pace_sec     <dbl> 447, 419, 429, 395, 435, 449, 414, 459, 456, 440, 424~
## $ best_pace_sec    <dbl> 388, 100, 370, 356, 374, 340, 237, 411, 302, 365, 292~
## $ week             <dbl> 16, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19, 19, 2~
## $ `sweat_loss(ml)` <dbl> 178, 50, 51, 122, 217, 166, 58, 367, 179, 220, 155, 1~
## $ aerobic_TE       <dbl> 3.7, 2.6, 2.5, 3.9, 5.0, 3.6, 2.3, 4.6, 3.9, 4.0, 3.3~
## $ aerobic_fct      <fct> Impacting, Maintaining, Maintaining, Impacting, Overr~
## $ anaerobic_value  <dbl> 0.3, 0.5, 0.2, 2.3, 3.1, 0.5, 0.6, 1.1, 3.4, 0.9, 2.5~
## $ anaerobic_fct    <fct> No Benefit, No Benefit, No Benefit, Maintaining, Impa~
## $ avg_spd          <dbl> 8.1, 8.6, 8.4, 9.1, 8.3, 8.0, 8.7, 7.9, 7.9, 8.2, 8.5~
## $ max_spd          <dbl> 9.3, 36.1, 9.7, 10.1, 9.6, 10.6, 15.2, 8.8, 11.9, 9.9~
## $ short_distance   <fct> Y, Y, Y, Y, Y, Y, Y, N, Y, Y, Y, Y, N, Y, Y, Y, Y, Y,~
## $ middle_distance  <fct> N, N, N, N, N, N, N, Y, N, N, N, N, Y, N, N, N, N, N,~
## $ long_distance    <fct> N, N, N, N, N, N, N, N, N, N, N, N, N, N, N, N, N, N,~
```

```
head(garmin)
```

```
## # A tibble: 6 x 25
##   id          distance calories avg_hr max_hr avg_run_cadence max_run_cadence
##   <chr>          <dbl>    <dbl>  <dbl>  <dbl>           <dbl>           <dbl>
## 1 2020-04-16-4       4      445    162    175             156             174
## 2 2020-04-28-1.2   1.2      136    169    186             152             224
## 3 2020-04-28-1.19 1.19      134    166    181             152             167
## 4 2020-04-29-3.11 3.11      350    178    192             154             167
## 5 2020-04-30-5.01 5.01      641    182    194             151             172
## 6 2020-05-01-3.7   3.7      426    163    181             151             181
## # ... with 18 more variables: total_ascent <dbl>, total_decent <dbl>,
## #   avg_stride <dbl>, min_elevation <dbl>, max_elevation <dbl>,
## #   avg_pace_sec <dbl>, best_pace_sec <dbl>, week <dbl>, sweat_loss(ml) <dbl>,
## #   aerobic_TE <dbl>, aerobic_fct <fct>, anaerobic_value <dbl>,
## #   anaerobic_fct <fct>, avg_spd <dbl>, max_spd <dbl>, short_distance <fct>,
## #   middle_distance <fct>, long_distance <fct>
```

Data Preparation

Early in my project, I started using data from the years 2013-2019. Initially, I thought it would be possible to model performance over time. During this period, however, I did not record my runs on a smart watch, so many of the necessary variables do not exist in this dataset. During the data preparation phase, I cleaned datasets which included data from the period 2013-2019. This data is included in the data preparation files and used in the exploratory analysis, but is not used for modeling.

The data sets available had more data and fields than necessary. The initial steps involved cleaning and tidying data, then joining the primary data sets together. Much of this work was focused on eliminating variables that were not related to running (swimming and cycling data), then ensuring that data from my run journal shared a unique identifier with my Garmin data to join on.

Modeling

Raw Data

Data included in this section contains my run logs from 2013 - 2021, as well as my resting heart rate from April 15, 2020 - present. Data I have available for use, but not yet ready to share includes my detailed post-run notes, sleep data, and other random calculations my Garmin watch does for me (VO2 Max, Pulse

OX, stress, etc.). I'm leaving these out initially because, while I'm interested in using them, I am also concerned they will greatly broaden the scope and complexity of this project.

```
summary(df1318)
```

Historical Data 2013 - 2018

```
##      day                date              weight          start_time
##  Length:302         Length:302         Mode:logical    Length:302
##  Class :character   Class :character   NA's:302        Class :character
##  Mode  :character   Mode  :character                   Mode  :character
##  run_type         distance_type        weather            location
##  Mode:logical     Length:302         Length:302       Length:302
##  NA's:302         Class :character   Class :character  Class :character
##                   Mode  :character   Mode  :character  Mode  :character
##    run_shoes          distance           total_time        average_pace
##  Length:302         Length:302         Length:302        Length:302
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##     splits             notes
##  Length:302         Mode:logical
##  Class :character   NA's:302
##  Mode  :character
```

```
head(df1318)
```

```
## # A tibble: 6 x 14
##   day      date      weight start_time run_type distance_type   weather   location
##   <chr>    <chr>     <lgl>  <chr>      <lgl>    <chr>           <chr>     <chr>
## 1 Sunday   1/20/2013 NA     10:06 AM   NA       Short Distance  55,clear  Bonterr~
## 2 Thursday 2/21/2013 NA     3:34 PM    NA       Short Distance  <NA>      Bonterr~
## 3 Saturday 4/6/2013  NA     10:33 AM   NA       Short Distance  59,clear  Bonterr~
## 4 Monday   4/8/2013  NA     4:40 PM    NA       Short Distance  75,clou~  Bonterr~
## 5 Tuesday  4/9/2013  NA     4:37 PM    NA       Short Distance  79,part~  Bonterr~
## 6 Thursday 4/11/2013 NA     10:11 AM   NA       Short Distance  70,clou~  Bonterr~
## # ... with 6 more variables: run_shoes <chr>, distance <chr>, total_time <chr>,
## #   average_pace <chr>, splits <chr>, notes <lgl>
```

Data from 2021 Data from 2019 - 2021 is located in ~/data/raw_data. These sample dataframes are a bit too messy to be useful in this section of the project - primarily because they contain header that are irrelevant. These primarily follow the same format as data from 2013 - 2018.

```
garminRun <- here::here("data","raw_data","garmin20211024.csv")
garmin <- read_csv(garminRun)
```

Data from Garmin Forerunner 245

```
## Rows: 490 Columns: 45
```

```
## -- Column specification -----------------------------------------------------
## Delimiter: ","
## chr  (29): Activity Type, Title, Calories, Avg HR, Max HR, Aerobic TE, Avg R...
## dbl   (9): Avg Stride Length, Avg Vertical Ratio, Avg Vertical Oscillation, ...
## lgl   (1): Favorite
## dttm  (1): Date
## time  (4): Time, Best Lap Time, Moving Time, Elapsed Time
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(garminRun)
```

```
## [1] "C:/Users/jmm03185/Documents/data/GitHub/joemartin-MADA-project/data/raw_data/garmin20211024.csv
```

```
rhr_data <- here::here("data","raw_data", "dailyRHR.csv")
```

```
rhr <- read_csv(rhr_data)
```

Resting Heart Rate 4/15/21 - Present

```
## Rows: 558 Columns: 3
```

```
## -- Column specification -----------------------------------------------------
## Delimiter: ","
## chr (2): date, weekday
## dbl (1): rhr(bpm)
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
summary(rhr)
```

```
##      date              weekday            rhr(bpm)
## Length:558         Length:558         Min.   :41.00
## Class :character   Class :character   1st Qu.:45.00
## Mode  :character   Mode  :character   Median :47.00
##                                       Mean   :47.21
##                                       3rd Qu.:48.00
##                                       Max.   :74.00
##                                       NA's   :3
```

```
head(rhr)
```

```
## # A tibble: 6 x 3
##   date      weekday 'rhr(bpm)'
##   <chr>     <chr>        <dbl>
## 1 4/15/2020 Wed             48
## 2 4/16/2020 Thu             48
## 3 4/17/2020 Fri             49
## 4 4/18/2020 Sat             43
## 5 4/19/2020 Sun             41
## 6 4/20/2020 Mon             42
```

Project Objectives

Through this project, I'm interested in understanding how factors of my training program affect my performance. I would like to see if there are any significant relationships between the frequency or intensity of my training and performance. My outcomes of interest would be finding whether there are significant patterns in my pace, active heart rate, or resting heart rate. I'm also interested in seeing if I can measure qualitative outcomes through the notes I take after each run. Some of the patterns I would look for in my data would include the relationship between a positive performance outcome and variables like shoes, number of speed workouts before a race or a long run, or weekly volume (total weekly mileage or time exercising). In order to search for relationships and patterns in my data, I will first need to identify specific events - like races or long runs - and compare weeks of training against each other. For example, I do one long distance run just about every week. I would be interested in understanding if my pace decreases on a long run if I had more active training days leading up to this run. I would want to see if my heart rate during similar runs (for example, long runs on the same route) decreases over time or increases after a break in training.

The other way I may go about analyzing patterns in my data would be by picking specific distances described the same way (easy, steady, tempo). For example, a common workout for me would be an up-tempo 10k (6.22 miles) or an easy 8-miler.

Finally, although I have few races in my data set, I do have a good understanding of my training program, which I follow for weeks in advance of a race. For some races, I have a detailed log of this program. I could measure my fitness ahead of a training program and compare it to my fitness just before or at race time. Measurement of fitness would include average pace, heart rate, and perhaps even ratings based on positive or negative notes taken about training sessions.

Further Background

Coaches and personal trainers are expensive to hire. Apps provide users with some basic stats about their runs, but users typically have to purchase a subscription to get useful insights into training. Further, apps often use algorithms that do not produce practical insights. For example, Garmin provides users with a Performance Condition measurement. Garmin devices display this measurement to users during a workout to tell them how their performance compares to their average fitness level. Using pace, heart rate, and heart rate variability data, Garmin displays a score from -20 (Poor) to values greater than 10 (Excellent).

This is just one example of a measurement which ignores natural variability in training sessions and directly contradicts training programs established over decades by expert coaches and researchers, like Pete Pfitzinger and Jack Daniels, PhD. These expert coaches outline training plans which have a general pattern for long distance races, from 5,000 m races, all the way to marathons. This pattern includes two general aerobic runs per week (about 1:00 per mile slower than race pace), one or two speed sessions (lactate threshold runs at race pace, tempo runs, or track intervals), one or two recovery runs (about 2:00 per mile slower than race

pace or heart rate in or below Zone 3), and a long distance run. These programs build in rest days, as well, which athletes are supposed to use for recovery or cross training.

Aggregate fitness data used by companies like Garmin and Polar is useful developing insights into human performance outcomes for the average person, like in "Human running performance from real-world big data" by Thorsten Emig & Jussi Peltonen. Due to the variability in quality distance training programs, however, aggregate measures and seemingly random algorithms are not useful in helping individual athletes understand whether they're training well.

