R lezione #1: tracciare e manipolare dati

Nicola Romanò

Introduzione

Il tracciamento è una parte estremamente importante dell'analisi dei dati. Infatti, dopo aver finito di raccogliere un set di dati, una delle prime cose che dovresti probabilmente fare è tracciare i dati. Questo ti permette di capire come sono fatti i tuoi dati e individuare errori evidenti nella loro collezione, o con la configurazione sperimentale. Tuttavia, un buon grafico può fare molto di più: ti può aiutare a scegliere il test statistico per analizzare quei dati, può aiutarti a comprendere meglio le cause alla base del fenomeno che stai osservando e aiutarti a spiegarle ai tuoi colleghi. Questo processo è così importante che John W. Tukey, uno dei padri della statistica moderna, ha coniato la frase *Observational Data Analysis* (ODA) per descriverlo [1].

[1] Tukey, Observational Data Analysis, 1977

La qualità dei dati ovviamente è molto importante. Ricorda la frase "garbage in, garbage out"! Se raccogli dati in modo "sciatto" non puoi aspettarti di ottenere risultati belli e riproducibili. Allo stesso modo, se non includi i controlli appropriati potresti avere dei dati meravigliosi ... ma che non significano nulla! Il modo in cui strutturi i tuoi dati è una questione estremamente importante da considerare. Certi formati potrebbero essere più facili da leggere da un essere umano, ma difficili da elaborare per un computer. È stato stimato che tra il 30% e l'80% dell'analisi dei dati consiste nella preparazione dei dati [2]. Questo è qualcosa che incontrerai più e più volte nella tua carriera di ricercatore, quindi è estremamente importante avere buone abitudini della raccolta dei dati.

[2] Dasu e Johnson 2003

Obiettivi formativi

Dopo aver completato questo workshop sarai in grado di:

- Leggere ed esplorare le caratteristiche principali di un set di dati
- Identificare dati sospetti / valori anomali e gestirli
- Gestire i punti dati mancanti
- Tracciare i dati e identificare visivamente le relazioni tra le variabili
- Convertire set di dati "larghi" in set di dati "lunghi"

Sezione 1 - Gestione e tracciamento dei dati di base mediante R

Questa prima sezione sarà un ripasso su come leggere, esplorare e tracciare i dati in R. Dovresti avere abbastanza familiarità con questi comandi, ma un aggiornamento aiuta sempre!

Per questa prima sezione useremo il file chiamato `metab-workshop1.csv`.

Iniziamo caricando il set di dati usando [3]

```
metab <- read.csv("metab-workshop1.csv")
```

[3] Dovrai cambiare il percorso in cui inserisci il file, ad es. `read.csv("C:/Workshop1/metab-workshop1.csv")`. In alternativa puoi dire a R di cercare i file in quella directory usando il comando `setwd`. Questo secondo approccio è molto utile se stai leggendo/scrivendo più file nello stesso script.

Questo set di dati contiene la concentrazione di un metabolita nel plasma dei pazienti dopo aver ricevuto un trattamento specifico. Abbiamo anche informazioni sul sesso e l'età del paziente. La prima cosa che dovremmo fare è esaminare i nostri dati.

Il comando `head` ci consente di esaminare le prime righe del set di dati. Semplicemente usalo come segue [4]:

```
head(metab)
```

[4] Prova ad usare il parametro `n` per vedere un numero specifico di linee. Se vuoi vedere la fine del tuo set di dati, puoi usare `tail` invece di `head`.

```
##   Concentration Sex Age Treatment
## 1      550.7270   F  79      CTRL
## 2      260.7356   M  41         A
## 3      450.8036   F  64      CTRL
## 4      324.3586   F  52      CTRL
## 5      228.7021   M  43         B
## 6      325.0527   M  53         A
```

Il set di dati mostra i dati per ciascun paziente in una riga e contiene 4 colonne che mostrano i valori misurati o le caratteristiche per quel paziente specifico. Possiamo usare il comando `colnames` per vedere il nome delle colonne nel nostro set di dati. [5]

[5] Non sorprende che ci sia anche una funzione `rownames`. Provalo!

```
colnames(metab)
```

```
## [1] "Concentration" "Sex"
## [3] "Age"           "Treatment"
```

Ora possiamo verificare quanti pazienti abbiamo. Poiché ogni riga del nostro set di dati rappresenta un paziente, possiamo semplicemente contare il numero di righe usando `nrow` [6].

