Informatica - Area scientifica
Dipartimento di Scienze matematiche, informatiche e multimediali
Università di Udine

Tesina Statistica Applicata

trackeR: Infrastructure for sport

Galvan Matteo (142985)
Giabardo Diletta (144176)

Anno accademico 2021/2022

Indice

Introduzione **3**

1 Package **5**
1.1 Definizione 5
1.2 Esempi 5

2 trackeR **6**
2.1 Spiegazione 6
2.2 Funzioni principali 6

3 Funzionalità **8**
3.1 Lettura 8
3.2 Sport supportati 9
3.3 Elaborazione dei dati 9
3.4 Analisi dei dati 9
3.5 Visualizzazione dei dati 10
3.6 Installazione 10
3.7 Caricamento 10

4 Classe trackeRdata **11**
4.1 Costruttore trackeRdata() 11
4.1.1 Threshold 11
4.1.2 Correzione della distanza 12
4.1.3 Unità di misura 12
4.1.4 Processo di imputazione 12
4.2 Runs 13

5 Analisi dataset **15**
5.1 Morning_Run.tcx 15
5.1.1 Summary 17
5.1.2 Indici generali 18
5.2 Grafici 20
5.2.1 Creazione grafico plot 20
5.2.2 Creazione mappa 21
5.2.3 Zones 22

5.2.4 Relazione Velocità-Cadenza 23
5.2.5 Profilo Distribuzione 24

6 Conclusione **26**

7 Sitografia **27**

Introduzione

R, implementazione di S, è un linguaggio di programmazione, orientato agli oggetti, con un ambiente software volto al calcolo statistico e alla grafica.
E' stato creato da Ross Ihaka e Robert Gentleman dell'Università di Auckland, in Nuova Zelanda, ed è stato introdotto nel 1993.
E' un linguaggio utilizzato per manipolazione e visualizzazione dei dati, calcoli statistici e generazione di grafici, inclusi modelli lineari e non lineari, test statistici classici, analisi di serie temporali, classificazione e clustering. Inoltre è completamente gratuito ed open source.
Nello specifico le sue caratteristiche sono:

- semplicità nella gestione e manipolazione dei dati;
- disponibilità di strumenti per calcoli su vettori, matrici ed altre operazioni complesse;
- accesso ad un insieme di strumenti integrati per l'analisi statistica;
- produzione grafiche particolarmente flessibili;
- possibilità di adoperare un vero e proprio linguaggio di programmazione orientato ad oggetti che consente l'uso di strutture condizionali e cicliche, nonché di funzioni create dall'utente.

R è multi piattaforma, difatti funziona su UNIX, Windows e MacOS. Utilizza l'interfaccia a riga di comando, sebbene siano disponibili diverse GUI. Per le attività ad alta intensità di calcolo, R consente di collegare codice scritto in C, C ++ e Fortran per essere chiamato in fase di esecuzione. Gli oggetti R possono anche essere manipolati direttamente con codice C.
Sono disponibili diverse interfacce grafiche che consentono di integrare R con diversi pacchetti tra cui Emacs tramite Emacs-mode ESS (Emacs Speaks Statistics). Una rassegna delle GUI disponibili può essere trovata sul sito R GUI projects. Una delle GUI più famose ed utilizzate risulta essere RStudio.

I punti di forza di R sono:

- la possibilità di accedere al codice sorgente e modificarlo;
- il costo zero per l'utente, essendo open source;
- amplia documentazione reperibile online;
- la disponibilità di librerie molto dettagliate per analisi statistiche, messe a disposizione da parte di ricercatori;
- l'enorme versatilità: permette di creare strumenti personali di analisi statistica, di utilizzare strumenti sofisticati, notevoli possibilità grafiche, molto rapido, multipiattaforma;
- una comunity altamente specializzata.

Esiste un'importante differenza tra R e la maggior parte degli altri packages statistici. Con gli altri software un'analisi statistica porta ad una grande quantità di output di informazioni e dati, mentre con R l'analisi statistica è condotta come una serie di passi, con risultati intermedi memorizzati in oggetti. Ad ogni passo dell'analisi gli output sono minimi e l'utente ha la possibilità di visualizzarli e manipolarli richiamando, quando è necessario, gli oggetti nei quali i risultati dell'analisi sono contenuti.

1 Package

1.1 Definizione

Il linguaggio R oltre alle librerie e pacchetti interni, permette di integrarne di esterni. I pacchetti R sono dei file esterni rispetto al linguaggio di programmazione R. Sostanzialmente i pacchetti contengono codice, dati e documentazione in un formato standard. I pacchetti possono essere installati da ogni utente che utilizza R e possono essere trovati tramite repository. La grande disponibilità di pacchetti, oltre che la grande documentazione di questi, ha portato R ad essere sempre più utilizzato. Rispetto alle librerie presenti in altri linguaggi di programmazione, i pacchetti di R devono essere conformi a delle specifiche molto ristrette. Seguendo le specifiche per creare pacchetti in R, viene garantita la possibilità di installare il pacchetto tramite il gestore dei pacchetti.

1.2 Esempi

I package maggiormente utilizzati sono:

- ggplot2: questi ultimi possono avere una singola variabile oppure multiple, possono averne di tipo numerico o categoriale. Si possono effettuare raggruppamenti tramite simboli, colori e altre caratteristiche.
- dplyr: questo pacchetto viene utilizzato per manipolare i dati tramite differenti chiamate a funzioni che permettono di renderli leggibili e utilizzabili da R.
- tidyr: questo pacchetto permette di avere dati ordinati e puliti. Sostanzialmente permette di avere dei dati in cui le colonne rappresentano le variabili e le righe le osservazioni.
- shiny: viene utilizzata per costruire delle applicazioni web senza la necessità di sviluppare il tutto in altri codici di programmazione. Permette di integrare CSS e altri linguaggi per costruire delle presentazioni interessanti.