```
pacman::p_load(pacman, tidyverse, here)

### Load Data ###
run_df <- read_rds(here::here("data","processed_data","run_data_clean.rds"))
garminRun <- read_rds(here::here("data","processed_data","garmin_data.rds"))
rhr <- read_rds(here::here("data","processed_data","resting_heart_rate.rds"))
```

Part 2 - Data Cleaning and Exploration

Instructions for reproducing my data cleaning and exploration process are in found in the readme.md file in the data folder. To reproduce code from Part 2 of this project, execute the script "mada_project_part2.R" located in "~code/processing_code". After executing this script, the figures below can be reproduced by executing the script "exploratory_analysis.R" in "~/code/analysis_code".

Data Cleaning

This stage of my project began with loading my raw data directly from the Google sheets I use with the googlesheets4 package. This is mentioned in the notes of my script, but no longer visible. The googlesheets4 package gives the ability to write to a googlesheet, so I thought it could be risky if I potentially gave that ability to anyone accessing my GitHub. Files for years 2013-2021 were loaded and saved as .rds files before processing. I left a line of code commented out so I'm able to paste in my Google sheet for year 2021 and update data frequently. At this stage, I also imported data from my Garmin Connect account for all of my runs since April 2020, as well as my resting heart rate data.

Over the years, some variables mattered to me and others didn't, so I began cleaning by removing variables that no longer seem to serve a purpose or have a small amount of data during one period of time.

Once I cleaned out the variables I didn't want, my data files had consistent column structure from 2013-2021. I coerced the data type of variables just enough so I could bind all of this data together.

My plan moving into data analysis is to keep a separate dataframe for my run journal data (2013-present), my Garmin data, and my resting heart rate. When I want to work with variables across each set, I will join them together.

To complete my data cleaning, I coerced each variable into its necessary type. The most challenging variable has proven to be average pace. This is likely one of the most important variables in my set (if not the most important), so I may need to rethink how I'm using this variable going forward. I ultimately ended by coercing it, as well as best_pace in the Garmin data set, using as.POSIXct. This is not ideal because as.POSIXct also assigns a date value. However, this was good enough to begin analysis. Following processing, clean datasets were saved as rds files in "~/data/processed_data". Notably, I saved two versions of my journal data - one set eliminates NAs for days I didn't run (1,202 observations), one of them retains them(more than 3,000 observations). The purpose of this was to retain notes I kept on days I did not run.

Exploratory Analysis

My exploratory analysis began primarily using the lm() function to see if any variables are highly correlated. I graphed these, as well, in order to better visualize this relationship. Unfortunately, lm() does not support date data and, therefore, could not give me useful information about correlations.

I purposely chose variables I anticipated would have significant p-values. I wanted to understand which variables would be most important to my future analysis and modeling, especially if I can introduce new variables, like miles per week and shoes.

The plots below were generated with the "exploratory_analysis.R" script.

Relationships

Resting Heart Rate   This plot show that my heart rate consistently remains around 47 bpm. There are significant outliers in this data that may skew the results higher. If I don't wear my Garmin watch all day and all night, my resting heart rate can be recorded as 60-70 bpm, which is very unlikely.

In future analyses, it will be interesting to compare this data to variables like weekly training volume, cadence, and average pace.

```r
rhr_plot <- rhr %>% ggplot(aes(x=date, y=`rhr(bpm)`))+
  geom_line()+
  geom_smooth(method = "lm")

rhr_plot
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 3 rows containing non-finite values (stat_smooth).
```
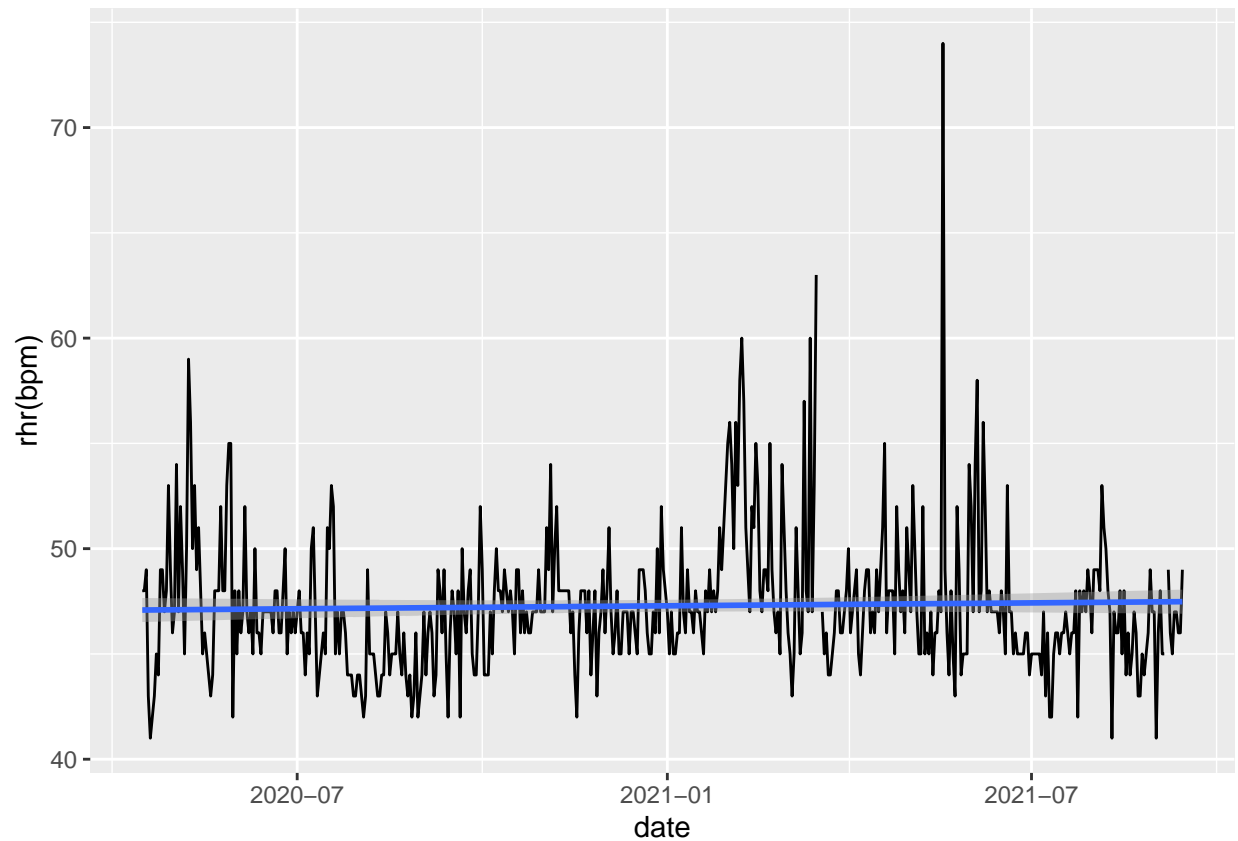
Average Pace Average pace is crucial to many of my analyses. Going forward, I will work on finding a better way to coerce this data into a time format that's usable in statistical models.

Over the past 8 years, my overall average pace has decreased, depsite my increas in true recovery runs this year. A recovery run is a very easy run where my heart rate stays in or below Zone 3.

