# Stats Analysis

## Introduction

I like to go running from time to time to keep fit. I record my running sessions using my phone to track my distance and pace. I'm quite casual with my running and don't really follow a formalised schedule. I decided to extract some of my running data and conduct some analysis to see if I could uncover some interesting insights from it. The data used in this analysis comes from my past running sessions between 14 Jan 2020 and 10 May 2020. During this time I was training to reach a distance goal of a half marathon, which is 21 kilometers.

The finish line at Melbourne Marathon

## Preliminary preparations

Here is a sample of what the raw data looks like. It includes a summary of the field names and data types. This dataset contains 35 running records in total and 12 different fields.

### Read data from file

```
f <- "running_data.csv"
raw_data <- read.csv("running_data.csv",header=TRUE)
head(raw_data)
```

```
##           datetime   weekday       time distKm totalTimeMin avgPaceMinKm
## 1  14/01/2020 8:30   Tuesday 8:30:00 am   3.09     20.27000     6.533333
## 2  16/01/2020 9:19  Thursday 9:19:34 am   1.70     10.32195     6.033333
## 3  18/01/2020 8:49  Saturday 8:49:40 am   3.07     20.21533     6.566667
## 4 20/01/2020 20:05    Monday 8:05:11 pm   4.31     30.29150     7.016667
## 5 22/01/2020 19:00 Wednesday 7:00:44 pm   4.29     31.66382     7.366667
## 6 24/01/2020 13:07    Friday 1:07:27 pm   2.47     16.59635     6.700000
##   totalAscentKm       weather temp nPastRuns daysSinceLastRun deltaDistPastRun
## 1         0.490        Cloudy   17         0                0             0.00
## 2         0.287          Fair   18         1                2            -0.82
## 3         0.789 Partly Cloudy   18         2                2             0.45
## 4         0.518          Fair   22         3                2             0.29
## 5         1.316          Fair   19         4                2            -0.01
## 6         0.431 Mostly Cloudy   26         5                2            -0.74
```

```
data <- copy(raw_data)
str(data)
```

```
## 'data.frame':    35 obs. of  12 variables:
##  $ datetime        : chr  "14/01/2020 8:30" "16/01/2020 9:19" "18/01/2020 8:49" "20/01/2020 20:05" .
##  $ weekday         : chr  "Tuesday" "Thursday" "Saturday" "Monday" ...
##  $ time            : chr  "8:30:00 am" "9:19:34 am" "8:49:40 am" "8:05:11 pm" ...
##  $ distKm          : num  3.09 1.7 3.07 4.31 4.29 2.47 5.03 2.75 3.42 5.02 ...
##  $ totalTimeMin    : num  20.3 10.3 20.2 30.3 31.7 ...
##  $ avgPaceMinKm    : num  6.53 6.03 6.57 7.02 7.37 ...
##  $ totalAscentKm   : num  0.49 0.287 0.789 0.518 1.316 ...
##  $ weather         : chr  "Cloudy" "Fair" "Partly Cloudy" "Fair" ...
##  $ temp            : int  17 18 18 22 19 26 22 23 24 22 ...
##  $ nPastRuns       : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ daysSinceLastRun: int  0 2 2 2 2 2 2 5 2 1 ...
##  $ deltaDistPastRun: num  0 -0.82 0.45 0.29 -0.01 -0.74 0.51 -0.83 0.2 0.32 ...
```

## Data preparation

Some fields are transformed into numerical values so that they can be analysed numerically:

1. The weekdays field is enumerated so that Monday=1, Tuesday=2, Wednesday=3…Sunday=7.
2. The time field is converted to 24 hour format and converted to integers, e.g. 1830000 = 6:30pm.

```
data$weekdayOrd <- factor(data$weekday, levels = c("Monday", "Tuesday", "Wednesday",
                          "Thursday", "Friday", "Saturday", "Sunday"),
            ordered = TRUE)


data["weekdayInt"] = as.integer(data$weekdayOrd)

head(data[c("weekday","weekdayInt")])
```

```
##     weekday weekdayInt
## 1   Tuesday          2
## 2  Thursday          4
## 3  Saturday          6
## 4    Monday          1
## 5 Wednesday          3
## 6    Friday          5
```

```
# Parse hours, minutes, seconds and am/pm, remove leading zeros and convert
# the 24 hour string to integers
data["time24hour"]=strtoi(str_remove(format(strptime(data$time, "%I:%M:%S %p"),
                                             format="%H%M%S"), "^0+"))

head(data[c("time","time24hour")])
```

```
##          time time24hour
## 1 8:30:00 am      83000
## 2 9:19:34 am      91934
## 3 8:49:40 am      84940
## 4 8:05:11 pm     200511
## 5 7:00:44 pm     190044
## 6 1:07:27 pm     130727
```

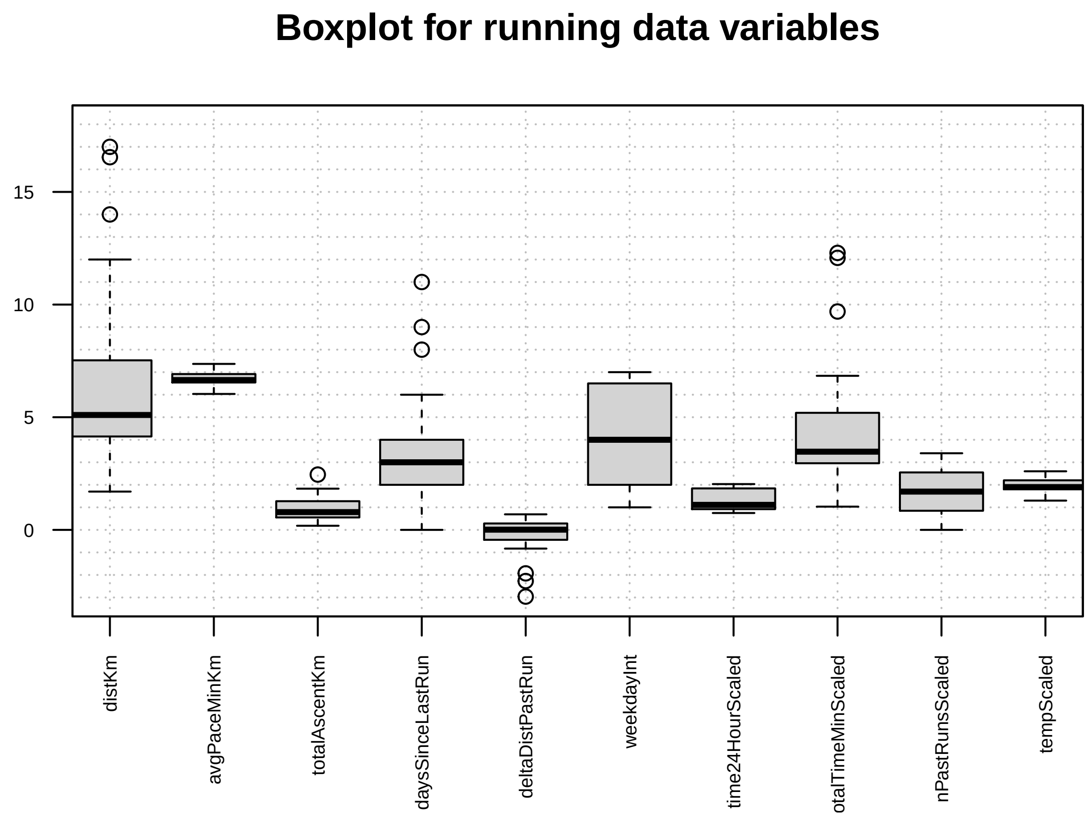
## Exploratory data analysis

### Boxplot

Box plots are drawn to show the distributions for all the numeric values in the dataset. The numerical ranges of the data vary greatly, so some fields are scaled by multiples of 10 just for visualisation purposes. This is so that they can be plotted on the same graph and so the distributions and outliers of the data can be more easily seen.