Esistono altri migliaia di pacchetti installabili tramite il gestore dei package. Alcuni sono più famosi e più utilizzati, mentre altri lo sono meno.

2 trackeR

2.1 Spiegazione

Il proposito di questo pacchetto è quello di provvedere all'infrastruttura per gestire dati riguardanti corsa, nuoto e ciclismo provenienti da un device GPS. I formati accettati sono: .tcx, Strava .gpx, .db3 e file .json.
Dopo l'estrazione e la manipolazione dei dati, questi vengono posizionati (stanziati) in oggetti basati sulla sessione e sensibili alle unità della classe trackeRdata. Le informazioni nell'oggetto risultante possono essere visualizzate o analizzate tramite i corrispondenti metodi.

2.2 Funzioni principali

Tramite il documento ufficiale, rilasciato dai creatori del pacchetto sulla pagina **cran**, di seguito vengono riportate le funzioni del pacchetto **trackeR** e la loro descrizione.

Funzione	Descrizione
readTCX()	Legge un file TCX
readGPX()	legge un file GPX
readDB3()	legge un file DB3
readJSON()	legge un file JSON
readContainer()	legge un file TCX/GPX/DB3/JSON
readDirectory()	legge tutti i file TCX/GPX/DB3/JSON in una cartella
trackeRdata()	costruisce un oggetto **trackeRdata**
c()	combina le sessioni
sort()	ordina le sessioni in base al tempo di inizio
unique()	estrae un'unica sessione
[ ]	ritorna il sottoinsieme di una sessione
plot()	costruisce il grafico di un profilo in una sua sessione
plotRoute()	costruisce il grafico di un percorso in una mappa statica
leafletRoute()	costruisce il grafico di un percorso in una mappa iterativa
threshold()	applica il minimo ed il massimo limite ad un range di dati
smoother()	applica i dati ad una funzione di riepilogo (media, ecc)

Funzione	Descrizione
getUnits()	ritorna l'unità di misura
changeUnits()	cambia l'unità di misura
nsessions()	ritorna il numero di sessioni
fortify()	converte un oggetto in un data frame per la costruzione di grafici
summary()	riepiloga le sessioni
print()	stampa il riassunto della sessioni
plot()	stampa il riassunto di vari parametri a seconda dell'input
timeline()	stampa il grafico del riassunto della sequenza temporale
zones()	tempo speso nelle zone
Wprime()	calcola W' balance e W' expended
session_times()	ritorna la data e l'orario di inizio e fine della sessione
session_duration()	ritorna la durata di ogni sessione
get_sport()	ritorna lo sport di ogni sessione dell'oggetto
concentrationProfile()	calcolare i profili di concentrazione
distributionProfile()	calcolare i profili di distribuzione
ridges()	grafici di cresta dei profili di concentrazione/distribuzione
profile2fd()	converte i profili ad una classe **fd**
funPCA()	analisi delle componenti principali funzionali

Descrizione dettagliata Alcune delle funzioni qui sopra riportate verranno descritte in maniera dettagliata nel momento dell'utilizzo su un opportuno dataset, nei capitoli successivi(capitolo 3- capitolo 5).

3 Funzionalità

A differenza di altri pacchetti precedentemente sviluppati, trackeR, permette di colmare la differenza tra la raccolta dei dati del gps e l'analisi di questi tramite R (vedi capitolo 4).
Vediamo di seguito le caratteristiche:

3.1 Lettura

Il package trackeR garantisce la possibilità di leggere i seguenti formati:

- .tcx (training centre XML);
- Strava .gpx(esportato da Strava, funziona anche nelle altre versioni);
- .db3 (utilizzati in dispositivi GPSports, sono utilizzabili in SQLite);
- file .json come file esterni.

L'utente che si allena ha la possibilità di caricare i dati dell'allenamento in maniera diretta, tramite delle specifiche funzioni fornite dalla libreria, che ritornano un dataset con una specifica struttura.
Per caricare i file utilizziamo una funzione in cui specifichiamo il percorso e il nome del file con l'estensione. Nell'esempio di seguito presentato la funzione per caricare un file .tcx:

```
1 dataset <- readTCX("filepath/file.tcx")
```

Esistono altre funzioni che permettono di leggere determinate tipologie di file descritte sopra.
Nello specifico ne esistono due generali:

- **readContainer()** : la funzione riceve in input un percorso ad una cartella e legge un file, con formato supportato, all'interno di questa. In output restituisce un oggetto di tipo trackeRdata;
- **readDirectory()** : la funzione riceve in input un percorso ad una cartella e legge tutti i file, con formato supportato, all'interno di questa. La funzione ritorno un oggetto di tipo trackeRdata ed estrae gli sport per ogni sessione.

Usando l'argomento **aggregate** in queste due funzioni è possibile dividere i dati in sessioni basate sulle timestamp, e non più sullo sport.
I dati nei vari formati vengono importati e salvati in un oggetto della classe **trackeRdata** che verrà analizzato nel capitolo successivo.

3.2 Sport supportati

Il package permette di tracciare, misurare, studiare tre attività sportive:
corsa, ***ciclismo***, ***nuoto***.
Ogni sport misura diversi dati. Tutti gli sport misurano la latitudine, la longitudine, l'altitudine, la distanza, il battito del cuore, la velocità, la temperatura, il ritmo e la durata dell'allenamento.
Il ciclismo misura, inoltre, la potenza e la cadenza della pedalata.
La corsa misura, inoltre, la cadenza della corsa.