```
average_pace_1321 <- run_df %>% ggplot(aes(x=date, y= avg_pace_sec))+
  geom_point()+
  geom_smooth(method = "lm")
average_pace_1321
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 142 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 142 rows containing missing values (geom_point).
```
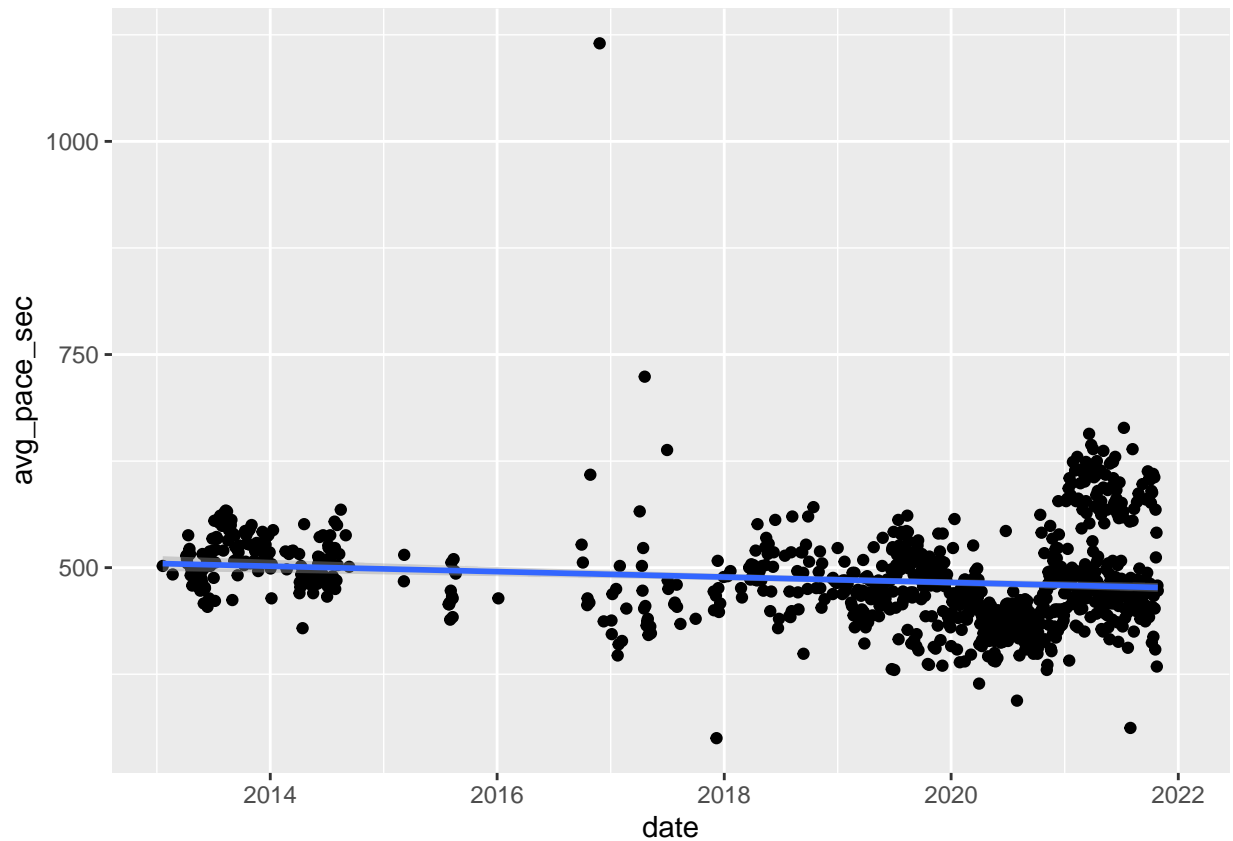
Over the past 3 years, however, my recovery runs have made my average pace slower.
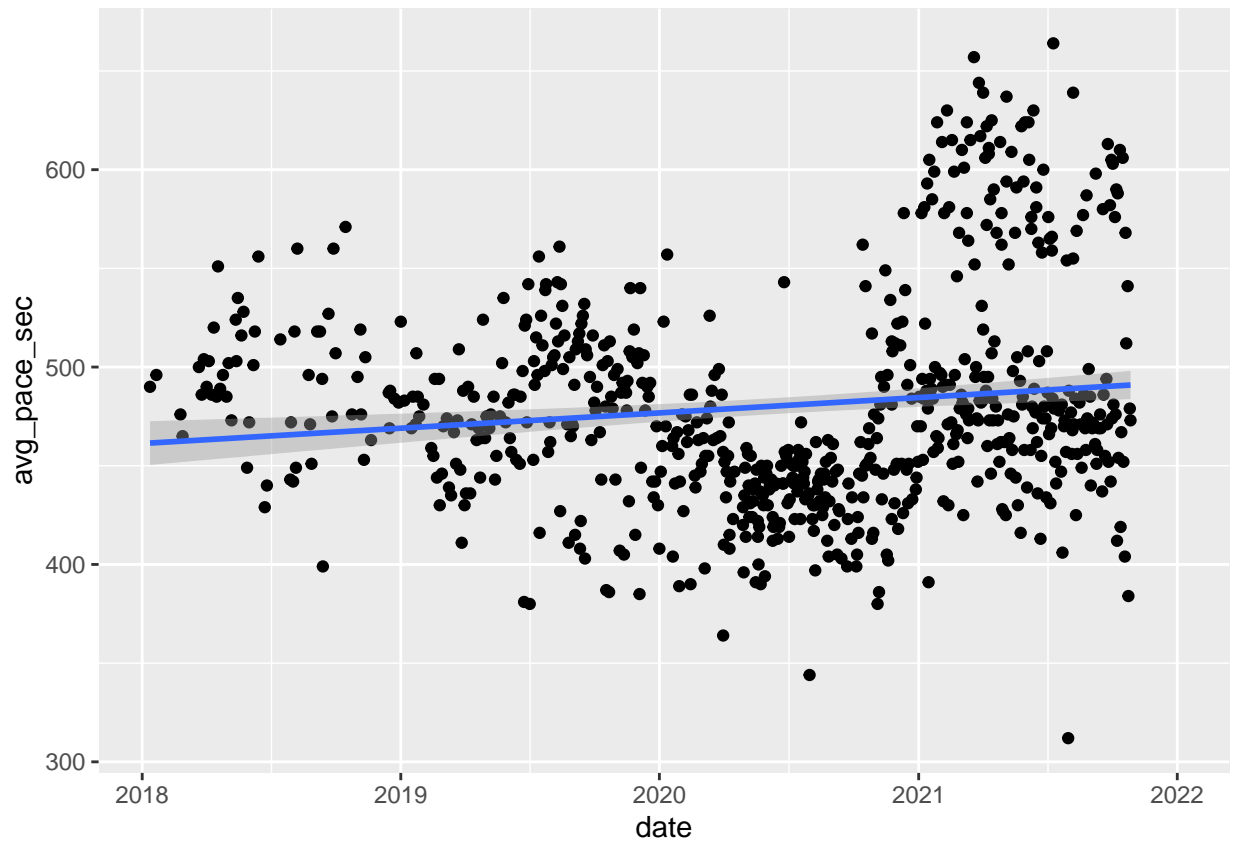
```
average_pace_1821 <- run_df %>% filter(date >= "2018-01-01") %>%
  ggplot(aes(x=date,y=avg_pace_sec))+
  geom_point()+
  geom_smooth(method = "lm")

average_pace_1821
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 140 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 140 rows containing missing values (geom_point).
```

For a better measure of my performance, I'll look at my longer distance runs - anything over 12 miles. The pace for these runs has decreased over time.

```
# Long Disance Runs
ld <- run_df %>%
  filter(distance >= 12) %>%
  filter(date >= "2018-01-01") %>%
  ggplot(aes(x=date,y=avg_pace_sec))+
  geom_point()+
  geom_smooth(method = "lm")

ld
```
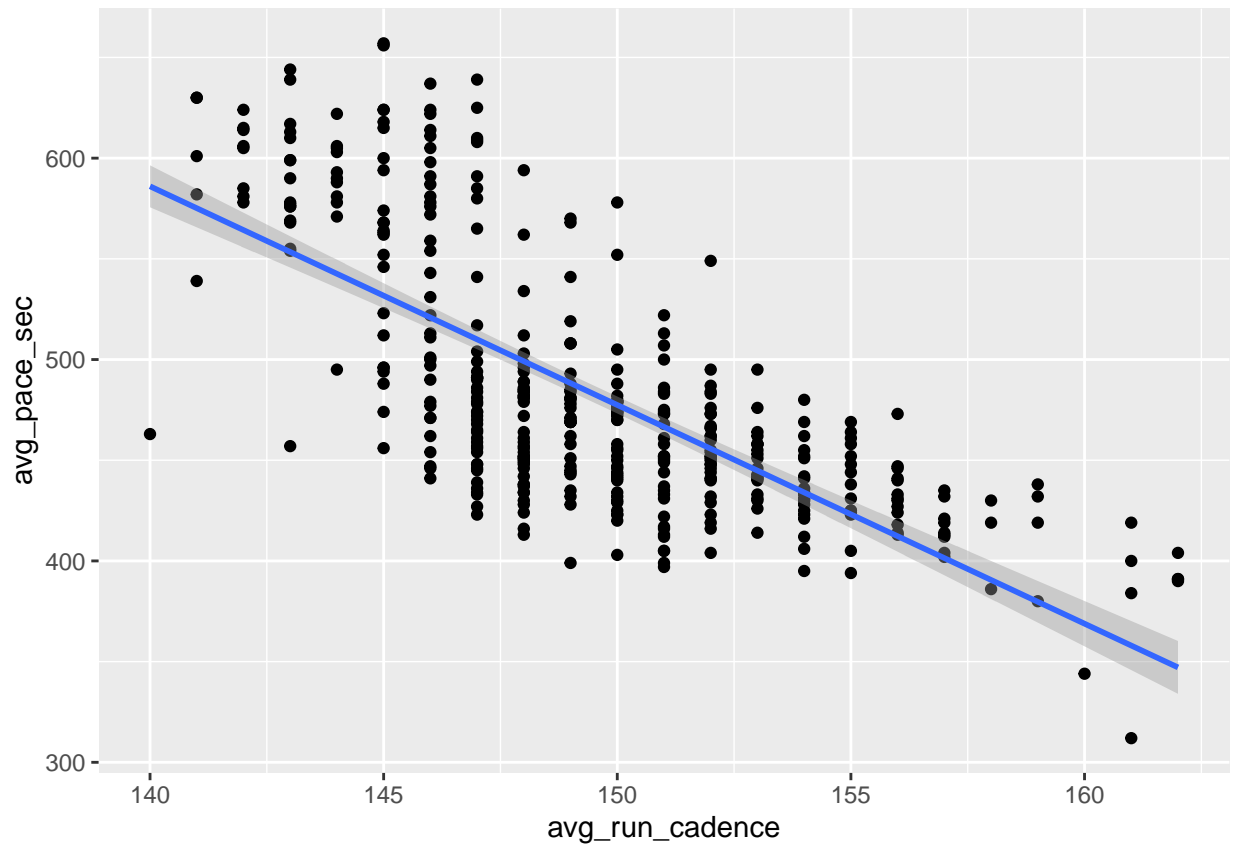
```
## `geom_smooth()` using formula 'y ~ x'
```
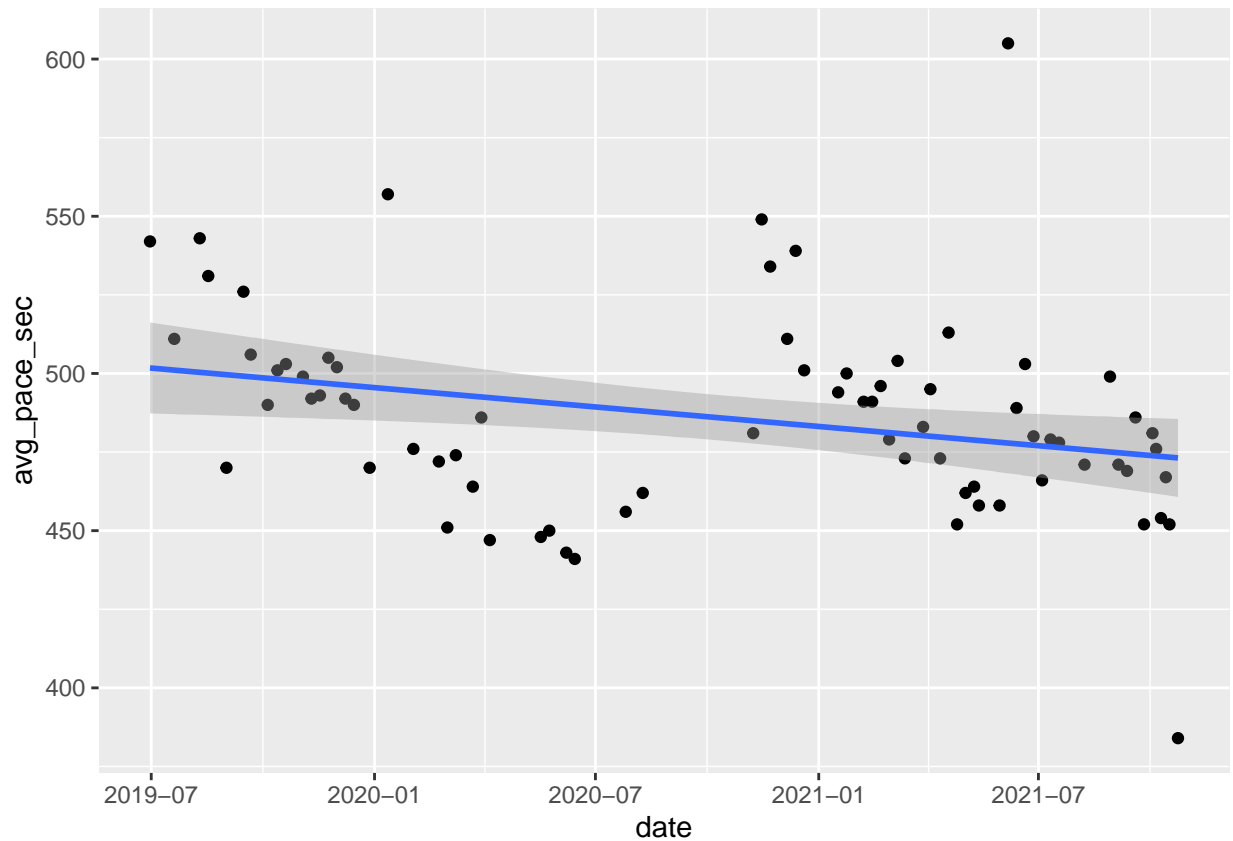
Cadence is the measure of how many times a runner's feet hit the ground per minute. A high cadence is typically associated with a faster pace.