There are outliers for the total time minutes and the distance, however, during the time I was training to run a half marathon, so I know that there will be some records with longer distances/times as I approached my distance goal. Thus, these outliers are valid and will be kept in the data.There are also 3 outliers for the "number of days since last run" field, and I have explained these outliers in the analysis later on.

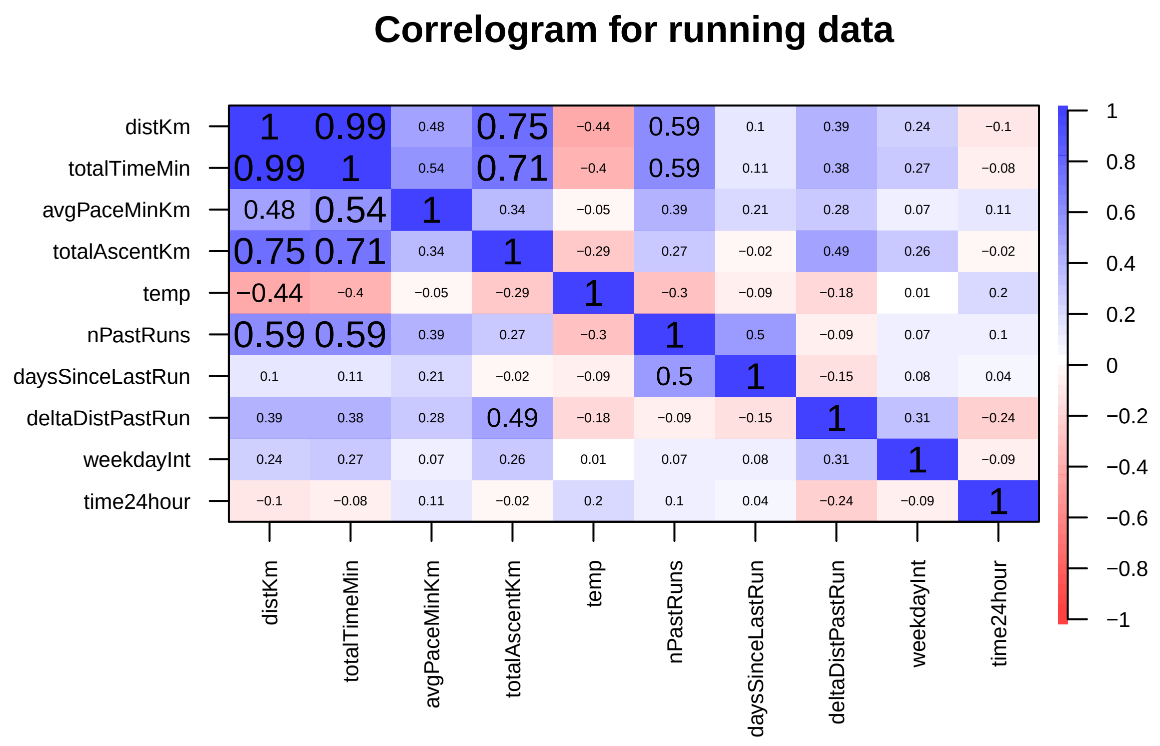
```
# Extract numeric/integer columns only as a separate dataset
numeric_data = data[sapply(data, is.numeric)]

numeric_scaled = copy(numeric_data)
numeric_scaled["time24HourScaled"]= numeric_data$time24hour/100000
numeric_scaled["totalTimeMinScaled"]= numeric_data$totalTimeMin/10
numeric_scaled["nPastRunsScaled"]= numeric_data$nPastRuns/10
numeric_scaled["tempScaled"]=numeric_data$temp/10
numeric_scaled = numeric_scaled[, names(numeric_scaled) != "time24hour"]
numeric_scaled = numeric_scaled[, names(numeric_scaled) != "totalTimeMin"]
numeric_scaled = numeric_scaled[, names(numeric_scaled) != "nPastRuns"]
numeric_scaled = numeric_scaled[, names(numeric_scaled) != "temp"]

# Draw an empty plot first, so that the grid can be drawn underneath
# and the box plot can be drawn on top
x <- 1:10
plot(x,x*1,type='n',main="Boxplot for running data variables",
     ylab="",xlab="",xaxt='n',yaxt='n',ylim=c(-3,18))
abline(h=-3:18,lty=3,col='grey')
abline(v=1:18,lty=3,col='grey')
boxplot(x=as.list(numeric_scaled),las=2,add=TRUE,cex.axis=0.55)
```



### Correlogram

A correlogram is drawn to show the strength of linear relationships between all variables by displaying the correlation coefficients. As expected, there are strong correlations between the distance of run and the time taken to complete a run. There are strong correlations between the total km's ascended/descended and the run distance. This is because when I run longer distances, I travel further out from my usual route and the landscape is hillier. Interestingly, there exists a positive correlation between temperature and total distance, which aligns with my preference of running in cooler temperatures compared to warmer temperatures. There also is some positive correlation between the "delta distance from past run" and the total ascent. This makes sense, because if I increase my distance from the last run, I am probably feeling more energetic and ambitious and so I will choose to run routes with more slopes. If I run a shorter distance compared to my past run, I am probably feeling tired that day and will try to avoid routes with hills.

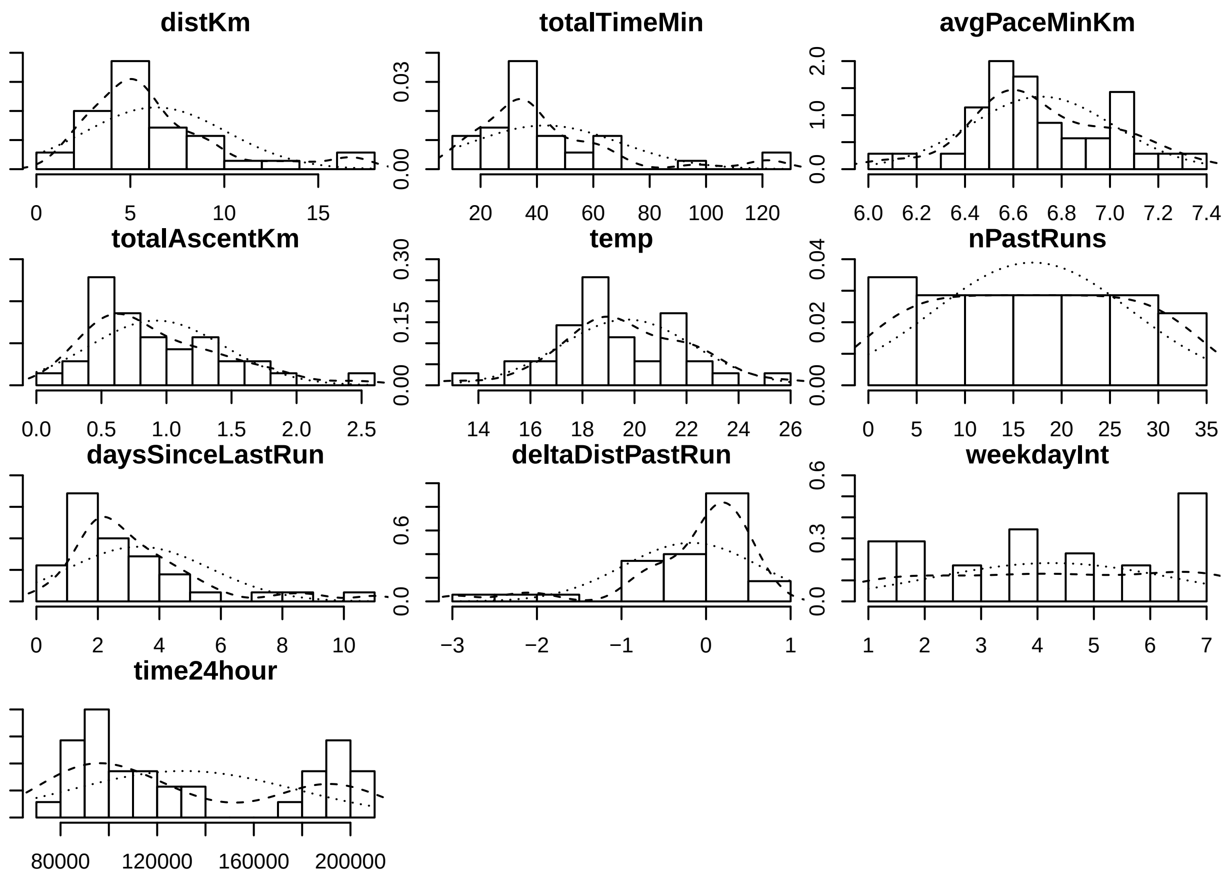
```
corPlot(numeric_data, cex = 0.8,cex.axis=0.7,las=2,main="Correlogram for running data")
```