3.3 Elaborazione dei dati

trackeR permette di identificare automaticamente le sessioni di allenamento tramite il time e il threshold e rende possibile l'aggiunta di informazioni relative a possibile pause o stop.
L'elaborazione dei dati permette di correggere i dati grezzi forniti dal dispositivo GPS oltre a "pulire" dati scorretti, per esempio distanze o velocità negative.
Dopo l'estrazione e la manipolazione dei dati, questi vengono posizionati (stanziati) in oggetti basati sulla sessione e sensibili alle unità della classe trackeRdata.

3.4 Analisi dei dati

Un punto fondamentale nell'analisi dei dati consiste nella creazione di un "riassunto" della sessione di allenamento che contiene un insieme di indici riferiti ai dati raccolti durante la stessa (tra cui durata, heart rate...).
Inoltre vengono analizzati il tempo speso in momenti specifici(ad una determinata velocità o ad un determinato battito cardiaco) e i momenti di lavoro critico.
trackeR fornisce un'implementazione per i profili di distribuzione tramite una serie di funzioni (descritto nel capitolo 2). I profili i distribuzione nascono dalla necessità di comparare le sessioni e usare i dati raccolti per fare delle comparazioni.

3.5 Visualizzazione dei dati

trackeR fornisce degli strumenti per la visualizzazione dei dati come:

- la sessione di allenamento nei suoi dati più specifici (velocità, battito del cuore, pressione e altro);
- le statistiche della sessione di allenamento in generale;
- il tempo trascorso in ogni singola zona di concentrazione e i profili di distribuzione.

3.6 Installazione

Per installare il pacchetto utilizzando la versione rilasciata su **CRAN** è necessario eseguire il comando:

```
1 install.packages("trackeR")
```

oppure tramite la versione rilasciata su GitHub:

```
1 # install.packages("devtools")
2 devtools::install\_github("trackerproject/trackeR")
```

3.7 Caricamento

Per utilizzare la libreria è necessario prima caricarla nell'ambiente di lavoro tramite il seguente comando:

```
1 library("trackeR")
```

4 Classe trackeRdata

Il cuore del pacchetto trackeR è la classe **trackeRData**.
L'oggetto **trackeRData** è basato su sessioni e su strutture oggettivamente organizzate, che permettono di trattare i dati come un elenco di oggetti **zoo** multivariati. Le singole osservazioni vengono salvate ognuna in una sessione separata e vengono ordinate in base al timestamp contenuto nel dato fornito dal dispositivo gps.
La funzione **trackeRData()** prende in input l'output della lettura di un file (tipo .tcx) tramite la funzione adatta alla tipologia di file (Vedi funzioni capitolo 2).

4.1 Costruttore trackeRdata()

Il costruttore della funzione **trackeRData()** permette, inoltre, di effettuare un trattamento dei dati, come la rimozione dei valori negativi o nulli.
Quando utilizziamo questa funzione l'unico parametro obbligatorio è il dataset.

```
1 trackeRdata(dat, units = NULL, sport = NULL,
2     session_threshold = 2, correct_distances = FALSE,
3     from_distances = TRUE, country = NULL, mask = TRUE,
4     lgap = 30, lskip = 5, m = 11, silent = FALSE)
```

Sono presenti poi dei parametri facoltativi, come l'unità di misura e la tipologia di sport, che permette di utilizzare o meno particolari funzioni, come la potenza.

4.1.1 Threshold

La funzione costruttore permette di raggruppare le osservazioni in sessioni in base al timestamp. Nello specifico il dato fornito dalla funzione di lettura **read*()**, viene prima ordinato in base al timestamp e successivamente inserito nella stessa sessione della misurazione precedente se il timestamp è minore del threshold. Di default è settato a 2 ore.

4.1.2 Correzione della distanza

Il parametro **correct distances** di default viene settato a FALSE. Nel caso in cui venga messo a TRUE, il parametro **distance**, che tratta la distanza complessiva, viene ricalibrato.
Se i dati registrati contengono una variazione di altitudine è necessario ricalibrare o correggere la distanza percorsa, poiché quando aumenta o diminuisce l'altitudine, percorriamo più strada. Per calcolare la distanza corretta da un punto t_{i-1} a un punto t_i utilizziamo il teorema di Pitagora:

$$d_i - d_{i-1} = \sqrt{(d_{2,i} - d_{2,i-1})^2 + (a_i - a_{i-1})^2}$$

con d_i e a_i distanze cumulative corrette e l'altitudine al tempo t_i. Se le misure dell'altitudine non sono disponibili, sono estratti in automatico da un database esterno. I parametri **country** e **mask** sono settati per l'estrazione dell'altitudine.

4.1.3 Unità di misura

Un'altra elaborazione consiste nella possibilità di cambiare l'unità di misura.
Gli oggetti della classe **trackeRdata** hanno come attributo anche l'unità di misura; il metodo **getUnits()** permette di visionarle. Tramite il metodo **changeUnits()** è reso possibile il cambio dell'unità di misura di una o più variabili.
Se oggetti con unità di misura differenti sono uniti viene tenuta l'unità di misura del primo oggetto.

```
1 runCU <- changeUnits(run, variable = "speed",
2     unit = "km_per_h", sport = "running")
```

Questo codice permette di cambiare la variabile velocità da m/s a km/h.

4.1.4 Processo di imputazione

I dispositivi di localizzazione abilitati per GPS in genere registrano dati spazio-temporali campionati in modo irregolare. Per questo motivo in trackeR viene adottato un approccio che tiene conto di questo caratteristica.
Può succedere che quando vengono raccolti i dati da un device gps, ci sia un elevata differenza di tempo tra due misurazioni successive. Questo accade quando l'atleta mette in pausa il device, oppure lo stesso device non rileva significanti cambiamenti di posizione, a livello di coordinate.
Di default viene assunto che nell'intervallo di tempo tra le due misurazioni non è stato eseguito alcun tipo di lavoro (non c'è velocità per la corsa e non c'è potenza per la bicicletta).
La funzione **trackeRdata()** aggiunge 10 misurazioni all'interno di questo intervallo che avranno come valore di velocità e/o potenza 0 e le coordinate uguali agli estremi, non compresi, di queste 10 misurazioni.