[6] Puoi indovinare come contare le colonne invece?

```
nrow(metab)
```

```
## [1] 430
```

Supponiamo di voler sapere quanti di questi 430 pazienti sono uomini e quante sono le donne. Ci sono alcuni modi per farlo in R. Ad esempio, è possibile suddividere i dati e quindi contare le righe. [7]

[7] Ricorda che R usa le parentesi quadre per accedere al contenuto di un set di dati (il nome R corretto per ciò che si sta utilizzando è un *data frame*). Ad esempio, scrivendo `data[1,5]` si otterrà l'elemento nella riga 1, colonna 5. Usando `data[1,]` si otterrà la riga 1 e tutte le colonne.

```
# Prendi tutte le righe con _Sex_ = 'F' e
# tutte le colonne.
females <- metab[metab$Sex == "F", ]
nrow(females)
```

```
## [1] 215
```

```
nrow(metab) - nrow(females)  # Numero di uomini = totale - donne
```

```
## [1] 215
```

Un modo più elegante per farlo è usare la funzione `table`.

```
table(metab$Sex)
```

```
##
##   F   M
## 215 215
```

È possibile passare più parametri alla tabella (separati da ","). Prova a creare una tabella per sesso e trattamento [8].

Quanti uomini ci sono nel gruppo di controllo? _____

Un altro modo intelligente per esplorare i dati è usare la funzione `summary`:

```
summary(metab)
```

```
##  Concentration   Sex          Age
##  Min.   :-10.5   F:215   Min.   :35.00
##  1st Qu.:286.9   M:215   1st Qu.:45.00
##  Median :372.6           Median :58.00
##  Mean   :385.9           Mean   :56.95
##  3rd Qu.:466.7           3rd Qu.:68.00
##  Max.   :862.5           Max.   :80.00
##  Treatment
##  A   :150
##  B   : 80
##  CTRL:200
##
##
##
```

Questo ci dice molto! Abbiamo quattro variabili (concentrazione, sesso, età e trattamento). La concentrazione è una variabile continua, quindi otteniamo statistiche sulla sua distribuzione. Il sesso è una variabile discreta (in R questi sono chiamati fattori) e ha due possibili livelli: F e M. [9] Che tipo di variabili sono età e trattamento?

_____

Noti qualche problema con questi dati? _____

[8] Tieni presente che l'ordine con cui si passano i parametri a `table` è importante! Prova a passare `metab\$Sex` prima o dopo `metab\$Treatment`. Che succede?

[9] R ordina i livelli alfabeticamente, vedremo come cambiarlo più avanti.

Riesci a pensare a un modo per correggere il problema?

Ovviamente puoi anche ottenere statistiche riassuntive usando funzioni R come `mean`, `median`, `range`, `min`, `max`, `quantile` [10]

Per esempio

```
median(metab$Age)
```

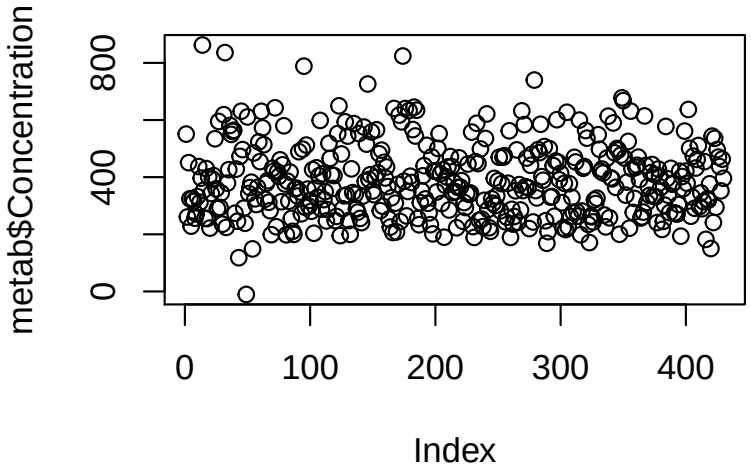
```
## [1] 58
```

[10] Di default la funzione quantile restituisce 5 quantili (min, 25%, mediana, 75%, max), ma prova a specificare il parametro `probs`, ad esempio come `quantile(metab $ Age, probs = c (0.3, 0.6))`. Che succede?