```
# Effect of Cadence on Average Pace
cad_ap <-garminRun %>% ggplot(aes(x=avg_run_cadence, y=avg_pace_sec))+
  geom_point()+
  geom_smooth(method = "lm")

cad_ap
```

```
## 'geom_smooth()' using formula 'y ~ x'
```

Aerobic TE is a measurement Garmin created to rate an aerobic effort. On a scale from 1-5, 5 is a maximum effort. This is good stress for the body, but requires a lot of rest time afterwards. 1 is minimal and typically associated with other types of exercise besides running. This

```
# Effect of Harder Aerobic Effort on Average Pace
ae_ap<- garminRun %>% ggplot(aes(x=aerobic_TE, y=avg_pace_sec))+
  geom_point()+
  geom_smooth(method = "lm")

ae_ap
```

```
## 'geom_smooth()' using formula 'y ~ x'
```
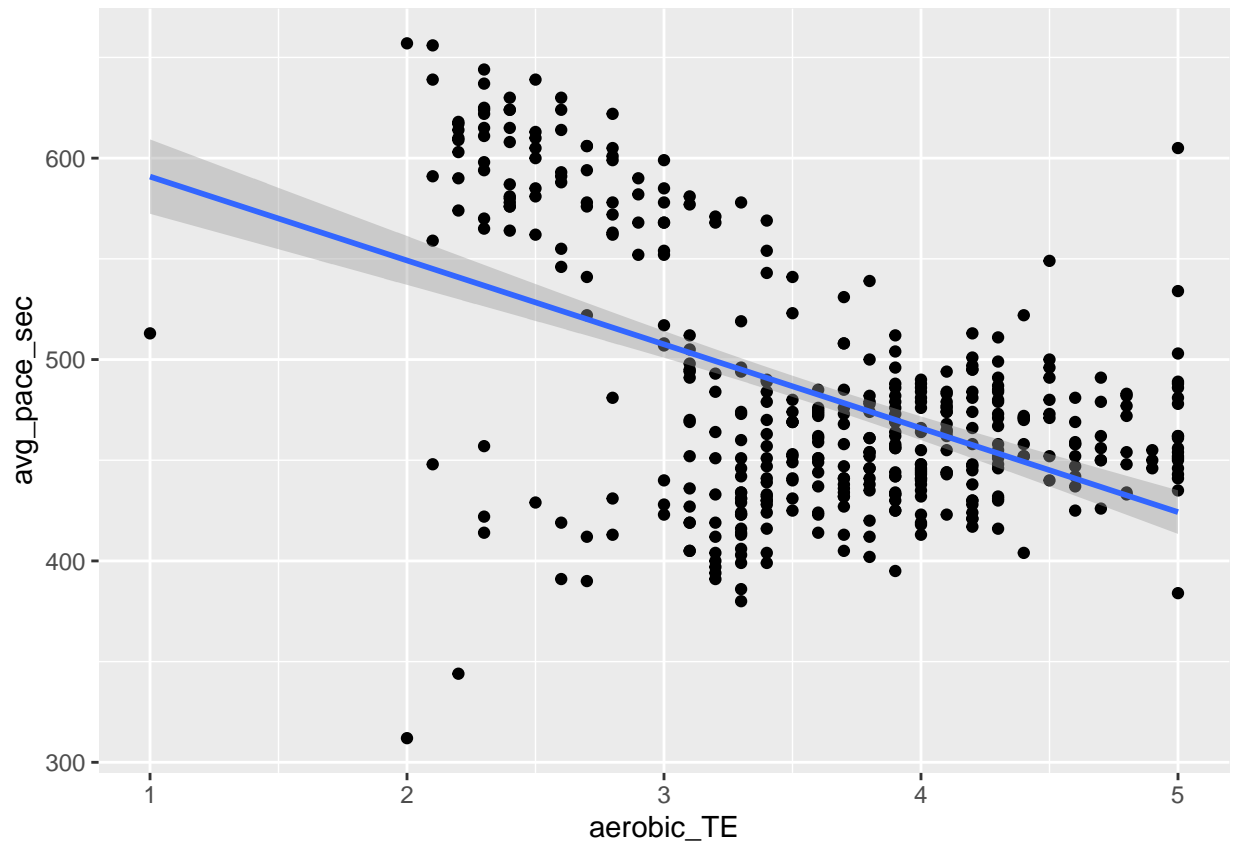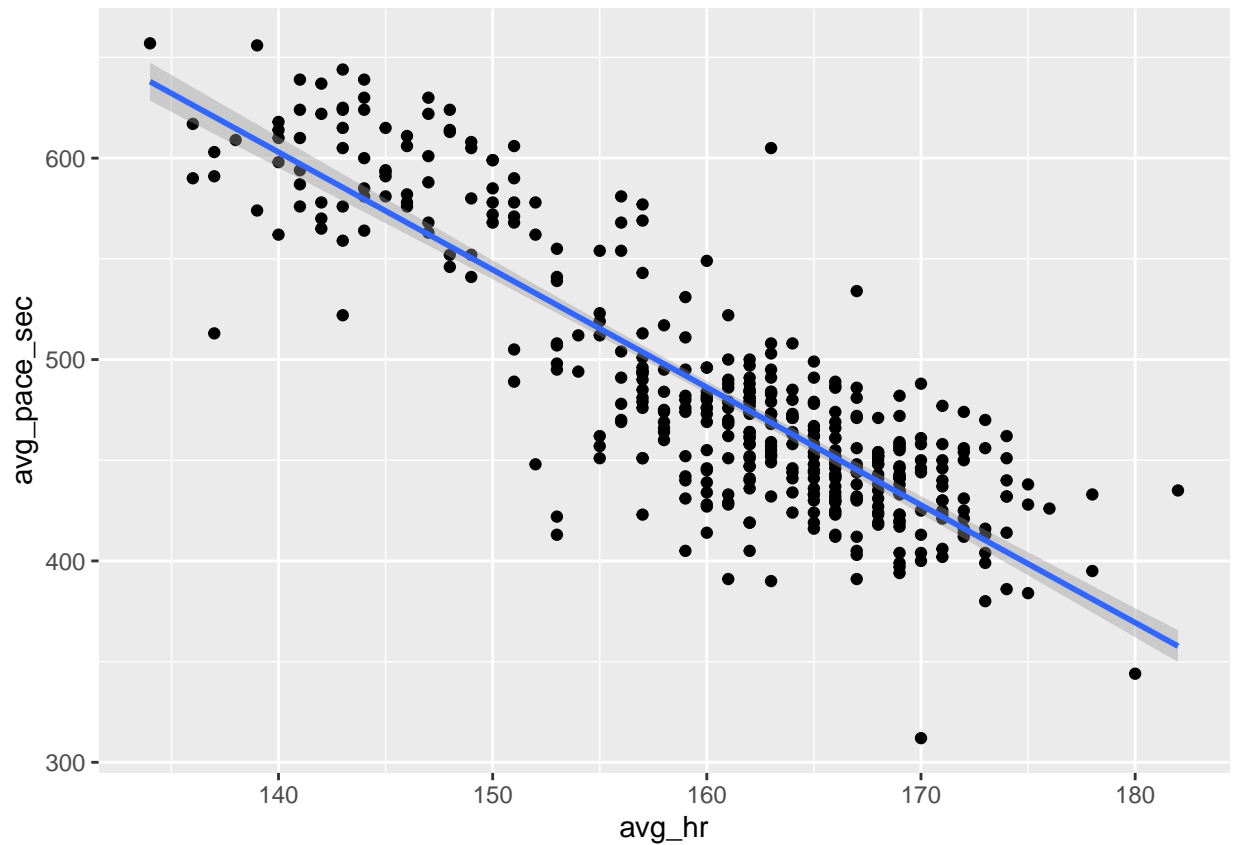
Average heart rate may be another important variable in my project.

```
# Average Pace and Average Heart Rate
hr_ap <- garminRun %>% ggplot(aes(x=avg_hr, y = avg_pace_sec))+
  geom_point()+
  geom_smooth(method = "lm")

hr_ap
```

```
## `geom_smooth()` using formula 'y ~ x'
```

Greater stride length seems to be associated with a greater pace. This seems counter-intuitive if thinking about cadence in this equations, but with good form and mechanics, greater cadence is more likely to be associated with greater stride length.