### Histograms

Histograms are drawn for all the numerical variables in the dataset to see how the values are distributed. Some data fields appear to roughly follow a Gaussian distribution, such as average pace and temperature. The histogram for the time24hour variable shows that I tend to start running either before 12pm or after 4:30pm, avoiding the hottest part of the day. nPastRuns is uniformally distributed because this is simply numbered sequentially from 0 to 35, representing the number of past runs that I have run.

```
multi.hist(numeric_data,breaks=10)
```



## Statistical analysis

### Estimating my average running pace

I tend to run at an easy, comfortable pace and I do not worry about increasing my speed. It can be assumed that my average pace is roughly Gaussian distributed - this is also shown by the histogram visualisation above. My data shows average pace, which is the average time taken to complete each kilometer for each run. These records will be used as sample distributions to estimate my true average pace.

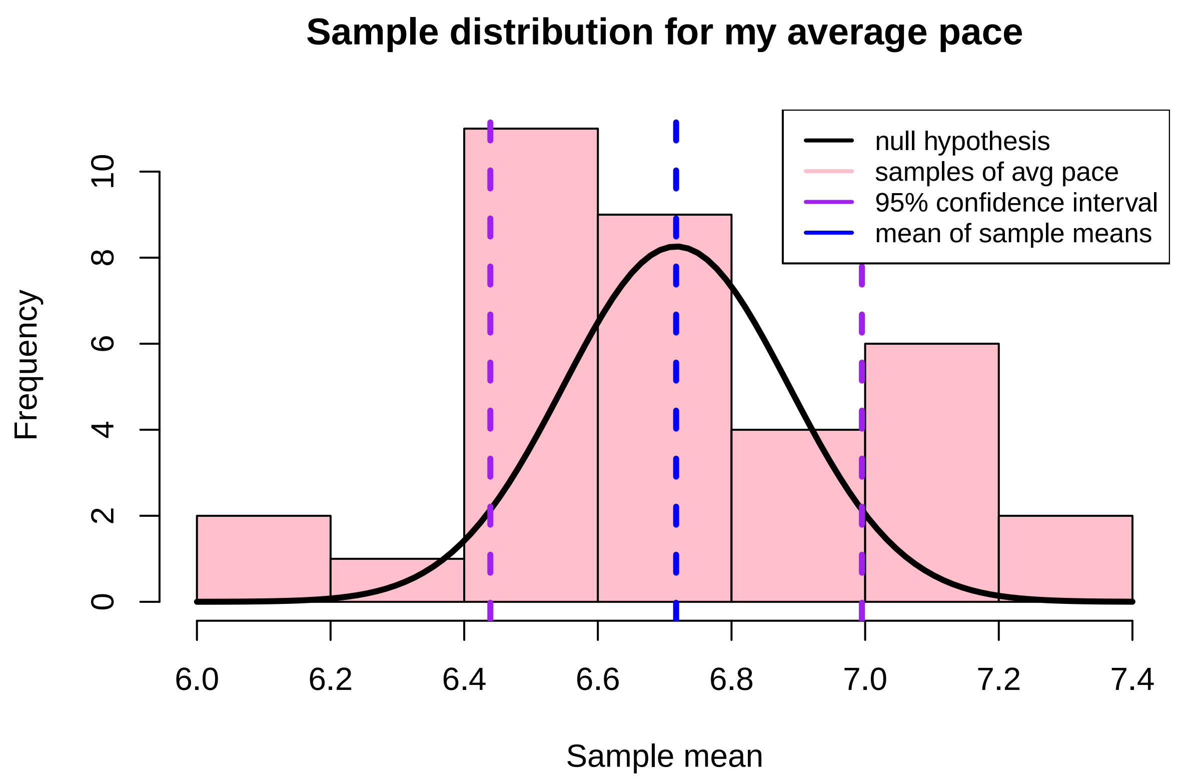
The distribution of sample means is modelled as a Gaussian distribution: $\bar{x} \sim N(\mu, \frac{1}{\sqrt{n}})$ where $\mu$ is the mean of all sample means and the standard deviation is the standard error of the mean.

Using a 95% confidence interval, it is concluded that my true average pace lies somewhere between 6.439 min/km and 6.995 min/km.

```
min_sample_avg_pace = min(data$avgPaceMinKm)
max_sample_avg_pace = max(data$avgPaceMinKm)
mean_sample_avg_pace = mean(data$avgPaceMinKm)
mean_sample_avg_pace
```

```
## [1] 6.717143
```

```
x <-seq(from=6,to=7.4,len=100)
width=0.25
n <-length(data$avgPaceMinKm)
SEM <-sd(data$avgPaceMinKm)/sqrt(n)
SEM <-1/sqrt(n)
hist(data$avgPaceMinKm,col="pink",xlab="Sample mean",
     main="Sample distribution for my average pace")
points(x,n*0.1*dnorm(x,mean=mean_sample_avg_pace,sd=SEM),
     type="l",lwd=3,main="Sample distribution for average pace")
crit_val_lower = qnorm(0.05,mean=mean_sample_avg_pace,sd=SEM)
crit_val_upper = qnorm(0.05,mean=mean_sample_avg_pace,sd=SEM,lower.tail=FALSE)
abline(v=crit_val_lower,lwd=3,lty=2,col='purple')
abline(v=crit_val_upper,lwd=3,lty=2,col='purple')
abline(v=mean_sample_avg_pace, lwd=3, lty=2,col='blue')
legend("topright",lwd=2,cex=0.8,col=c("black","pink","purple","blue"),bg="white",
       legend=c("null hypothesis","samples of avg pace","95% confidence interval"
                ,"mean of sample means"))
```



### The number of days rested between runs

I am a causal runner and I just go for a run whenever I feel like it. I normally need a few days between runs to rest and regain energy and motivation to keep running. I wondered if I maintain any sort of consistency with the number of days I choose to rest between runs.