Sezione 2 - Tracciamento di base

Ora vogliamo iniziare a tracciare i valori della tabella.

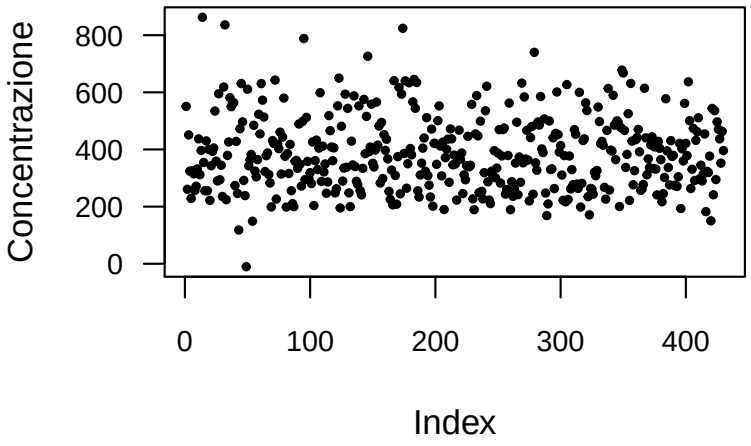
```
plot(metab$Concentration)
```



Questo grafico mostra le 430 misure di concentrazione nel nostro set di dati e ci consente facilmente di stimare la media [11]. A parte questo, non è particolarmente utile, è piuttosto brutto e disordinato. Facciamolo diventare più carino!

[11] Puoi intuire la media dei valori osservando il grafico? Usa la funzione `mean` per calcolare la media effettiva. Eri vicino?

```
plot(metab$Concentration, pch = 20, cex = 0.7,
    cex.axis = 0.8, las = 1, ylab = "Concentrazione")
```



Abbiamo aggiunto diversi parametri:

- `pch` cambia il *p* lotting *ch* aracter [12]
- `cex` modifica la dimensione dei punti di tracciamento. Più piccolo è il valore, più piccola è la dimensione del punto, il valore predefinito è 1. A seconda della configurazione del computer, valori diversi possono dare risultati più belli.
- `cex.axis` modifica la dimensione delle etichette degli assi (sia x che y).
- `ylab` imposta l'etichetta per l'asse y [13].
- Prova a cambiare il parametro `las` e vedere cosa fa.

Ci sono molti più parametri che puoi usare, ne vedremo alcuni, ma se vuoi vedere il tipo di lista completa `help(par)` [14].

Ora che il grafico sembra più bello, proviamo a renderlo più utile! Possiamo colorare i punti in base al valore nella colonna Sesso. Creiamo una nuova variabile, che contiene 430 elementi, uno per ogni campione, uguale a `"darkgreen"` se il campione proviene da un uomo e `"purple"` se proviene da una donna. [15]. Assegniamo `"darkgreen"` a tutti, usando la funzione `rep` [16], Quindi cambiamo in `"purple"` i valori corrispondenti ai campioni delle donne.

[12] Ci sono molti caratteri di plottaggio diversi, prova diversi numeri per `pch` e vedi cosa succede. Puoi persino utilizzare un carattere di tua scelta, ad es. `pch = 'x'`.