4.2 Runs

All'interno del package di trackeR è presente un oggetto trackeRdata chiamato "runs" che contiene una serie di dati simulati riguardo varie sessioni di corsa.

Caricamento del dataframe
Per caricare l'oggetto trackeRdata **runs** all'interno dell'ambiente di lavoro RStudio utilizziamo la funzione **data()**. Nella funzione come primo argomento scriviamo il dataframe che vogliamo caricare e come secondo argomento il pacchetto in cui esso è contenuto.

```
1 data(runs, package = "trackeR")
```

Contenuto del dataframe
Per visionare la tipologia di variabili dalle quali è formato l'oggetto **runs**, scriviamo il nome dell'oggetto e ci ritorna le caratteristiche e le unità di misura delle variabili.

```
1 runs
```

Il risultato del comando sono le statistiche degli allenamenti, l'inizio e fine, il numero di sessioni di allenamento e la durata totale delle sessioni.

```
A trackeRdata object
Sports: running

Training coverage: from 2013-06-01 19:32:15 to 2013-06-30
    09:48:02
Number of sessions: 27     Training duration: 22.74 h

Units
 latitude         degree         cycling
 longitude        degree         cycling
 altitude         m              cycling
 distance         m              cycling
 heart_rate       bpm            cycling
 speed            m_per_s        cycling
 cadence_cycling  rev_per_min    cycling
 power            W              cycling
 temperature      C              cycling
```

```
pace             min_per_km     cycling
duration         min            cycling
latitude         degree         running
longitude        degree         running
altitude         m              running
distance         m              running
heart_rate       bpm            running
speed            m_per_s        running
cadence_running  steps_per_min  running
temperature      C              running
pace             min_per_km     running
duration         min            running
latitude         degree         swimming
longitude        degree         swimming
altitude         m              swimming
distance         m              swimming
heart_rate       bpm            swimming
speed            m_per_s        swimming
temperature      C              swimming
pace             min_per_km     swimming
duration         min            swimming
```

Oltre alle statistiche degli allenamenti, per ogni sport vengono descritte le variabili e le unità di misura di queste.

5 Analisi dataset

Il dataset è un insieme di dati raggruppato in una struttura in cui ogni colonna rappresenta una variabile di interesse e le righe rappresentano le osservazioni di queste variabili. I dataset vengono creati per raccogliere dati da un esperimento o da un evento.

Runs
Come descritto nel paragrafo 4.2, è presente all'interno del pacchetto trackeR un oggetto **trackeRdata** chiamato runs.
Questo oggetto, il dataset fornito dalla libreria trackeR, è stato analizzato in maniera esaustiva all'interno del documento pubblicato da Hannah Frick dell'università di Londra(sitografia 7). Quindi è stato deciso di fare un'analisi di un dataframe esterno.

5.1 Morning_Run.tcx

Morning_Run.tcx è un dataset esterno fornitoci da Alberto Donadel, studente dell'Università di Udine. I dati sono stati raccolti tramite un orologio durante una sessione mattutina di corsa svoltasi il 24/06/2019, e la loro elaborazione primaria è stata effettuata dall'applicazione **Strava**. Il file .tcx contenente i dati è stato in seguito scaricato direttamente dall'applicazione.

Caricamento del dataframe

Per caricare **Morning_Run.tcx**, all'interno di R, è necessario eseguire il seguente codice:

```
1 morningRun <- readTCX("filepath/Morning_Run.tcx")
```

Per visionare la tipologia di dati contenuti all'interno del dataframe utilizziamo la funzione:

```
1 str(morningRun)
```

E otteniamo i seguenti dati:

```
'data.frame':   605 obs. of  11 variables:
 $ time           : POSIXct, format: "2019-06-24 08:52:41"
                    "2019-06-24 08:52:42" "2019-06-24 08:52:47" ...
 $ latitude       : num  45.9 45.9 45.9 45.9 45.9 ...
 $ longitude      : num  12.2 12.2 12.2 12.2 12.2 ...
 $ altitude       : num  162 162 163 163 163 ...
 $ distance       : num  0.3 1.2 1.2 5 23.2 26.9 34.1 41.5 49.4 61.8 ...
 $ heart_rate     : num  102 103 100 101 104 104 110 112 116 120 ...
 $ speed          : num  0 0.9 0.2 0.6 3.7 3.7 3.6 3.7 3.8 4.1 ...
 $ cadence_running: num  49 0 80 82 84 84 86 88 88 84 ...
 $ cadence_cycling: logi  NA NA NA NA NA NA ...
 $ power          : logi  NA NA NA NA NA NA ...
 $ temperature    : logi  NA NA NA NA NA NA ...
 - attr(*, "sport")= chr "running"
 - attr(*, "file")= chr "C:/Users/asus/Desktop/Morning_Run.tcx"
```

Nel dataframe sono presenti delle colonne per dati riferiti a sessioni di corsa e visionando il contenuto sono risultati assenti i dati correlati alla temperatura e quindi non verrà considerata all'interno della successiva analisi. I parametri cadence_cycling, power e termperature contengono dati NA (Not Avaiable) poichè non vengono misurati all'interno della corsa. Di conseguenza anche nelle analisi successive risulteranno non disponibili.