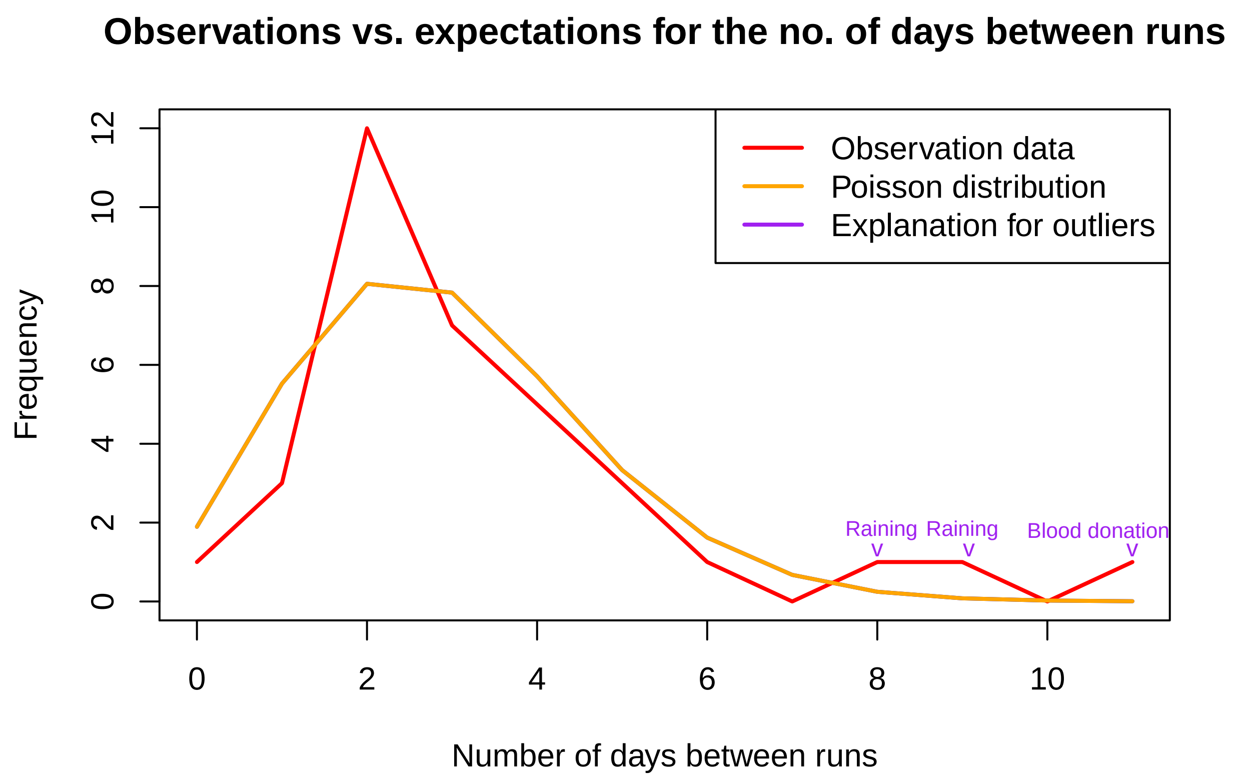
To test this, I propose a null hypothesis: the number of days that I rest between runs is modelled as a Poisson distribution.

A Poisson distribution is used to calculate the expected number of rest days. The mean/variance parameter for the Poisson distribution, $\lambda$, is calculated as the average number of rest days across my entire dataset.

The expectations (from a Poisson distribution) and the true observations are used to calculate the Badness of fit. Then, the chi-squared test is used to test the null hypothesis.

The chi-squared test calculated a p-value of $p = 0$, which is less than the standard threshold of 5%. This means that the null hypothesis can be rejected, inferring that the the number of days I rested between runs does not belong to a Poisson distribution.

```
observations_raw = table(data$daysSinceLastRun)
# Fill in observations table with categories with zero frequencies
observations_raw["7"] = 0
observations_raw["10"] = 0

observations=(c(observations_raw[1:7],
                observations_raw[11],observations_raw[8:9],
                observations_raw[12],observations_raw[10]))


minDays = min(data$daysSinceLastRun)
maxDays = max(data$daysSinceLastRun)

daysSinceLastRunRange = minDays:maxDays

nRows = length(data$daysSinceLastRun)
mean_daysSinceLastRun = mean(observations)

# Here is the expected values for days since last run
expectations_pois = dpois(daysSinceLastRunRange,lambda=mean_daysSinceLastRun)*nRows
plot(daysSinceLastRunRange,
     expectations_pois,
     type='l',ylim=c(0,12),col='blue',lwd=2,
     xlab="Number of days between runs", ylab="Frequency",
     main="Observations vs. expectations for the no. of days between runs")

points(daysSinceLastRunRange,
       observations,
     type='l',col='red',lwd=2)


points(daysSinceLastRunRange,
       expectations_pois,
       type='l',col='orange',lwd=2)

legend("topright",lwd=2, col=c("red","orange","purple"),bg="white",
       legend=c("Observation data",
                "Poisson distribution","Explanation for outliers"))

text(8.05,1.8,'Raining',col='purple',cex=0.7)
text(8,1.3,'v',col='purple',cex=0.75)

text(9,1.8,'Raining',col='purple',cex=0.7)
text(9,1.3,'  v',col='purple',cex=0.75)

text(10.6,1.8,'Blood donation',col='purple',cex=0.7)
text(11,1.3,'v',col='purple',cex=0.75)
```



```
# Now perform a chi-square test to check how well the observations fit the
# Poisson distribution to test the null hypothesis
Badness_pois = sum((observations-expectations_pois)^2/expectations_pois)
Badness_pois
```

```
## [1] 177.7814
```

```
# The degrees of freedom is 11-1-1 = 9: there are 11 categories,
# minus one because the total is known, minus another one because the
# probability was estimated using data
p_pois = 1-pchisq(Badness_pois,df=9)
p_pois
```

```
## [1] 0
```

### The number of days I rested between runs: external factors

When comparing the expectations to the observations, I had noticed that there were unusually high observations for 3 of my running sessions, where I had rested for 8, 9 and 11 days.

I decided to investigate this further and managed to find historic rainfall data showing that it was raining during two of these sessions. During the third session, I had given a blood donation, so I would have rested for additional days before restarting my running training.

I decided to remove these unusually long rest days from the dataset and repeat this test to see if the data could still be modelled as a Poisson distribution, if these external factors were excluded.

As calculated below, the p value for the Poisson distribution test is $p = 0.8495059$, which is well above the standard threshold of 5%. This means that there is insufficient evidence and we fail to reject the null hypothesis. Therefore, it can be inferred that the numbers of days I rested between runs is consistent with a Poisson distribution, when external factors are excluded. Therefore it can be inferred that I maintained good consistency with the number of rest days between runs despite not following any formalised running schedule.