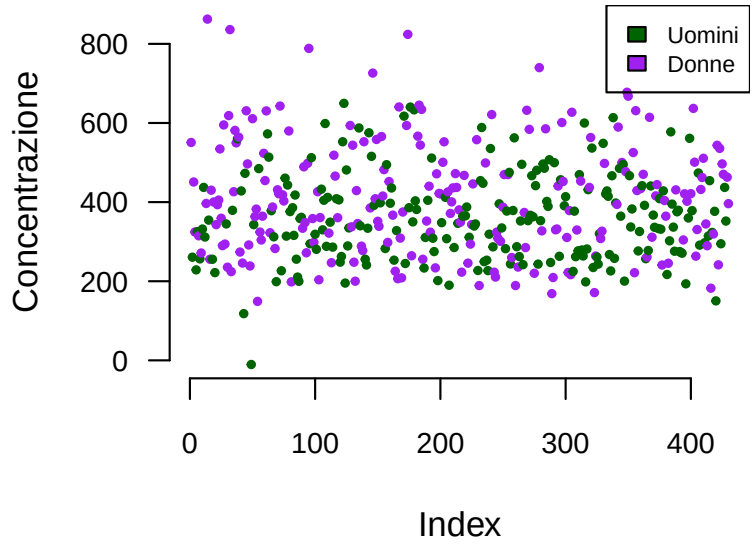
[13] Prova a cambiare l'etichetta dell'asse x

[14] `par` è una funzione estremamente potente, ti permette di impostare il tracciamento predefinito parametri per tutti i grafici che verranno tracciati dopo averlo chiamato. Permette anche di avere più grafici nella stessa figura, usando il parametro `mfrow`. Prova ad esempio scrivendo `par(mfrow = c (1, 3))`, quindi facendo 3 diversi grafici. Puoi tracciare la concentrazione degli uomini in alto e delle donne in basso? Presta attenzione alla scala dell'asse y!

[15] Curioso di sapere quali colori sono disponibili in R? Digita `colours()`!

[16] La funzione `rep` ripete un valore un numero specifico di volte. Ad esempio, prova, `rep(5, 8)`. Notare le virgolette attorno ai valori, indicando che questi non sono nomi di variabili ma i valori effettivi che vogliamo.

```
pointcolor <- rep("darkgreen", nrow(metab))
pointcolor[metab$Sex == "F"] <- "purple"
plot(metab$Concentration, col = pointcolor, bty = "n",
    pch = 20, cex = 0.7, cex.axis = 0.8, las = 1,
    ylab = "Concentrazione")
legend("topright", legend = c("Uomini", "Donne"),
    fill = c("darkgreen", "purple"), cex = 0.7)
```



Ci sono anche altri modi per generare il colore per i punti. Uno è usare la funzione `ifelse`. Potremmo scrivere:

```
pointcolor <- ifelse(metab$Sex == "F", "purple",
    "darkgreen")
```

Questo è equivalente alle prime due righe del codice sopra. In sostanza si legge "Per ogni elemento di `metab$Sex`, se è uguale a `"F"`, allora imposta `pointcolor` a `purple`, altrimenti impostalo a `"darkgreen"`". Funziona perché abbiamo solo due opzioni, le situazioni più complesse con molti colori potrebbero richiedere un po più di lavoro!

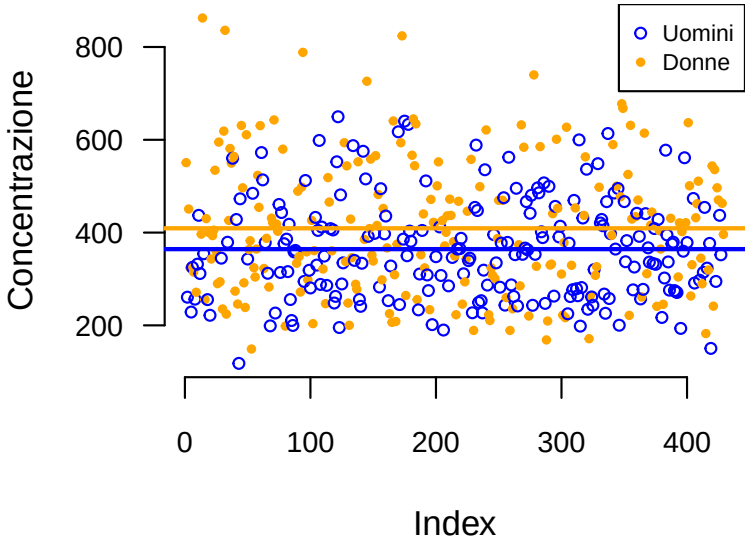
A questo punto, avresti dovuto vedere che c'è un valore negativo di concentrazione! Poiché questo è chiaramente un errore ˆ [Perché?], Possiamo tranquillamente rimuoverlo.

```
# Questa funzione ottiene le righe che
# contengono valori negativi, in questo caso
# riga 49
wrong.val <- which(metab$Concentration < 0)
# Usando - davanti al numero di riga (o
# colonna) rimuoviamo quella riga (o colonna).
# Riassegnando il risultato a metab,
# sovrascriviamo il suo valore precedente
metab <- metab[-wrong.val, ]
```