Oggetto trackeRdata

Per rendere il dataframe **Morning_Run.tcx** un oggetto trackeRdata utilizziamo questo comando, descritto nella sezione 4.1:

```
1 morningRunTD <- trackeRdata(morningRun)
```

Quando chiamiamo l'oggetto **morningRunTD** otteniamo lo sport, il giorno e l'ora della sessione, con le variabili che contiene, come è stato visto nella sezione 4.2)

5.1.1 Summary

Le statistiche riassuntive di ogni allenamento del dataframe possono essere richiamate tramite la funzione **summary()**. Siccome è un riassunto della sessione, stima la distanza totale coperta, il tempo che si è speso muovendosi e altri indici come media della velocità, ritmo medio, cadenza media, potenza media.
Utilizziamo il comando:

```
1 summary(morningRunTD)
```

Quello che otteniamo è un oggetto trackeRdataSummary:

```
*** Session 1 : running ***

 Session times: 2019-06-24 10:52:36 - 2019-06-24 11:28:42
 Distance: 6768.27 m
 Duration: 36.1 mins
 Moving time: 28.82 mins
 Average speed: 3.12 m_per_s
 Average speed moving: 3.91 m_per_s
 Average pace (per 1 km): 5:20 min:sec
 Average pace moving (per 1 km): 4:15 min:sec
 Average cadence running: 82.17 steps_per_min
 Average cadence cycling: NA rev_per_min
 Average cadence running moving: 82.75 steps_per_min
 Average cadence cycling moving: NA rev_per_min
 Average power: NA W
 Average power moving: NA W
 Average heart rate: 165.57 bpm
 Average heart rate moving: 165.1 bpm
 Average heart rate resting: 177.86 bpm
 Average temperature: NA C
 Total elevation gain: 76.94 m
 Work to rest ratio: 3.96

 Moving thresholds: 2.0 (cycling) 1.0 (running) 0.5
                    (swimming) m_per_s
 Unit reference sport: running
```

Questa funzione ritorna un riassunto preliminare differente per ogni sessione contenuta nel file .tcx caricato. In questo caso noi abbiamo una unica sessione di corsa. summary() restituisce un oggetto di tipo **trackeRdataSummary**.
Osservando il risultato di questo specifico summary possiamo osservare che vengono restituiti diversi valori. Viene definita la tipologia di allenamento e l'intervallo di tempo in cui è stato effettuato.

5.1.2 Indici generali

Di seguito andiamo a vedere in maniera più specifica i vari parametri e come vengono calcolati all'interno del dataset:

Distance
Il parametro contiene la distanza totale percorsa durante l'allenamento, misurata in metri.
Questa viene calcolata effettuando la sommatoria di tutte le distanze, oppure, semplicemente, moltiplicando la durata dell'allenamento per la velocità media. Tenendo conto anche della variazione dell'altitudine, visto che un percorso diagonale è più lungo di uno orizzontale. (vedi capitolo 4.1.2)

Duration
Il parametro contiene la durata totale dell'allenamento, dall'inizio della sessione fino allo stop finale. Viene misurata in minuti.

Moving Time
Il parametro contiene la misura del tempo trascorso in movimento, misurata in minuti.
Per la misura viene controllata la posizione dell'utente, se è uguale alla posizione precedentemente trovata allora l'utente non si sta muovendo e il **Moving Time** non verrà aumentato. Al contrario se la posizione è cambiata vorrà dire che l'utente si è spostato e di conseguenza verrà aumentato il tempo di spostamento.

Average Speed
Sono presenti due variabili per quanto riguarda Speed:

- Average speed: tratta la velocità media ottenuta dividendo la **Distance** per la **Duration**.
- Average speed moving: è calcolata come la **Distance** diviso il **Moving Time**.

Average Pace
Sono presenti due variabili per quanto riguarda Pace (il ritmo).

- Average pace (per 1 km): è calcolato come l'inversa di **Average speed**.
- Average pace moving (per 1 km): è calcolato come l'inversa di **Average speed moving**.

Average cadence/heart rate
Sono presenti due variabili per quanto riguarda Cadence (cadenza).

- Average cadence running: è un dato che rappresenta il numero di passi al minuto considerando tutta **Distance** e **Duration**.
- Average cadence running moving: è un dato che rappresenta il numero di passi al minuto considerando tutta **Distance** e **Duration** (senza il tempo delle pause).

Sono presenti tre variabili per quanto riguarda Heart Rate (battito cardiaco).

- Average heart rate: è un dato che rappresenta il numero medio di battiti sulla variabile **Duration**.
- Average heart rate moving: è un dato che rappresenta il numero medio di battiti sulla variabile **Duration** (senza il tempo delle pause).
- Average heart rate resting: è un dato che rappresenta il numero medio di battiti sulla variabile **Duration** considerando solamente le pause.

La media per Cadence ed Heart Rate sono ponderate con dei pesi che dipendono dalla differenza di tempo che esiste tra l'osservazione e la successiva. Infatti queste medie tengono conto anche della mancanza di osservazioni.

Total Elevation gain
Questo parametro rappresenta la somma della variazione di altitudine quando questa aumenta.

Work to rest Ratio
Il parametro contiene il rapporto lavoro/riposo.
Viene calcolato tramite formula matematica: $\frac{MovingTime}{Duration - MovingTime}$

Moving thresholds
Contiene la massima finestra di tempo, misurata in ore, che consente di definire due allenamenti differenti prese due misure adiacenti.

5.2 Grafici

5.2.1 Creazione grafico plot

Per la creazione di un primo grafico, utilizziamo il comando **plot**, che permette di creare un grafico. La funzione prevede come parametri:

- l'oggetto trackeRdata;
- le sessioni che verranno visualizzate e che qui non sono presenti perché il file ha solo una sessione;
- le variabili da visualizzare.