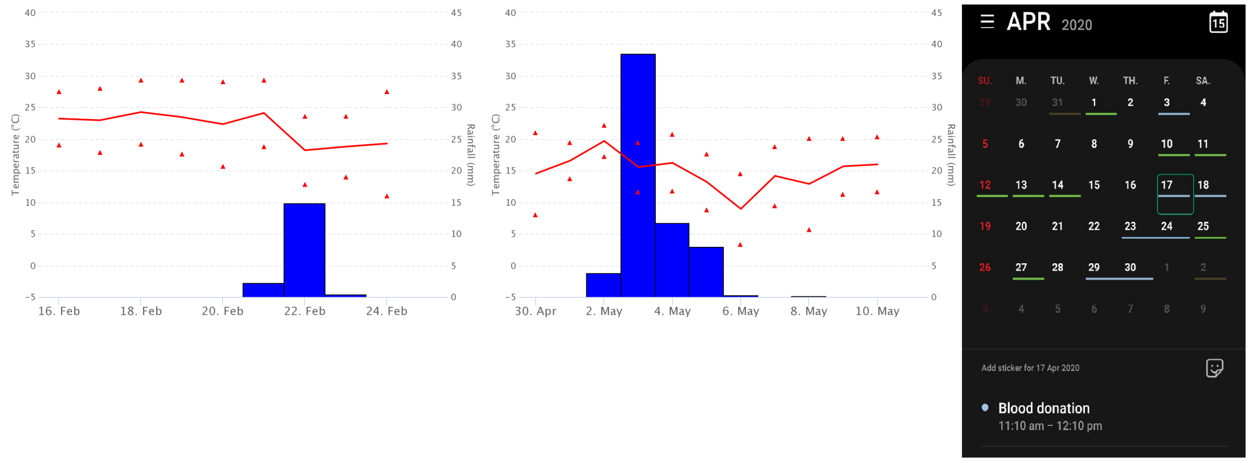
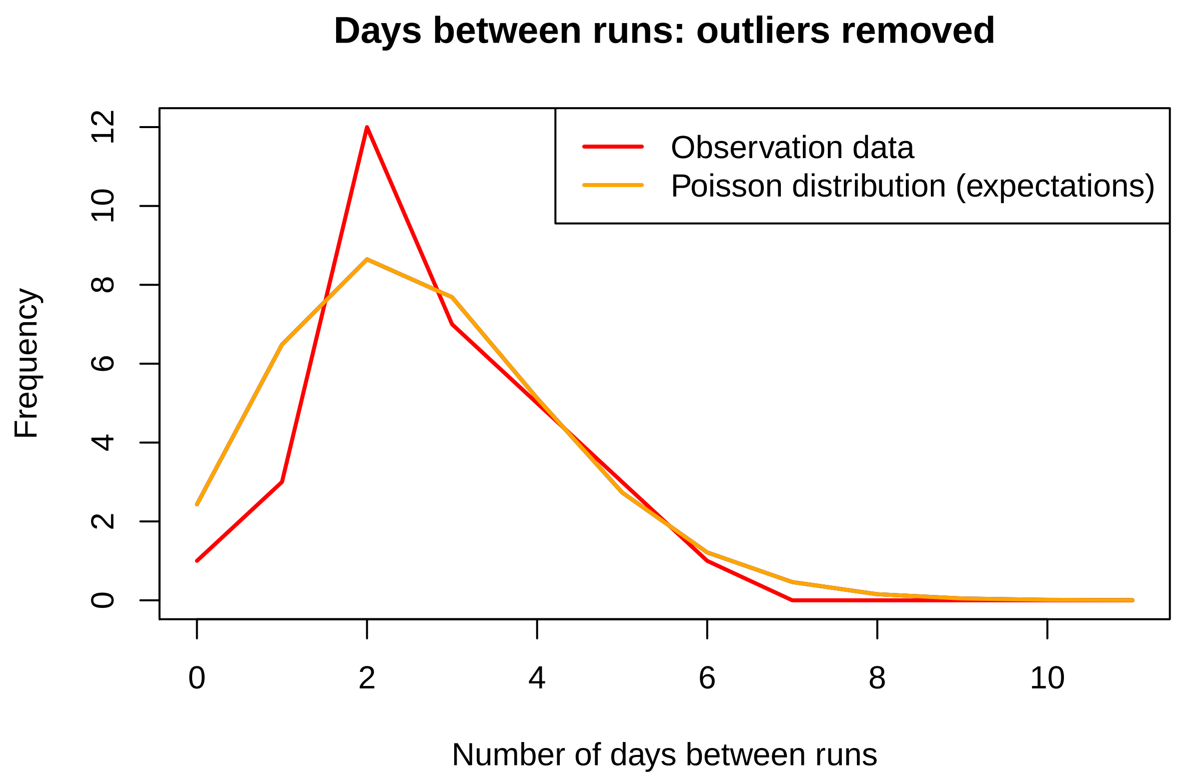


Figure 1: Reasons explaining long rests between runs

```
observations_raw = table(data$daysSinceLastRun)
# Fill in observations table with categories with zero frequency
observations_raw["7"] = 0
observations_raw["10"] = 0

observations=(c(observations_raw[1:7],
                observations_raw[11],observations_raw[8:9],
                observations_raw[12],observations_raw[10]))

# Removing unusually high rest day counts which were explained by external factors
observations[9]=0
observations[10]=0
observations[12]=0

minDays = min(data$daysSinceLastRun)
maxDays = max(data$daysSinceLastRun)

daysSinceLastRunRange = minDays:maxDays
nRows = length(data$daysSinceLastRun)
```

```
mean_daysSinceLastRun = mean(observations)

# Here is the expected values for days since last run
expectations_pois = dpois(daysSinceLastRunRange,lambda=mean_daysSinceLastRun)*nRows
plot(daysSinceLastRunRange,
     expectations_pois,
     type='l',ylim=c(0,12),col='blue',lwd=2,
     xlab="Number of days between runs", ylab="Frequency",
     main="Days between runs: outliers removed")

points(daysSinceLastRunRange,
       observations,
       type='l',col='red',lwd=2)


points(daysSinceLastRunRange,
       expectations_pois,
       type='l',col='orange',lwd=2)

legend("topright",lwd=2, col=c("red","orange"),bg="white",
       legend=c("Observation data","Poisson distribution (expectations)"))
```



```
# Now perform a chi-square test to check how well the observations fit the
# Poisson distribution to test the null hypothesis
Badness_pois = sum((observations-expectations_pois)^2/expectations_pois)
```

```
Badness_pois
```

```
## [1] 4.822414
```

```
p_pois = 1-pchisq(Badness_pois,df=9)
p_pois
```

```
## [1] 0.8495059
```

### Do I run further on weekends or weekdays?

As I am a casual runner and don't really pay attention to maintaining any sort of schedule, I was curious as to see whether or not I ran longer distances on weekends. I have more free time during weekends, so it would be reasonable to believe that I may have gone on longer runs during weekends. However, this running data was recorded when New Zealand was under a Level 4 Covid-19 lockdown, so it could be argued that I had more free time overall, and so I did not go on longer runs on weekends.

The null hypothesis proposed here is that my mean running distance is no greater on week days than it is on weekends. The T test is used to calculate if there is any statistical significance between my running distances during weekdays vs. weekends. This will be a one sided test, with the alternative hypothesis being that I run further on average during weekends.