Infine, possiamo ritracciare i dati, e questa volta aggiungeremo anche linee orizzontali corrispondenti alla media dei valori per uomini e donne. Per disegnare una linea retta su un grafico esistente puoi usare il comando `abline`. Ad esempio, questi due comandi disegneranno una linea rossa parallela all'asse x a $y = 5$ e una linea verde parallela all'asse y a $x = 10$

```
abline(h = 5, col = "red")
abline(v = 10, col = "green")
```

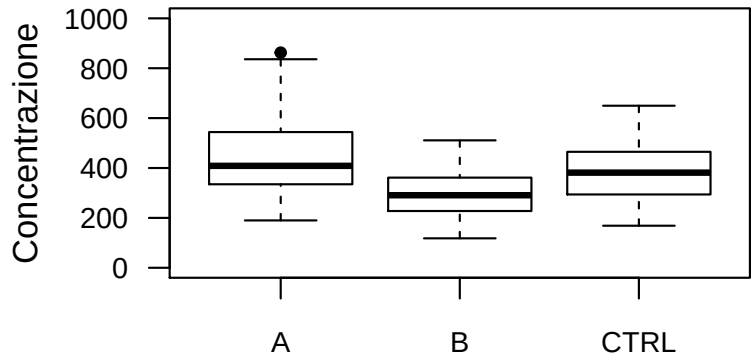
Ora prova a generare il grafico seguente usando i comandi che hai imparato finora.



Sezione 3 - Raggruppare i dati

Tracciare singoli punti di dati è utile, ma il più delle volte vogliamo raggruppare i dati secondo qualche fattore significativo. I `boxplot` (conosciuti anche come grafici `box-and-whisker`) sono molto utili per questo scopo.

```
boxplot(Concentration ~ Treatment, metab, las = 1,
    ylim = c(0, 1000), pch = 20, ylab = "Concentrazione")
```



La funzione `boxplot` ci permette di tracciare un valore rispetto a uno o più fattori [17]. Come abbiamo notato sopra, R ordina i fattori alfabeticamente, quindi i gruppi appaiono nell'ordine A, B e Control. Volessimo cambiare l'ordine, potremmo fare:

[17] Perché c'è un punto sopra il boxplot per il gruppo A?

```
metab$Treatment <- factor(metab$Treatment, levels = c("CTRL",
    "A", "B"))
```

Dopo aver fatto ciò, il boxplot sarà ordinato correttamente [18].

[18] Un altro punto di discussione: cosa fa esattamente la linea sopra?

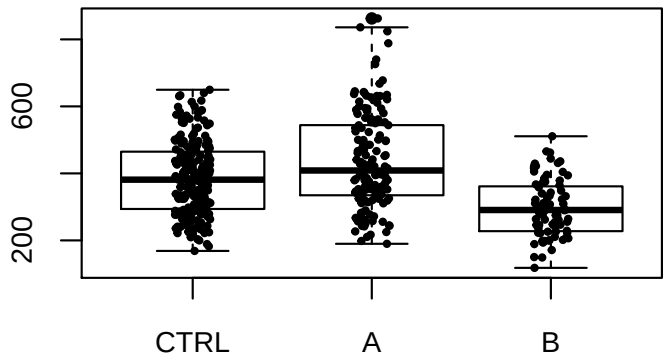
Possiamo anche combinare più di un fattore nel boxplot, usando + (ad esempio `Concentration ~ Treatment + Sex`)

Prova a tracciare la concentrazione a seconda del trattamento, separatamente per uomini e donne.