Nell'esempio viene utilizzato l'oggetto trackeRdata morningRunTD e alcune variabili di interesse:

```
1 plot(morningRunTD, what = c("speed", "pace",
2     "altitude","heart_rate"))
```

Questo comando permette di creare un grafico che rappresenta il riepilogo dei dati in particolare visualizzando i parametri di altitudine (m), battito cardiaco (bpm), di velocità (m/s) e di ritmo (min/km).

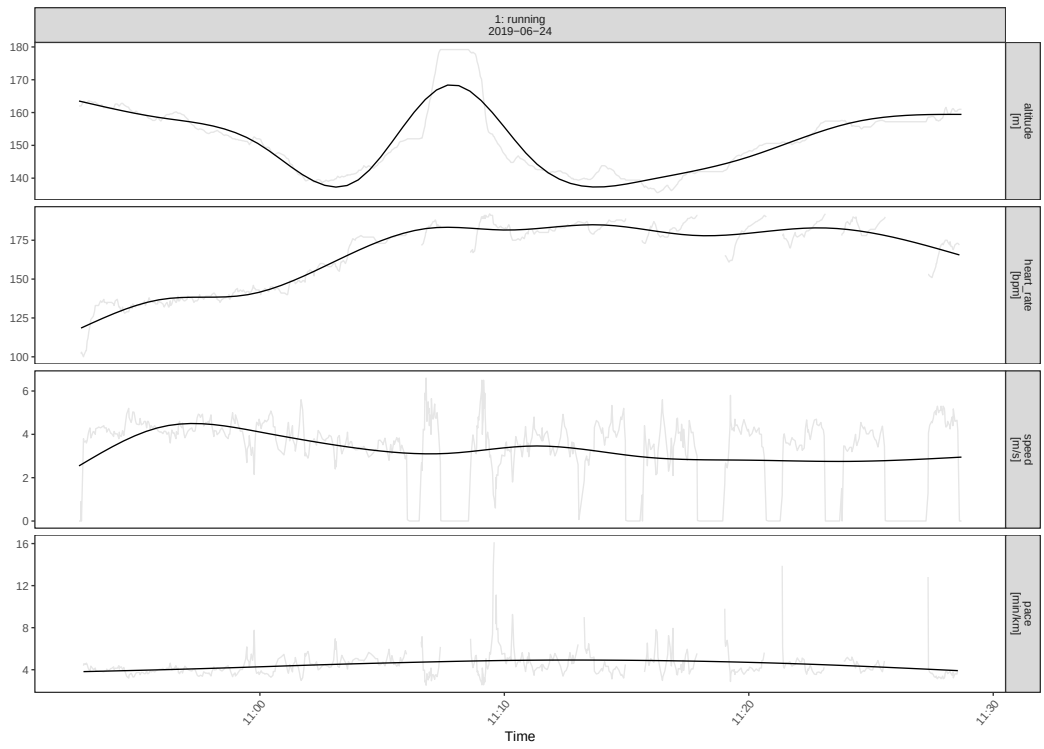


Figura 5.1: Plot workout data

I quattro grafici disegnati tramite **plot()** sono formati da due linee una più chiara e l'altra più scura. La linea leggera in trasparenza è formata da tutte le misurazioni riferite a quel dato. La linea più scura e in risalto rappresenta invece l'andamento più dolce dei dati, in modo da comprendere in maniera più semplice come si muovono.
Osservando i dati e il grafico per quanto riguarda le 4 variabili, notiamo che le prime misurazioni non vengono inserite all'interno del grafici perchè abbiamo, in alcuni casi, dei valori NA.
Si vede in modo chiaro che la sessione di corsa è stata svolta in un territorio con un dislivello elevato, come si denota anche dai dati (total elevation gain).
Per quanto riguarda il grafico del battito cardiaco c'è una certa costanza dovuta all'intensità della corsa. Osservando il grafico riferito alla velocità media e del ritmo, c'è una forte variabilità che dipende anche molto dalle pause e dall'aumento e diminuzione della altitudine.

5.2.2 Creazione mappa

Il comando **leaflet_route()** permette di disegnare il percorso in una mappa.

```
1  leaflet_route(morningRunTD)
```

Questo comando permette di visualizzare il percorso di allenamento, tracciato tramite longitudine e latitudine, su una mappa.
Come parametro bisogna inserire l'oggetto trackeRdata contenente tutte le informazioni necessarie per disegnare la mappa interattiva. Questa presenta il percorso colorato di blu.

5.2.3 Zones

Un buon modo per studiare i dati di una sessione viene fornito dal pacchetto e permette di calcolare quanto tempo è stato speso dall'atleta in determinate **zone**. La grandezza delle zone può essere creata manualmente per studiare l'andamento dell'allenamento. Possono essere applicate a tutte le variabili, come velocità o battito cardiaco.
Per visionare un esempio di zone nell'ambito dell'altitudine si utilizza:

```
1 zones_alt <- zones(morningRunTD, what = "altitude",
2     breaks = c(135:164, 179:180))
3 plot(zones_alt)
```

Nel codice precedente chiamiamo **zones()** che riceve in input l'oggetto trackeRdata, la variabile di interesse e gli intervalli da visionare nel grafico. Per esempio, in questo caso, visiona da: 135 a 164 con intervalli di 1, da 164 a 179 in un unico intervallo e poi l'intervallo 179-180.
Utilizziamo anche il comando **plot()** del pacchetto trackeR che vuole in input un oggetto creato da zones. Tramite questo comando si ottiene la percentuale di tempo speso dall'atleta in quella specifica altitudine.