The t test produces a p-value of $p = 0.04946$, which is only just smaller than the standard threshold of p=0.05 by a small margin. This means that the difference between the distances during weekdays and weekends is statistically significant. There is evidence to reject the null hypothesis and infer that I do go on longer runs on weekends.

```
# Split dataset into weekdays and weekends
data$weekday
```

```
##  [1] "Tuesday"   "Thursday"  "Saturday"  "Monday"    "Wednesday" "Friday"
##  [7] "Sunday"    "Friday"    "Sunday"    "Monday"    "Thursday"  "Sunday"
## [13] "Thursday"  "Sunday"    "Monday"    "Wednesday" "Thursday"  "Sunday"
## [19] "Tuesday"   "Sunday"    "Thursday"  "Monday"    "Tuesday"   "Saturday"
## [25] "Tuesday"   "Thursday"  "Sunday"    "Tuesday"   "Friday"    "Sunday"
## [31] "Saturday"  "Monday"    "Friday"    "Wednesday" "Sunday"
```

```
weekdays = filter(data,weekdayInt<6)["distKm"]
weekends = filter(data,weekdayInt>=6)["distKm"]

weekdays_dist_mean = mean(weekdays$distKm)
weekends_dist_mean = mean(weekends$distKm)
weekdays_dist_mean
```

```
## [1] 5.418261
```

```
weekends_dist_mean
```

```
## [1] 8.164167
```

```r
t.test(weekdays,weekends,alternative="less")
```

```
##
##  Welch Two Sample t-test
##
## data:  weekdays and weekends
## t = -1.7684, df = 13.911, p-value = 0.04946
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##          -Inf -0.009713793
## sample estimates:
## mean of x mean of y
##  5.418261  8.164167
```

### Am I more likely to increase my distance if I run in the morning?

When I go for a run, I aim to increase my distance from my last run so that I can reach my distance goal of a half marathon. At times, I may feel low on energy or my muscles feel worn, so I will go for a shorter run instead.

I tend to find that I have more energy in the morning, before midday. I enjoy running in the cooler weather, I am more relaxed in the morning and I generally feel like I run better before having heavy meals.

I would like to investigate if running before noon makes a difference as to whether or not I end up running a shorter distance. The time of run start will be divided into two categories: before midday or after midday (includes afternoon and evening runs). The delta distance run shows the percentage increase or decrease of run distance compared to the previous run. Runs with greater than 0% delta distance are categorised as 'longer' runs, and runs with 0% or less delta distance are categorised as 'shorter' runs for this test.

The null hypothesis proposed here is that the probability that I increase my run distance is not impacted by whether or not I start running before noon. The Fisher's test is used here to test this hypothesis. The alternative hypothesis is that if I start running before midday I am more likely to increase my run distance from my last run.

The p-value is $p = 0.2099$, which is higher than the standard threshold of 5%. This means that the result is not statistically significant. Thus, there is insufficient evidence and we fail to reject the null hypothesis. We can infer that when I start running before midday I am no more likely to increase my distance from my previous run than if if I start running later in the day.

```r
data["startBeforeNoon"] = data$time24hour<120000
table(data$startBeforeNoon)
```

```
##
## FALSE  TRUE
##    16    19
```

```r
data["deltaDistIncrease"] = data$deltaDistPastRun>0
table(data$deltaDistIncrease)
```

```
##
## FALSE  TRUE
##    16    19
```

```
contingency <- table(data$startBeforeNoon, data$deltaDistIncrease)
dimnames(contingency) <- list(startTime=c("after midday","before midday"),
                              deltaDist=c("Decrease","Increase"))
contingency
```

```
##                deltaDist
## startTime       Decrease Increase
##   after midday         9        7
##   before midday        7       12
```

```
fisher.test(contingency,alternative='greater')
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  contingency
## p-value = 0.2099
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  0.5760274       Inf
## sample estimates:
## odds ratio
##   2.153494
```

### Designing a good training schedule

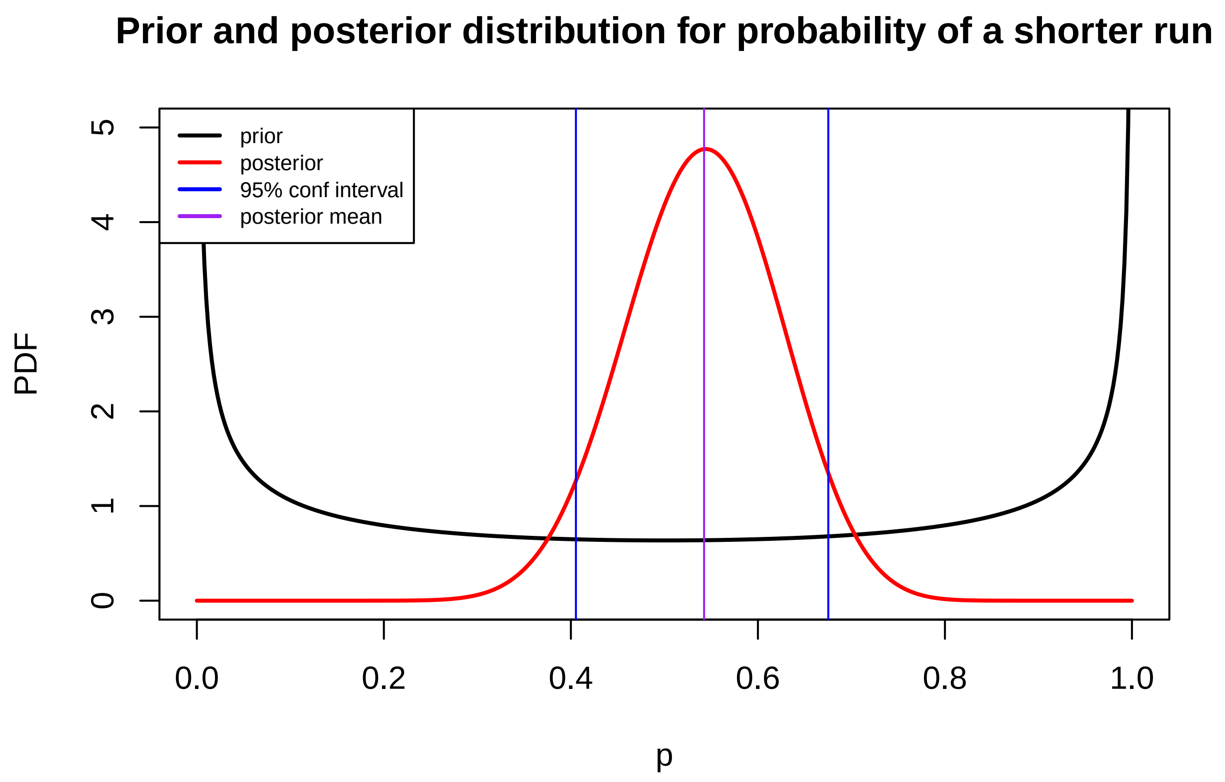
Typical long distance running training schedules advise that you try to improve your distance each time, but also mix it up with some shorter, easier sessions. One popular running guide suggests that beginners should run shorter sessions 50% of the time and longer sessions 50% of the time.

My proportion of long and short runs can be analysed using a Baysian approach. The prior probabilities are $\alpha = 0.5$ and $\beta = 0.5$, representing the number of long runs and short runs expected from a typical beginner's running training program. The posterior will be represented by the delta distance between runs. If the delta distance from the previous run is greater than zero, this will be interpreted as a longer run. Delta distances smaller than zero will be interpreted as a shorter run.