```
# Stride and Average Pace
st_ap <- garminRun %>% ggplot(aes(x=avg_stride, y = avg_pace_sec))+
  geom_point()+
  geom_smooth(method = "lm")

st_ap
```

```
## `geom_smooth()` using formula 'y ~ x'
```
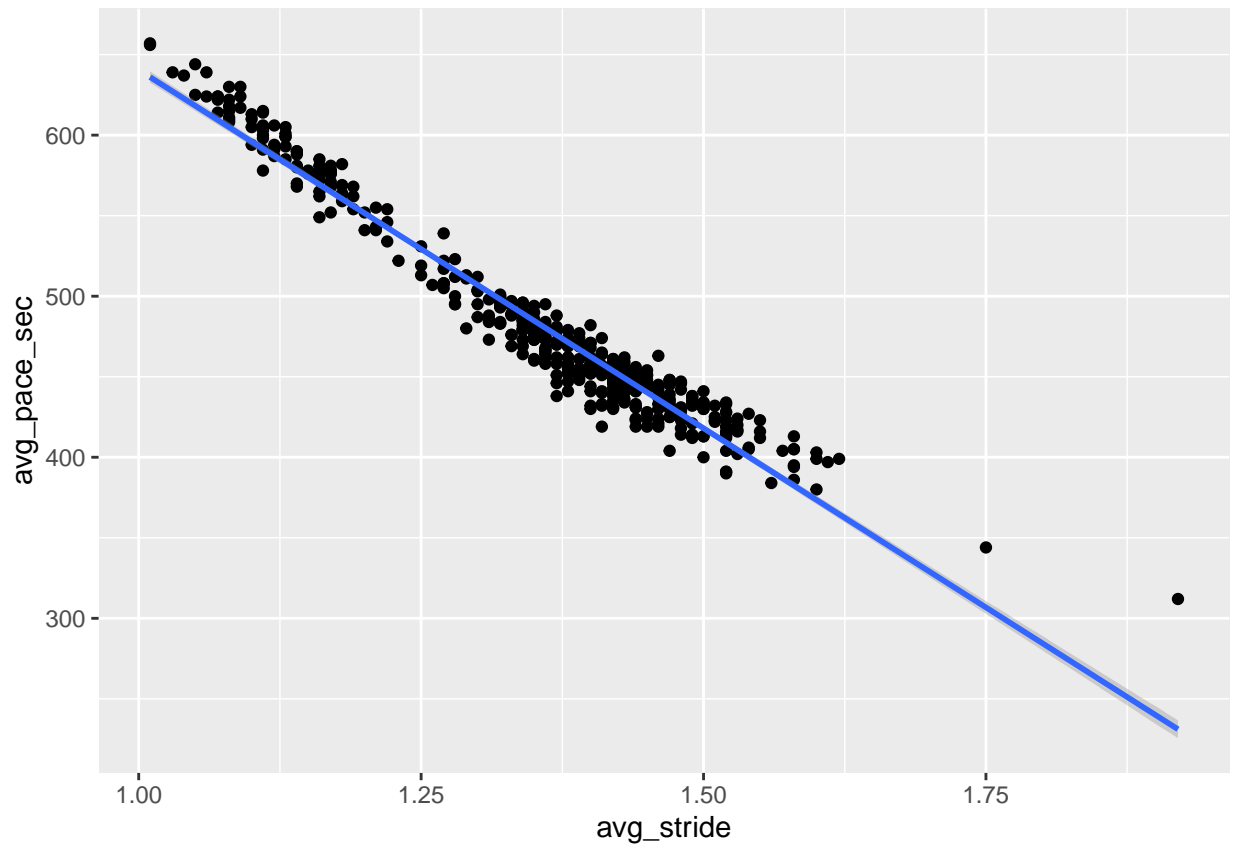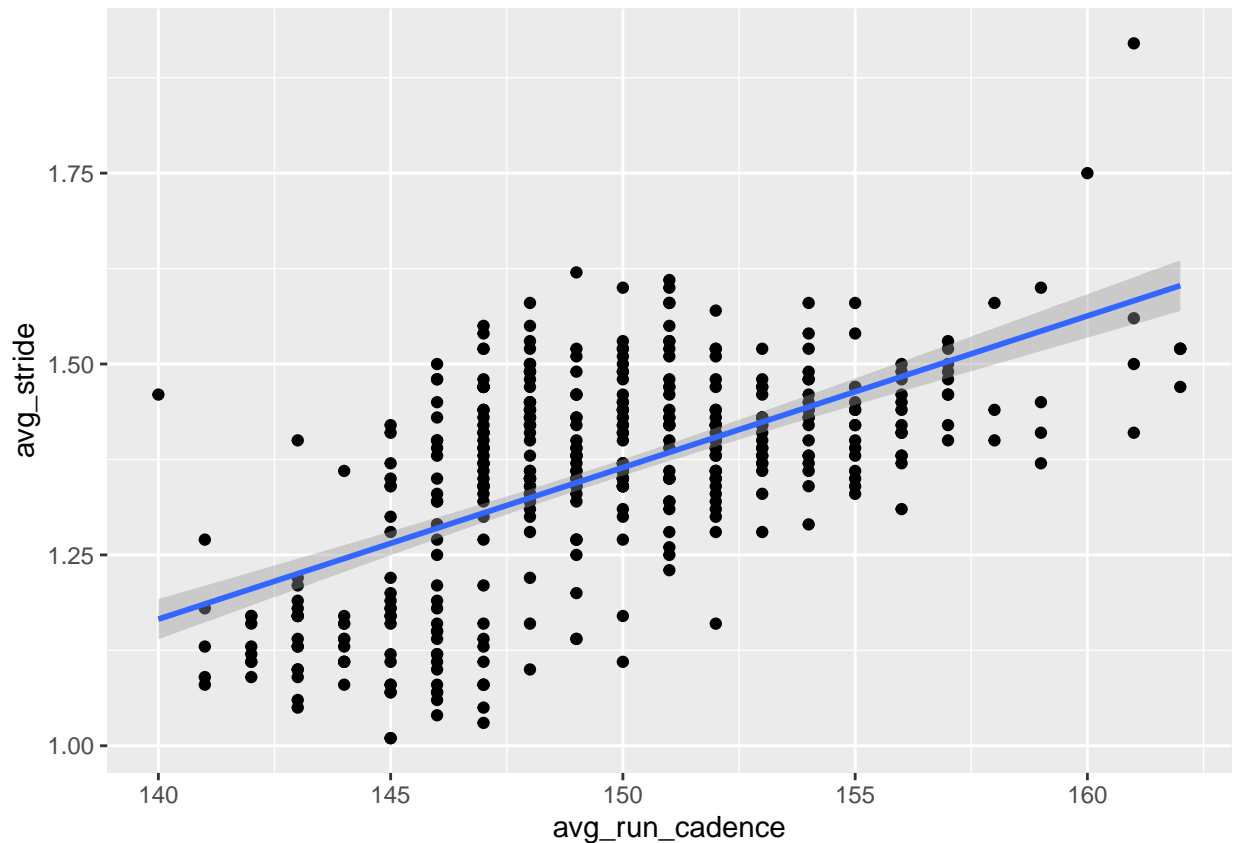
Other important variables Correlation between cadence and stride

```
ca_stride <- garminRun %>% ggplot(aes(x=avg_run_cadence, y=avg_stride))+
  geom_point()+
  geom_smooth(method = "lm")

ca_stride
```

```
## `geom_smooth()` using formula 'y ~ x'
```

Living in the Southern US, we have to deal with high temperatures. Marathons typically aren't held in the summer for a reason. Excessive heat is not only known to decrease performance, but can be dangerous. Although it can't be totally avoided, I try not to run when it's too hot out.

```
# Temperature
# I'm less likely to run far when it gets hot out.
fit1 <- lm(distance ~ temperature, run_df)
summary(fit1)
```

```
##
## Call:
## lm(formula = distance ~ temperature, data = run_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4399 -2.4470 -0.6639  1.2273 20.6184
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.812101   0.556440  15.837  < 2e-16 ***
## temperature -0.046151   0.008301  -5.559 3.79e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.45 on 733 degrees of freedom
##   (335 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.04046,    Adjusted R-squared:  0.03915
## F-statistic: 30.91 on 1 and 733 DF,  p-value: 3.793e-08
```
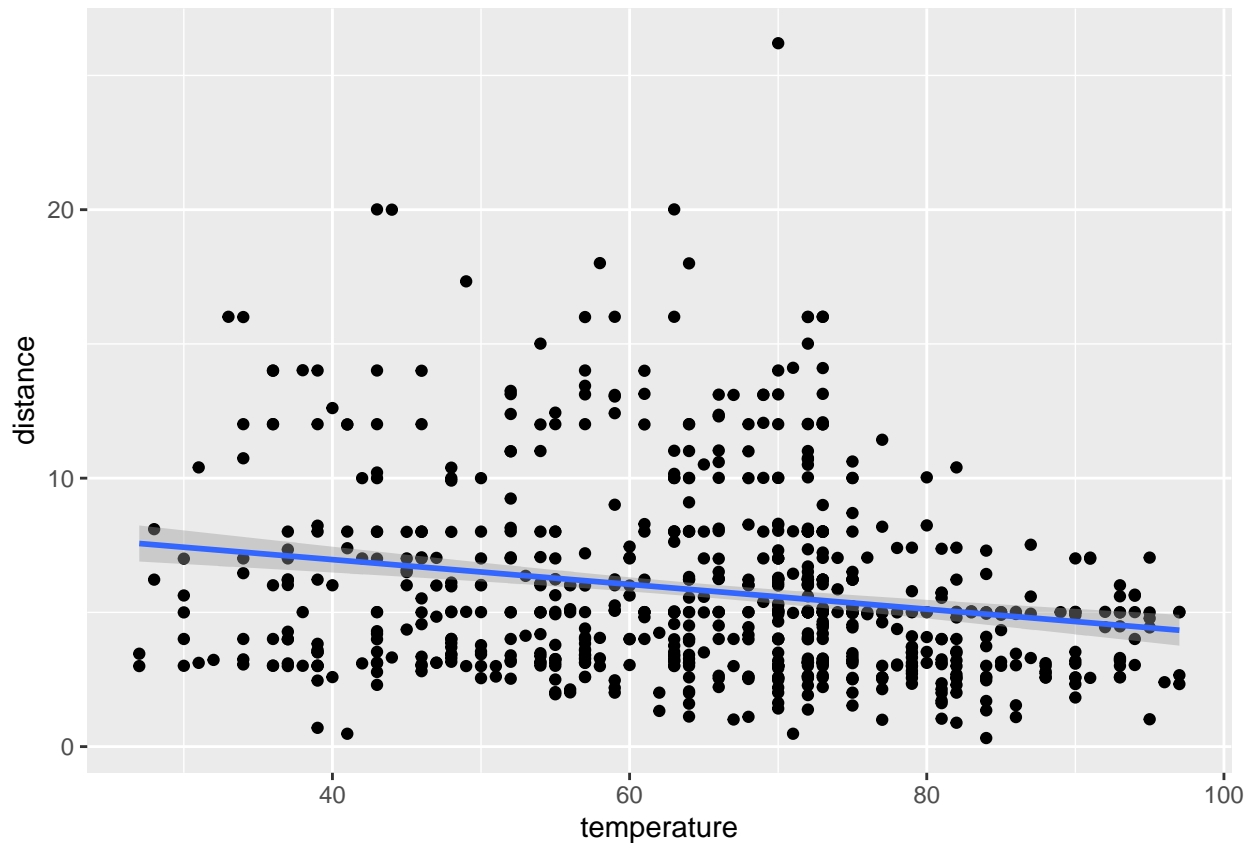
```
temp_dist <- run_df %>% ggplot(aes(x=temperature, y = distance))+
  geom_point()+
  geom_smooth(method = "lm")