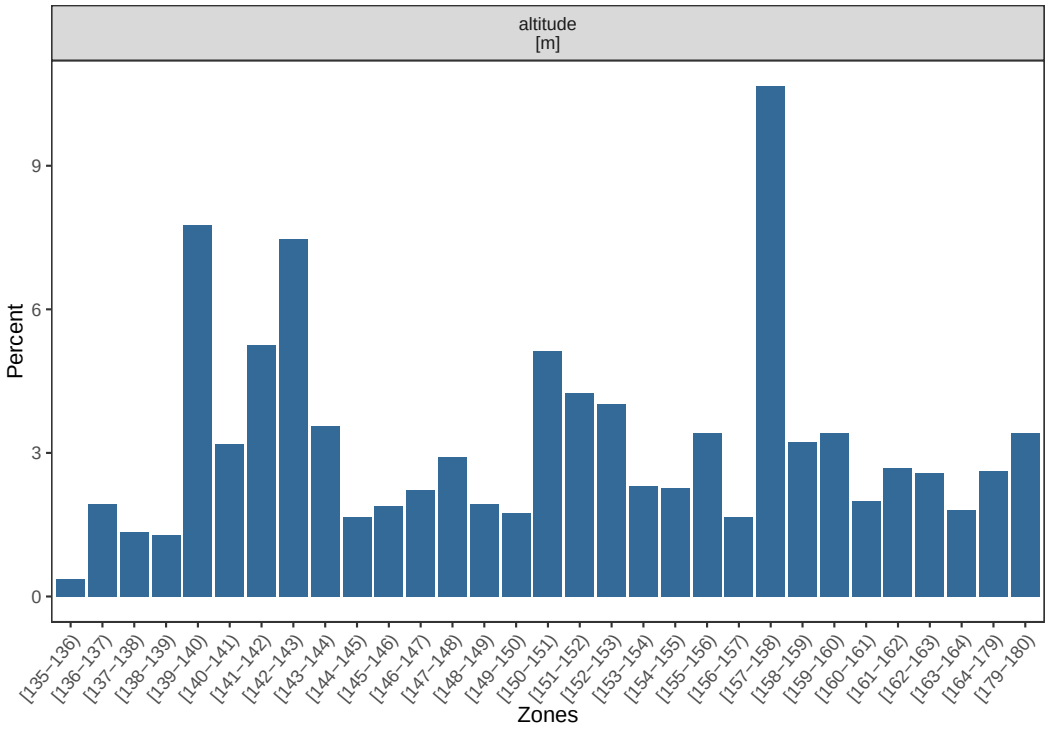
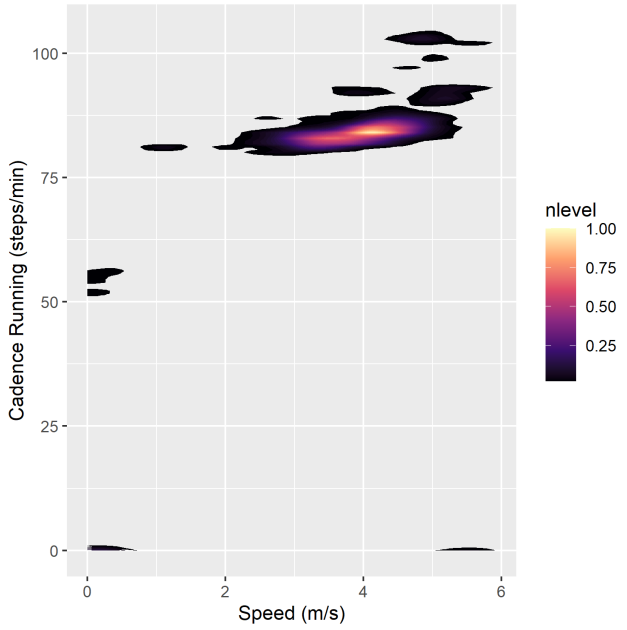


Figura 5.2: Zones - Altitude

5.2.4 Relazione Velocità-Cadenza

Il pacchetto **rayshader** creato da Tyler Morganwall e disponibile su GitHub, permette di creare grafici per la visualizzazione in R.
Il pacchetto si basa sull'uso di dati di elevazione ed una combinazione di algoritmi di ombreggiatura e sovrapposizioni per generare mappe 2D e 3D.
I modelli posso essere esaminati in maniera interattiva ed essere ruotati.
Nel seguente codice vengono utilizzate alcune funzioni del pacchetto **rayshader** e la libreria ggplot2, che permettono di vedere con più accuratezza la relazione tra la velocità e la cadenza di corsa.

```
ggmorning=ggplot(morningRun) +
    stat_density_2d(aes(x=speed, y=cadence_running,
        fill=stat(nlevel)), geom="polygon",
        n=200, bins=50, contour=TRUE) +
    scale_fill_viridis_c(option="A")
plot_gg(ggmorning, width = 5, height=5, raytrace FALSE,
    preview = TRUE)
```



Nello specifico il grafico che viene generato permette di visualizzare la relazione e la corrispondenza tra la velocità (m/s) e la cadenza di corsa (passi/min). Il colore, che varia da nero a bianco, fornisce una misura sulla quantità di dati presenti in quella zona del grafico.
Si può notare come nelle zone gialle si abbia una forte presenza di dati, mentre nelle zone nere, ce ne siano pochi.

5.2.5 Profilo Distribuzione

Il concetto di profili di distribuzione viene introdotto per la prima volta nel 2015 da Kosmidis e Passfield.
I profili di distribuzione vengono implementati in trackeR vista la necessità di comparare le diverse sessioni ed usare le informazioni durante la sessione.
Di seguito è presentato il codice che permette di visualizzare il profilo di distribuzione del battito cardiaco e della velocità:

```
1 dProfile <- distributionProfile(morningRunTD,
2     session = 1, what = c("speed", "heart_rate"),
3     grid = list(speed = seq(0, 12.5, by = 0.05),
4     heart_rate = seq(0, 250)))
5 plot(dProfile, multiple = TRUE)
```

distributionProfile restituisce un oggetto di classe **distrProfile**.
Nello specifico il grafico che viene generato permette di visualizzare i profili di distribuzione, prima del battito cardiaco, che ci dà un'idea di quanto abbia lavorato il cuore rispetto al tempo, poi della velocità, che ci permette di visualizzare la distribuzione della velocità rispetto al tempo.