Using the calculations below, the posterior probability lies somewhere between 0.41 and 0.68 with a 95% confidence interval. So this infers that a good beginner's running training schedule that suits me would be one where I run shorter sessions somewhere between 41% and 68% of the time. The posterior mean shows that I am most likely running shorter sessions 54.24467% of the time.

```
alpha=1/2
beta=1/2
post_alpha = unname(table(data$deltaDistPastRun>0)[2])
post_beta = unname(table(data$deltaDistPastRun>0)[1])
p <-seq(from=0,to=1,len=500)# probability on horizontal axis
plot(p,p*0,ylim=c(0,5),ylab="PDF",type="n",
     main="Prior and posterior distribution for probability of a shorter run")
points(p,dbeta(p,alpha,beta),lwd=2,type="l",col='black')# prior
points(p,dbeta(p,alpha+post_alpha, beta+post_beta),lwd=2,type="l",col='red')# posterior

post_conf_lower = qbeta(0.05,alpha+post_alpha, beta+post_beta)
```

```
post_conf_upper = qbeta(0.95,alpha+post_alpha, beta+post_beta)
post_conf_mid = qbeta(0.5,alpha+post_alpha, beta+post_beta)
abline(v=c(post_conf_lower,post_conf_upper),col="blue")
abline(v=c(post_conf_mid),col="purple")

legend("topleft",lwd=2,col=c("black","red","blue","purple"),cex=0.7,bg="white",
       legend=c("prior","posterior","95% conf interval","posterior mean"))
```



**Running in overcast weather**

My favourite weather to run in is overcast with full cloud cover. Bayes's Theorem can be used here to calculate the probability that it is overcast weather, given the fact that I have been running on a particular day.

Overcast weather is a fairly rare occurrence - out of all my running sessions, I managed to run on an overcast day 11.4% of the time. 88.5% of the time the cloud cover was partial or non existent.

My running data spans 117 days between, 14 Jan to 10 May, it has been overcast weather 20 times during that time. Using Bayes's Theorem, there is a 2.59% probability that it was overcast on a particular day, given that I had been running. Overcast days are a rare occurrence and also I do not go on a run every single day, so it looks like I was unable to run on every overcast day, hence why the probability is so low.

```
P_overcast = 20/117
P_not_overcast = 1-20/117
P_running_when_overcast = sum(data$weather == "Cloudy")/length(data$weather)
```

```
P_running_when_not_overcast = sum(data$weather != "Cloudy")/length(data$weather)
P_overcast_if_I_ran = (P_overcast*P_running_when_overcast)/
        (P_overcast*P_running_when_overcast+P_not_overcast*P_running_when_not_overcast)
P_overcast_if_I_ran
```

```
## [1] 0.02591513
```

### Predicting my average pace

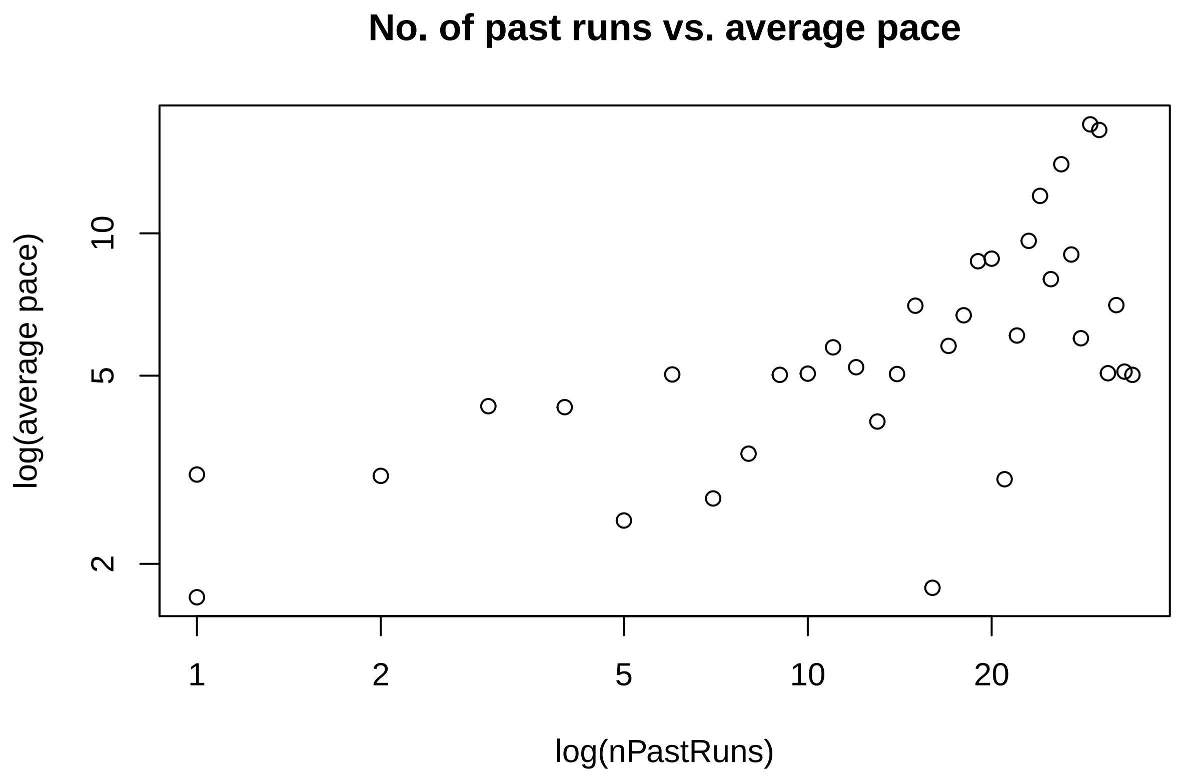
I would like to try to predict my average pace. A linear regression model is built which predicts average pace using the number of past runs and the total ascent in kilometers.

As calculated below, the formula for the linear regression model is:

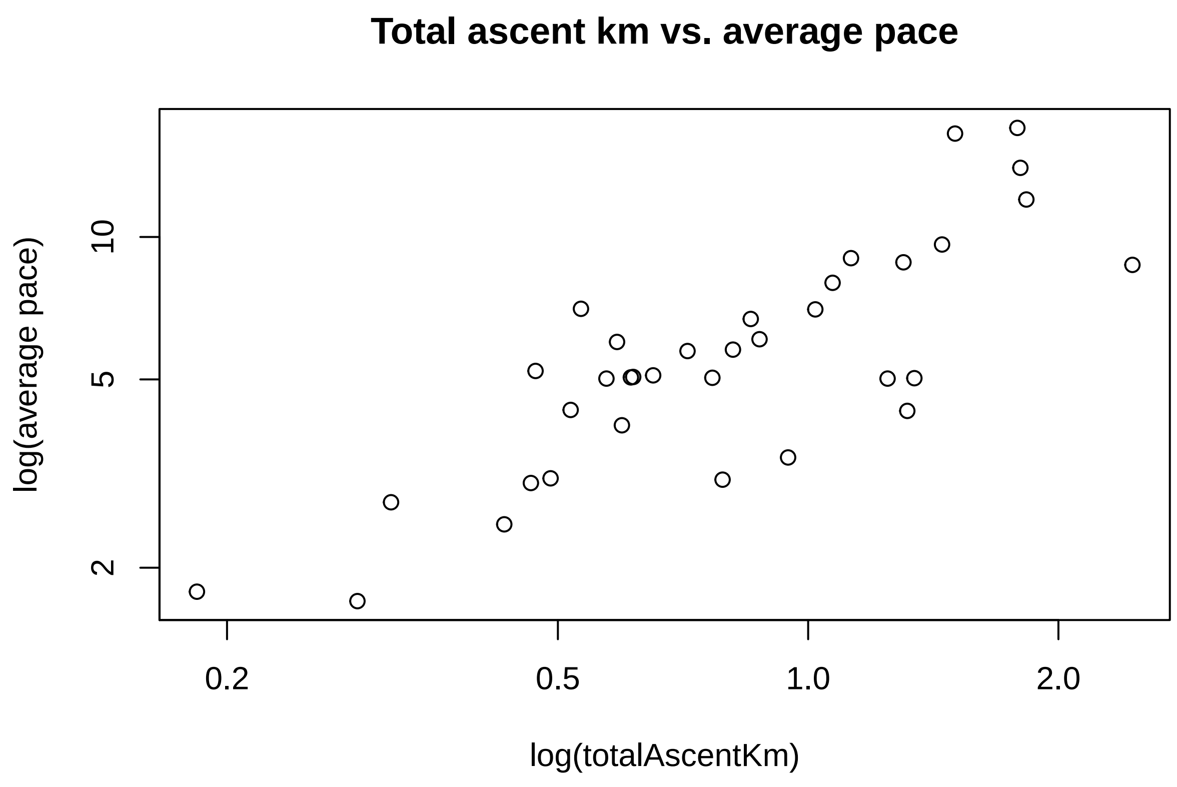
$Y_{distance} = 0.85434 + 0.37378X_{nPastRuns} + 0.01490X_{ascent} + 0.24236X_{nPastRuns}X_{ascent} + \epsilon$