temp_dist
```

```
## 'geom_smooth()' using formula 'y ~ x'
```

```
## Warning: Removed 335 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 335 rows containing missing values (geom_point).
```
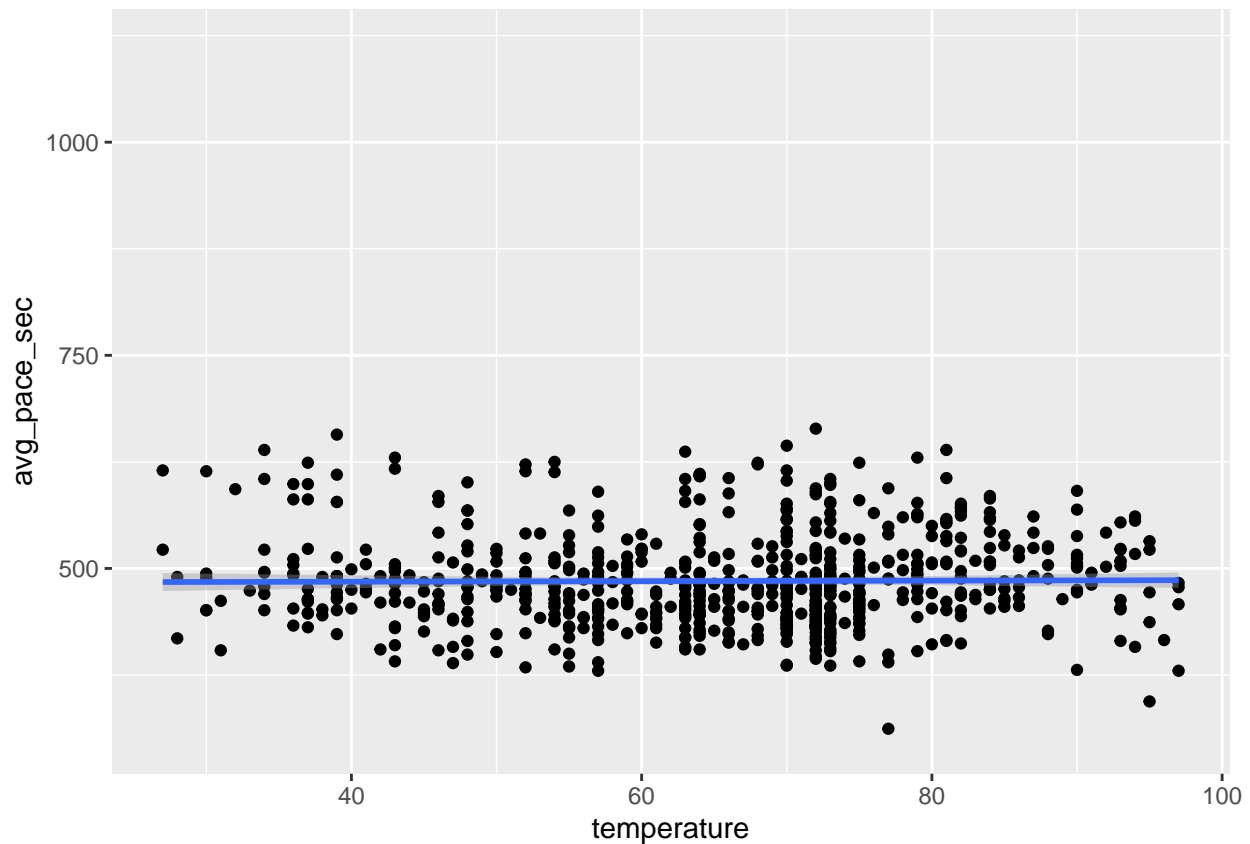


```
temp_pace <- run_df %>% ggplot(aes(x=temperature, y = avg_pace_sec))+
  geom_point()+
  geom_smooth(method = "lm")

temp_pace
```

```
## 'geom_smooth()' using formula 'y ~ x'
```

```
## Warning: Removed 340 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 340 rows containing missing values (geom_point).
```


Hill training is crucial for success in any race and research shows that hill intervals can better prepare runners than any other type of training.

```
#avg_hr, total_ascent
hr_ascent <- lm(avg_hr~total_ascent,garminRun)
summary(hr_ascent)
```

```
##
## Call:
## lm(formula = avg_hr ~ total_ascent, data = garminRun)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.6834  -5.2914   0.7117   6.2895  23.3226
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.559e+02  8.028e-01 194.186  < 2e-16 ***
## total_ascent 1.397e-02  1.994e-03   7.006 9.89e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.836 on 417 degrees of freedom
##   (9 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.1053, Adjusted R-squared:  0.1032
## F-statistic: 49.09 on 1 and 417 DF,  p-value: 9.893e-12
```
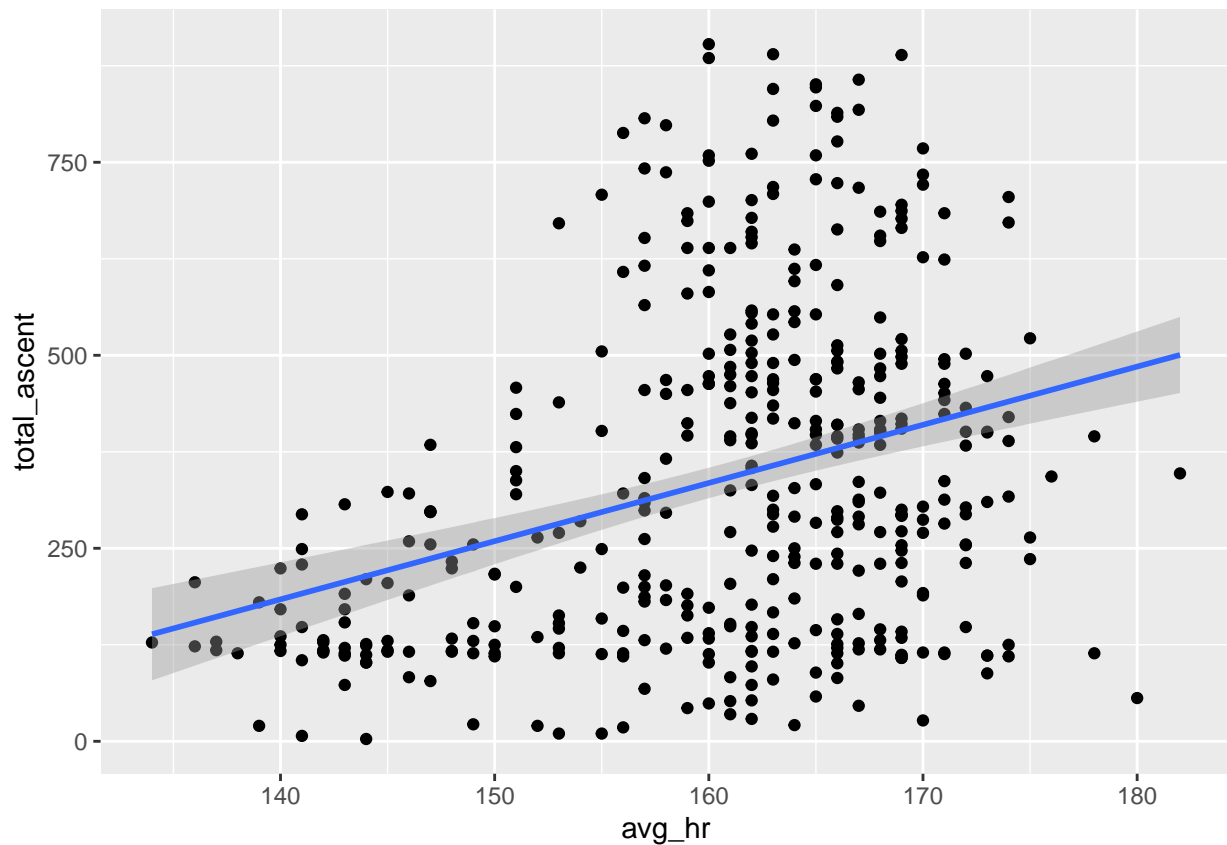
```
hr_ascent_plot <-garminRun %>% ggplot(aes(x=avg_hr, y = total_ascent))+
  geom_point()+
  geom_smooth(method = "lm")

hr_ascent_plot
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 9 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 9 rows containing missing values (geom_point).
```



```
ae_ta <- garminRun %>% ggplot(aes(x=aerobic_TE, y=total_ascent))+
  geom_point()+
  geom_smooth(method = "lm")

ae_ta
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 9 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 9 rows containing missing values (geom_point).
```
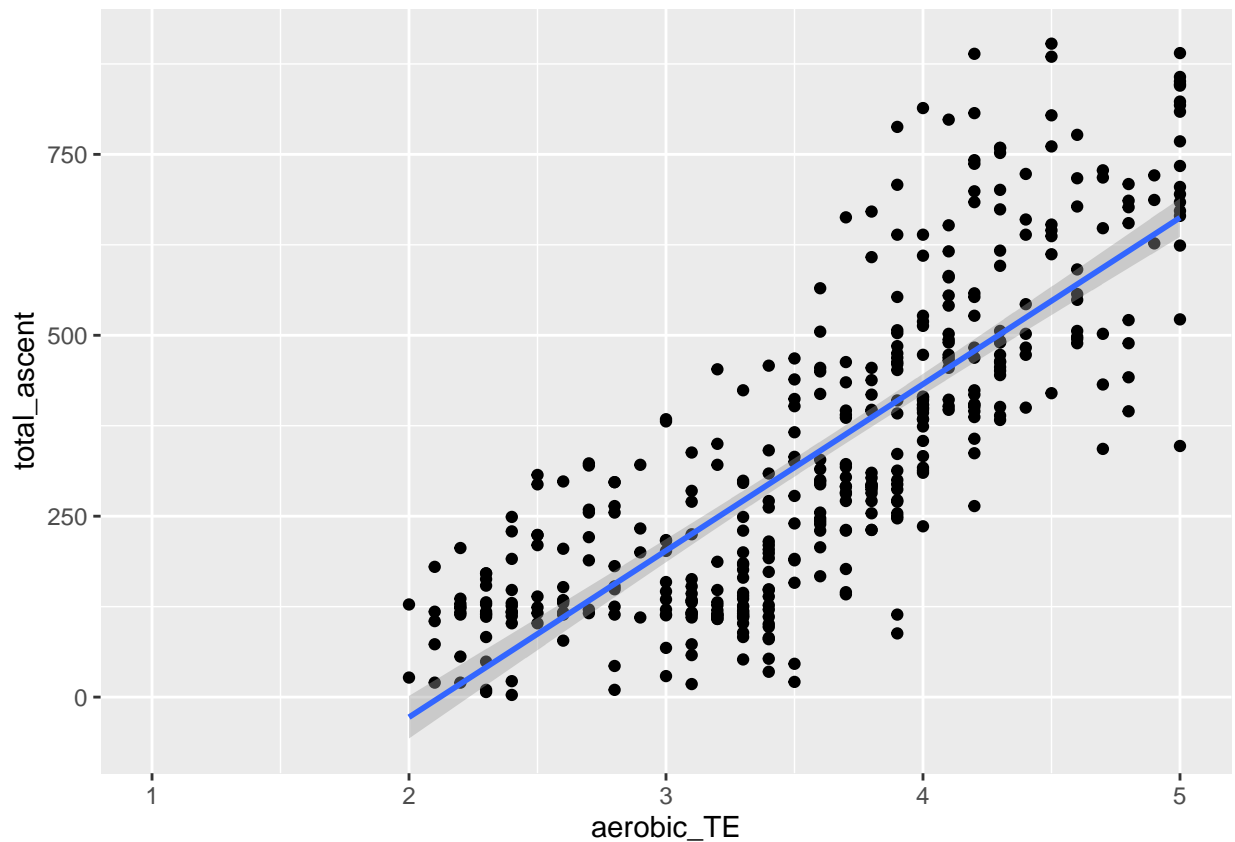

Statistical Modeling

Adding New Variables

In "Human runnign performance from real-world big data," authors Emig and Peltonen create a statistical model based on fields captured by the Polar V800 fitness watch, including measures like p = power, v = velocity, t = time, as well as maximal aerobic power (MAP), VO2 Max, anaerobic endurance. These variables go into their model to predict performance during a marathon.

Since processing my data for the first time, I found even more measurements I was interested in inspecting on Garmin Connect, the online dashboard which hosts the data from my watch. For some reason, this data is readily available on individual run records in the website dashboard, but not included in any csv files I'm allowed to export. Therefore, I created a new script which uses RSelenium to take this data from Garmin Connect. This adds the variables Anaerobic Training Effect (anaerobic_value), Average Speed (avg_spd), Average Moving Speed(avg_moving_spd), and Maximum Speed (max_spd). Additionally, I added the variable for estimated sweat loss out of curiosity.

Being unable to access the precise variables Emig and Peltonen describe, I intend to use similar variables to investigate relationships within my dataset.

Trying models

Since I'm trying to predict performance, I wanted to see which variables would be best for doing so. I started by seeing if it is possible to use a linear regression to predict pace. In my analysis, the best model I was able

to create used all variables and predicted average pace within 5.4 seconds.