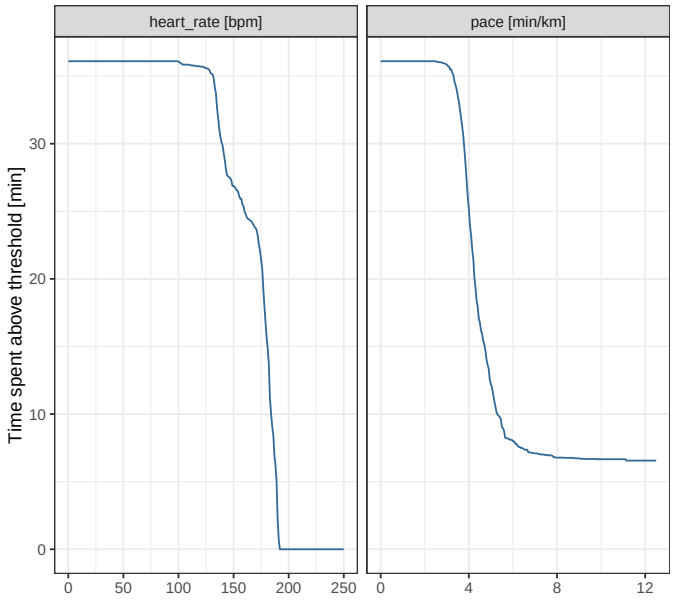


Figura 5.3: Profili di distribuzione

La funzione di distribuzione dei profili diventa molto utile nel momento in cui ci sono più sessioni nel dataset che si sta studiando. L'obiettivo di uno studio di questo tipo è il confronto tra queste sessioni.

Lo stesso ragionamento è valido per la funzione di concentrazione che si può trovare con:

```
dProfile <- distributionProfile(morningRunTD,
    session = 1, what = c("pace", "heart_rate"),
    grid = list(pace = seq(0, 12.5, by = 0.05),
    heart_rate = seq(0, 250)))
cProfile <- concentrationProfile(dProfile,
    what = "pace")
plot(cProfile, multiple = TRUE, smooth = TRUE)
```

Nel codice viene ricreato il dProfile che diventa l'argomento della funzione **concentrationProfile()**. L'output che produce è una curva che mostra come il ritmo(min/km) rispetto al tempo di durata della sessione.

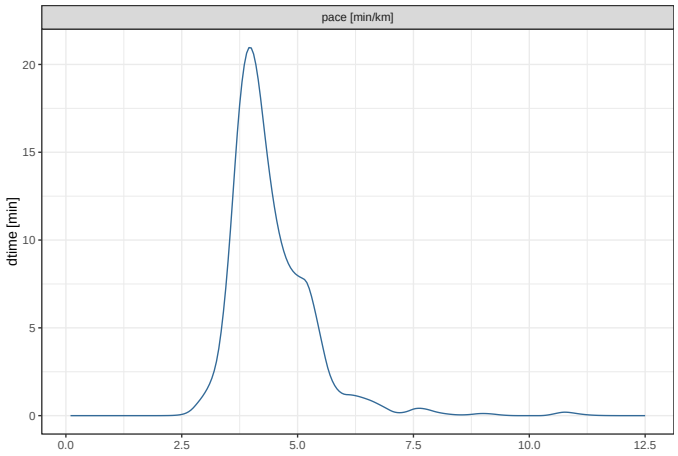


Figura 5.4: Profili di concentrazione

6 Conclusione

Prima della realizzazione di trackeR non era mai stato costruito un pacchetto che permettesse l'analisi grafica di sessioni di corsa, bici o nuoto.
Tuttavia sono stati creati altri strumenti per lavorare con i dati spazio-temporali, che includono adehabtitatLT (Calenge 2006), viaggio (Sumner 2015) e spostamento (Kranstauber e Smolla 2015). L'obiettivo principale di questi pacchetti è il monitoraggio degli animali, come la stima delle scelte dell'habitat, e non sono direttamente adatti per monitorare i vari aspetti dell'attività degli atleti. Nonostante l'ampia gamma di pacchetti R disponibili, ne esistono pochi specifici per i dati sportivi e la loro analisi.
I pacchetti disponibili si concentrano su argomenti come gestione sportiva (RcmdrPlugin.SM, Champely 2012), classifica squadre sportive (mvglmmRank, Karl e Broatch 2015) e accesso alle quote scommesse (pinnacle.API, Blume, Jhirad e Gassem 2015). SportsAnalytics è un pacchetto che si concentra sull'analisi dei dati sulle prestazioni (Eugster 2013). Il pacchetto cycleRtools (Mackie 2015) fornisce funzionalità per importare i dati di ciclismo in R, nonché strumenti per analisi descrittive specifiche per il ciclismo.
Di conseguenza è facile affermare che trackeR è l'unico pacchetto che permette l'analisi di questa tipologia di dati. Le sue caratteristiche lo rendono quasi indispensabile per un'analisi di questo tipo.

GitHub
Di seguito è presente il link GitHub della directory contenente il file R con i comandi utilizzati all'interno della tesina e la relazione in pdf: GitHub

7 Sitografia

[1] R

[2] GitHub trackeR

[3] Gpx

[4] Open Street Map

[5] GitHub Rayshader

[6] GitHub Tesina

[7] Hannah Frick