where $\epsilon$ is some random error. This model has an adjusted $R^2 = 0.8516$, which shows that this model is a good fit for the observed data and is able to predict average pace reasonably well. First I will plot the variables to show the relationships for average pace vs. n Past runs and average pace vs. total ascent km. Then, I will calculate the linear regression model which includes an interaction term and then visualise this as a 3D plot.

```
data$nPastRuns[1]=1 # Change a value of 0 to 1 so that the log function can be applied
plot(distKm~nPastRuns,data=data, log='xy',
     ylab="log(average pace)",
     xlab="log(nPastRuns)",
     main="No. of past runs vs. average pace")
```



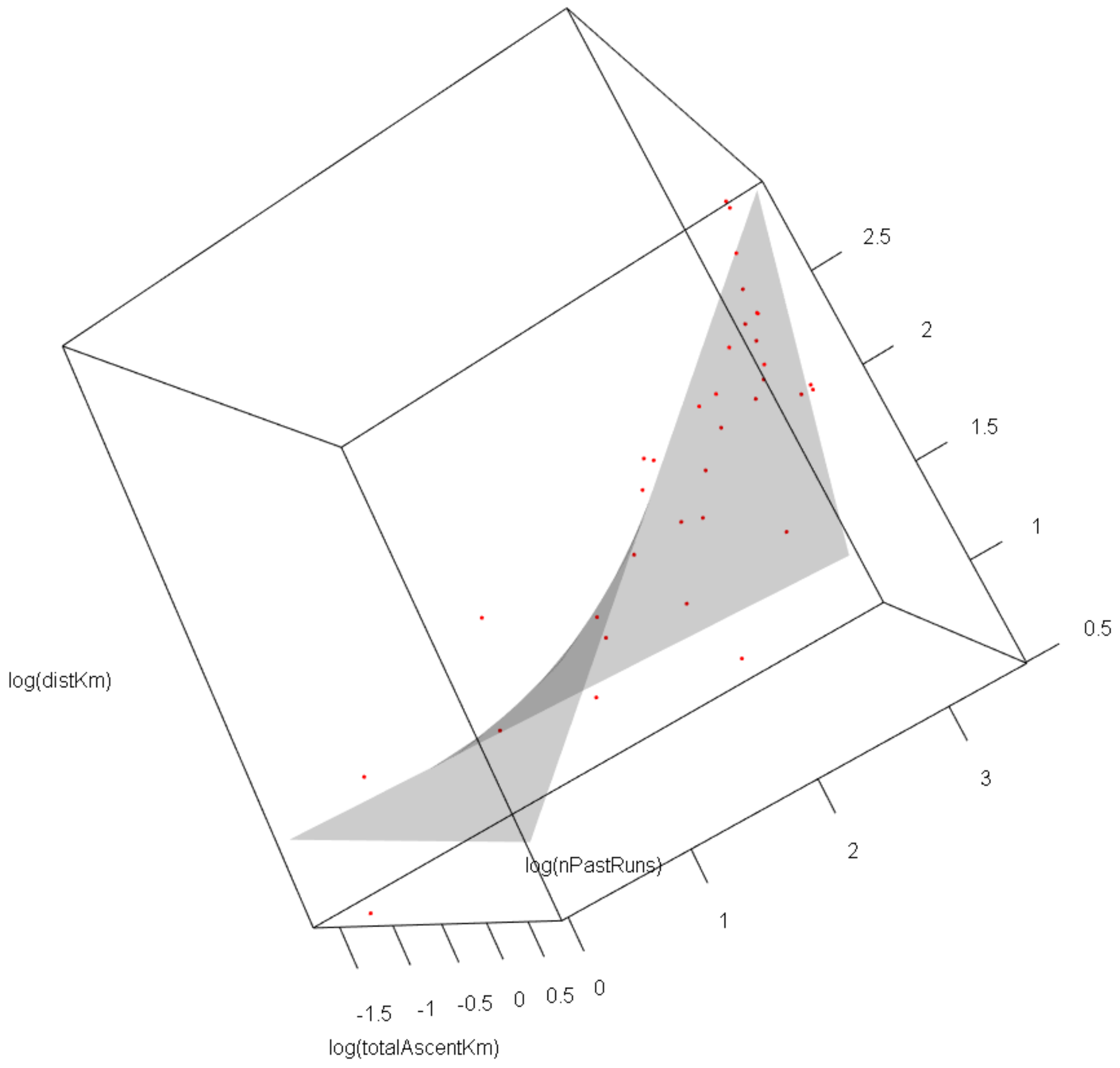
```
plot(distKm~totalAscentKm,data=data, log='xy',
     ylab="log(average pace)",
     xlab="log(totalAscentKm)",
     main="Total ascent km vs. average pace")
```



```
summary(lm(log(distKm)~log(nPastRuns)+log(totalAscentKm),data=data))
```

```
##
## Call:
## lm(formula = log(distKm) ~ log(nPastRuns) + log(totalAscentKm),
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47513 -0.15564  0.02757  0.12526  0.47964
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.23688    0.13237   9.344 1.16e-10 ***
## log(nPastRuns)      0.24239    0.04678   5.181 1.17e-05 ***
## log(totalAscentKm)  0.65105    0.07732   8.420 1.28e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2448 on 32 degrees of freedom
## Multiple R-squared:  0.819,  Adjusted R-squared:  0.8077
## F-statistic: 72.41 on 2 and 32 DF,  p-value: 1.323e-12
```

Here is the plot of the linear regression model that was created. This is a 3D plot because there are 3 variables to plot (total ascent, total distance and n Past runs). This 3D visualisation is interactive if run as an R markdown file in RStudio (provided that WebGL support is availble in the browser and that RGL is available).



The code that was used to create this 3D visualisation is shown below. It will run in RStudio as an Rmd file if the latest versions of webgl and rgl are available.

```
my_surface <- function(f, n=10, ...) {
  ranges <- rgl:::.getRanges()
  x <- seq(ranges$xlim[1], ranges$xlim[2], length=n)
  y <- seq(ranges$ylim[1], ranges$ylim[2], length=n)
  z <- outer(x,y,f)
  surface3d(x, y, z, ...)
}
library(rgl)
f <- function (nPastRuns, ascent)
  0.85434 + 0.37378 * nPastRuns + 0.01490 * ascent + 0.24236*nPastRuns*ascent


n <- 200
nPastRuns <- log(data$nPastRuns)
totalAscentKm <- log(data$totalAscentKm)
distKm <- log(data$distKm)

plot3d(nPastRuns,totalAscentKm,distKm, type="p", col="red", xlab="log(nPastRuns)",
ylab="log(totalAscentKm)", zlab="log(distKm)", site=5, lwd=15)
my_surface(f, alpha=.2 )
```