```
prelim_pace <- lm(avg_pace_sec ~ ., df)
summary(prelim_pace)
```

Linear Regression

```
##
## Call:
## lm(formula = avg_pace_sec ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.013  -3.313  -0.427   3.011  47.420
##
## Coefficients: (1 not defined because of singularities)
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.185e+03  6.190e+01  19.143  < 2e-16 ***
## distance                 -4.443e+00  1.027e+00  -4.326 1.93e-05 ***
## avg_hr                    3.786e-01  1.501e-01   2.522  0.01206 *
## max_hr                   -8.380e-03  9.638e-02  -0.087  0.93075
## avg_run_cadence          -1.787e+00  3.790e-01  -4.715 3.37e-06 ***
## max_run_cadence           5.933e-03  3.445e-02   0.172  0.86337
## total_ascent             -9.885e-03  1.116e-02  -0.886  0.37613
## total_decent              3.999e-03  1.057e-02   0.378  0.70547
## avg_stride               -2.333e+02  4.015e+01  -5.811 1.29e-08 ***
## min_elevation            -2.627e-02  1.542e-02  -1.703  0.08930 .
## max_elevation             2.642e-02  1.575e-02   1.678  0.09417 .
## best_pace_sec             2.566e-02  1.494e-02   1.718  0.08661 .
## 'sweat_loss(ml)'          9.787e-02  1.862e-02   5.257 2.41e-07 ***
## aerobic_TE               -8.762e+00  2.429e+00  -3.606  0.00035 ***
## aerobic_fctImpacting     -4.257e+00  1.318e+00  -3.231  0.00134 **
## aerobic_fctMaintaining    5.590e+00  2.542e+00   2.199  0.02849 *
## aerobic_fctOverreaching   5.209e+00  2.069e+00   2.518  0.01220 *
## anaerobic_value          -1.135e+00  1.584e+00  -0.716  0.47435
## anaerobic_fctMaintaining  2.063e+00  2.509e+00   0.822  0.41144
## anaerobic_fctNo Benefit   1.926e-01  4.972e+00   0.039  0.96913
## anaerobic_fctSome Benefit 1.513e+00  3.754e+00   0.403  0.68709
## avg_spd                  -2.101e+01  7.205e+00  -2.916  0.00374 **
## max_spd                   6.393e-02  3.708e-01   0.172  0.86321
## short_distanceY          -5.004e-01  2.750e+00  -0.182  0.85569
## middle_distanceY         -1.782e-01  2.044e+00  -0.087  0.93054
## long_distanceY                  NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.525 on 392 degrees of freedom
##   (11 observations deleted due to missingness)
## Multiple R-squared:  0.9901, Adjusted R-squared:  0.9895
## F-statistic:  1641 on 24 and 392 DF,  p-value: < 2.2e-16
```

To begin predicting run performance, an initial linear regression model will be built below using all available data. Based on the preliminary linear regression above, an aerobic training effect that has a high impact (value between 4 and 4.9) is strongly related to average pace. This variable will be the target variable in the logistic regression that follows.

```r
set.seed(456)
# Split data into training and testing sets
df_split <- initial_split(df, prop = 3/4)

train_df <- training(df_split)
test_df <- testing(df_split)

# Create recipe
pace_rec <- recipe(avg_pace_sec ~ ., data = train_df)

summary(pace_rec)
```

```
## # A tibble: 22 x 4
##    variable         type    role      source
##    <chr>            <chr>   <chr>     <chr>
##  1 distance         numeric predictor original
##  2 avg_hr           numeric predictor original
##  3 max_hr           numeric predictor original
##  4 avg_run_cadence  numeric predictor original
##  5 max_run_cadence  numeric predictor original
##  6 total_ascent     numeric predictor original
##  7 total_decent     numeric predictor original
##  8 avg_stride       numeric predictor original
##  9 min_elevation    numeric predictor original
## 10 max_elevation    numeric predictor original
## # ... with 12 more rows
```

```r
lm_pace <- linear_reg() %>%
  set_engine("lm")

pace_wflow <- workflow()%>%
  add_model(lm_pace) %>%
  add_recipe(pace_rec)

pace_fit <- pace_wflow %>%
  fit(data = train_df)

tidy(pace_fit)
```

```
## # A tibble: 26 x 5
##    term            estimate std.error statistic  p.value
##    <chr>              <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)     1166.       73.6      15.8   4.97e-41
##  2 distance          -3.81      1.22     -3.12  2.00e- 3
##  3 avg_hr             0.400     0.181     2.21  2.80e- 2
##  4 max_hr            -0.104     0.119    -0.880 3.80e- 1
##  5 avg_run_cadence   -1.70      0.456    -3.73  2.27e- 4
##  6 max_run_cadence    0.0245    0.0436    0.562 5.75e- 1
```

```
##  7 total_ascent      -0.00996     0.0139    -0.714 4.76e- 1
##  8 total_decent       0.00380     0.0130     0.293 7.70e- 1
##  9 avg_stride      -233.          48.9      -4.76  3.09e- 6
## 10 min_elevation     -0.0261      0.0193    -1.35  1.77e- 1
## # ... with 16 more rows
```

```
predict(pace_fit, test_df)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
## # A tibble: 107 x 1
##     .pred
##     <dbl>
##  1  450.
##  2  448.
##  3  436.
##  4  440.
##  5  397.
##  6  414.
##  7  427.
##  8  435.
##  9  433.
## 10  448.
## # ... with 97 more rows
```

```
pace_aug <- augment(pace_fit, test_df)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
pace_aug %>% select(avg_pace_sec, .pred)
```

```
## # A tibble: 107 x 2
##    avg_pace_sec .pred
##           <dbl> <dbl>
##  1          447  450.
##  2          449  448.
##  3          432  436.
##  4          438  440.
##  5          391  397.
##  6          414  414.
##  7          419  427.
##  8          432  435.
##  9          430  433.
## 10          444  448.
## # ... with 97 more rows
```

The R Mean-Squared Error for this model is 5.41. In other words, this model can predict average pace within 5.41 seconds.

```
pace_error <- pace_aug %>%
  rmse(truth = avg_pace_sec, .pred)

pace_error
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        5.41
```

Logistic Regression

Working with this data, I hypothesized that aerobic training effects that had a high impact (value of 4.0 - 4.9) would be a good target variable for a predictive model. The logistic regression models were both successful with ROC_AUC values greater than .7.

All Variables   This first logistic regression is meant to predict whether an activity highly impacts aerobic training. This variable is relevant because it is the highest measure for aerobic conditioning without being over-reaching. In this analysis, calories and other variables related to aerobic training effect were removed.

```
set.seed(456)

df2 <- df %>% mutate(high_impact = ifelse(aerobic_fct == "Highly Impacting", 1,0))
df2$high_impact <- factor(df2$high_impact)

# Split data into training and testing sets
df2_split <- initial_split(df2, prop = 3/4)

train_df2 <- training(df2_split)
test_df2 <- testing(df2_split)

# Create recipe. Use all variables except aerobic_TE and related
aerobic_rec <- recipe(high_impact ~ short_distance + middle_distance + long_distance +
                        max_spd + avg_spd + anaerobic_value + `sweat_loss(ml)` + best_pace_sec +
                        avg_pace_sec + max_elevation + min_elevation + avg_stride +
                        total_decent + total_ascent + max_run_cadence + avg_run_cadence +
                        max_hr + avg_hr + distance, data = train_df2)

summary(aerobic_rec)
```

```
## # A tibble: 20 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 short_distance  nominal predictor original
##  2 middle_distance nominal predictor original
##  3 long_distance   nominal predictor original
##  4 max_spd         numeric predictor original
##  5 avg_spd         numeric predictor original
##  6 anaerobic_value numeric predictor original
##  7 sweat_loss(ml)  numeric predictor original
##  8 best_pace_sec   numeric predictor original
```

```
##  9 avg_pace_sec    numeric predictor original
## 10 max_elevation   numeric predictor original
## 11 min_elevation   numeric predictor original
## 12 avg_stride      numeric predictor original
## 13 total_decent    numeric predictor original
## 14 total_ascent    numeric predictor original
## 15 max_run_cadence numeric predictor original
## 16 avg_run_cadence numeric predictor original
## 17 max_hr          numeric predictor original
## 18 avg_hr          numeric predictor original
## 19 distance        numeric predictor original
## 20 high_impact     nominal outcome   original
```

```
log_reg <- logistic_reg() %>%
  set_engine("glm")

aero_wkfl <- workflow()%>%
  add_model(log_reg) %>%
  add_recipe(aerobic_rec)

aero_fit <- aero_wkfl %>%
  fit(data = train_df2)

tidy(aero_fit)
```

```
## # A tibble: 20 x 5
##    term               estimate std.error statistic  p.value
##    <chr>                 <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)        -111.     107.       -1.04    0.297
##  2 short_distanceY       0.114    1.34      0.0851  0.932
##  3 middle_distanceY      1.89     0.900     2.09    0.0363
##  4 long_distanceY       NA       NA        NA      NA
##  5 max_spd              -0.811    1.05     -0.775   0.439
##  6 avg_spd             -17.1      5.36     -3.19    0.00144
##  7 anaerobic_value      -0.670    0.425    -1.58    0.115
##  8 'sweat_loss(ml)'     -0.00296  0.0227   -0.130   0.896
##  9 best_pace_sec        -0.0278   0.0347   -0.800   0.423
## 10 avg_pace_sec         -0.0252   0.0851   -0.297   0.767
## 11 max_elevation        -0.0283   0.0118   -2.40    0.0165
## 12 min_elevation         0.0288   0.0102    2.81    0.00495
## 13 avg_stride           88.9     32.8       2.71    0.00673
## 14 total_decent          0.00536  0.00665   0.806   0.420
## 15 total_ascent          0.00340  0.00672   0.506   0.613
## 16 max_run_cadence       0.00974  0.0226    0.432   0.666
## 17 avg_run_cadence       0.747    0.298     2.50    0.0123
## 18 max_hr                0.0856   0.0742    1.15    0.249
## 19 avg_hr                0.115    0.0643    1.79    0.0734
## 20 distance              0.279    1.11      0.252   0.801
```

```
predict(aero_fit, test_df2)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## # A tibble: 107 x 1
##    .pred_class
##    <fct>
##  1 0
##  2 0
##  3 0
##  4 0
##  5 0
##  6 0
##  7 0
##  8 1
##  9 0
## 10 1
## # ... with 97 more rows
```

```r
aero_aug <- augment(aero_fit, test_df2)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```
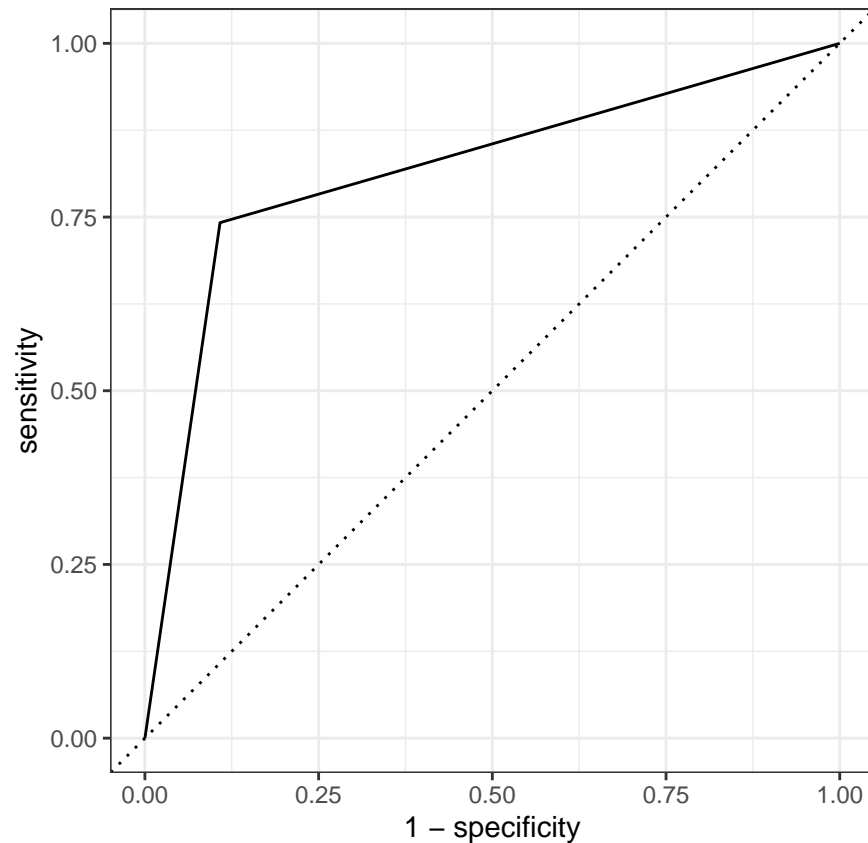
```r
aero_aug %>% select(high_impact, .pred_class)
```

```
## # A tibble: 107 x 2
##    high_impact .pred_class
##    <fct>       <fct>
##  1 0           0
##  2 0           0
##  3 0           0
##  4 0           0
##  5 0           0
##  6 0           0
##  7 0           0
##  8 1           1
##  9 0           0
## 10 1           1
## # ... with 97 more rows
```

```r
aero_aug$.pred_class <- as.character(aero_aug$.pred_class)
aero_aug$.pred_class <- as.numeric(aero_aug$.pred_class)

aero_aug %>%
  roc_curve(truth = high_impact, .pred_class, event_level="second") %>%
  autoplot()
```

```
aero_aug %>%
  roc_auc(truth = high_impact, .pred_class, event_level="second")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.817
```

```
# Create recipe. Use all variables except aerobic_TE and related
aerobic_rec2 <- recipe(high_impact ~ middle_distance + avg_spd + min_elevation + avg_stride +
                       avg_run_cadence + avg_hr, data = train_df2)

summary(aerobic_rec2)
```

Most significant variables

```
## # A tibble: 7 x 4
##   variable        type    role      source
##   <chr>           <chr>   <chr>     <chr>
## 1 middle_distance nominal predictor original
## 2 avg_spd         numeric predictor original
## 3 min_elevation   numeric predictor original
```

```
## 4 avg_stride       numeric predictor original
## 5 avg_run_cadence numeric predictor original
## 6 avg_hr           numeric predictor original
## 7 high_impact      nominal outcome   original
```

```
log_reg <- logistic_reg() %>%
  set_engine("glm")

aero_wkfl2 <- workflow()%>%
  add_model(log_reg) %>%
  add_recipe(aerobic_rec2)

aero_fit2 <- aero_wkfl2 %>%
  fit(data = train_df2)

tidy(aero_fit2)
```

```
## # A tibble: 7 x 5
##   term              estimate std.error statistic  p.value
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)        -133.     34.0       -3.93 8.63e- 5
## 2 middle_distanceY     2.72     0.358      7.60 3.07e-14
## 3 avg_spd            -14.7      4.15      -3.55 3.85e- 4
## 4 min_elevation       -0.0101   0.00576   -1.75 7.94e- 2
## 5 avg_stride          77.6     23.3        3.33 8.57e- 4
## 6 avg_run_cadence      0.764    0.224      3.40 6.63e- 4
## 7 avg_hr               0.185    0.0444     4.17 2.99e- 5
```

```
predict(aero_fit2, test_df2)
```

```
## # A tibble: 107 x 1
##     .pred_class
##     <fct>
##  1 0
##  2 0
##  3 0
##  4 0
##  5 0
##  6 0
##  7 0
##  8 1
##  9 0
## 10 1
## # ... with 97 more rows
```

```
aero_aug2 <- augment(aero_fit2, test_df2)

aero_aug2 %>% select(high_impact, .pred_class)
```
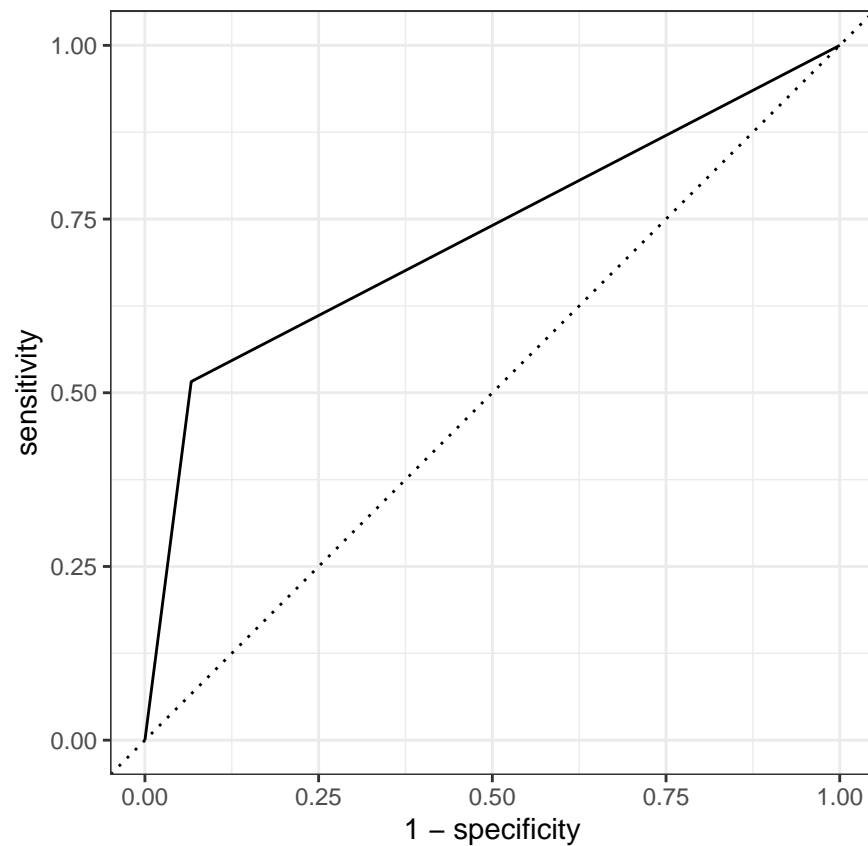
```
## # A tibble: 107 x 2
##    high_impact .pred_class
##    <fct>        <fct>
```

```
##  1 0          0
##  2 0          0
##  3 0          0
##  4 0          0
##  5 0          0
##  6 0          0
##  7 0          0
##  8 1          1
##  9 0          0
## 10 1          1
## # ... with 97 more rows
```

```
aero_aug2$.pred_class <- as.character(aero_aug2$.pred_class)
aero_aug2$.pred_class <- as.numeric(aero_aug2$.pred_class)

aero_aug2 %>%
  roc_curve(truth = high_impact, .pred_class, event_level = "second") %>%
  autoplot()
```



```
aero_aug2 %>%
  roc_auc(truth = high_impact, .pred_class, event_level = "second")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.725
```

Both of these models are acceptable for predicting whether a run highly impacts performance. These analyses provide a good starting point for building a more complex model that can predict good performance. The possible next step is to use k-fold cross validation to predict when good performance will happen given a series of events.

References

Emig, T., Peltonen, J. Human running performance from real-world big data. Nat Commun 11, 4936 (2020). https://doi.org/10.1038/s41467-020-18737-6