Ora prova a scambiare Sex and Treatment, cosa succede?

A volte è utile tracciare singoli punti sul boxplot. Questo può essere fatto usando la funzione `stripchart`.

```
boxplot(Concentration ~ Treatment, metab, pch = 20)
stripchart(Concentration ~ Treatment, metab, add = TRUE,
    vert = TRUE, pch = 20, cex = 0.6, method = "jitter",
    jitter = 0.1)
```



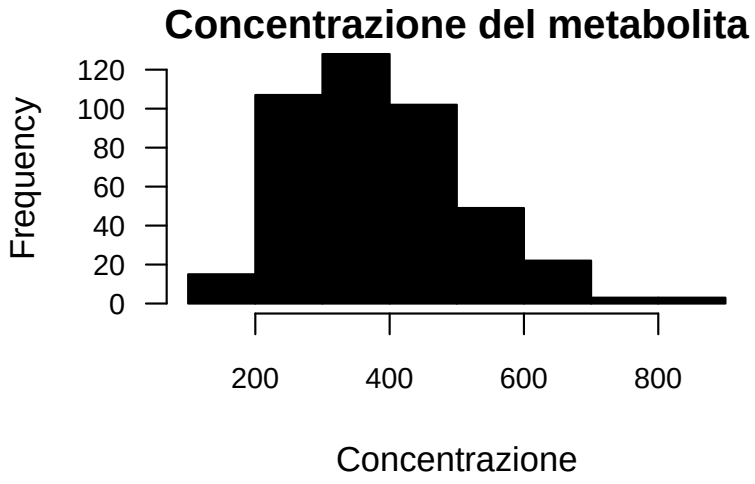
Dobbiamo passare due parametri importanti a `stripchart:add` dice a R di disegnare sopra il grafico esistente, invece di produrne uno nuovo. `vert` specifica di tracciare la stripchart verticalmente e non orizzontalmente [19].

[19] Il parametro `jitter` sposta i punti ed è molto utile quando ci sono molti punti sovrapposti. Funziona solo quando si specifica `method ="jitter"`. Prova a cambiare il valore di `jitter` e vedi cosa succede!

Sezione 4 - Istogrammi

Gli istogrammi sono un ottimo modo per riepilogare i dati. Generiamo un istogramma della concentrazione del nostro set di dati.

```
hist(metab$Concentration, col = "black", las = 1,
    main = "Concentrazione del metabolita", xlab = "Concentrazione")
```

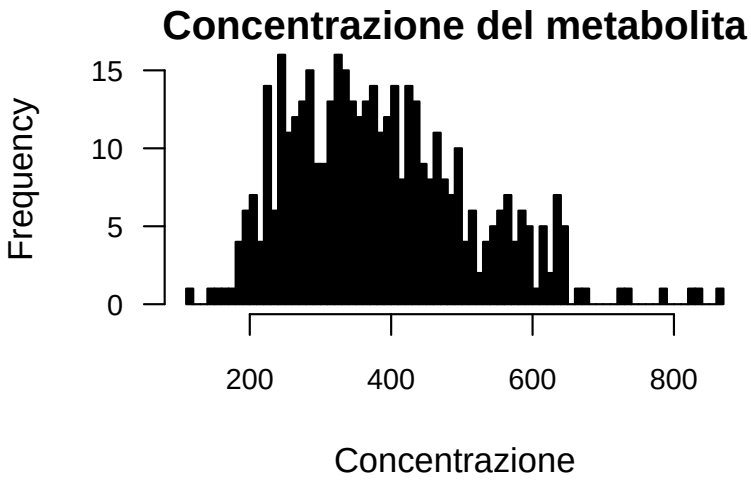


Il parametro `br` può essere usato per specificare il numero di colonne (i 'break') nell'istogramma [20]. Può essere utilizzato in due modi: specificando il numero di colonne o specificando i valori nei quali vogliamo suddividere l'istogramma.

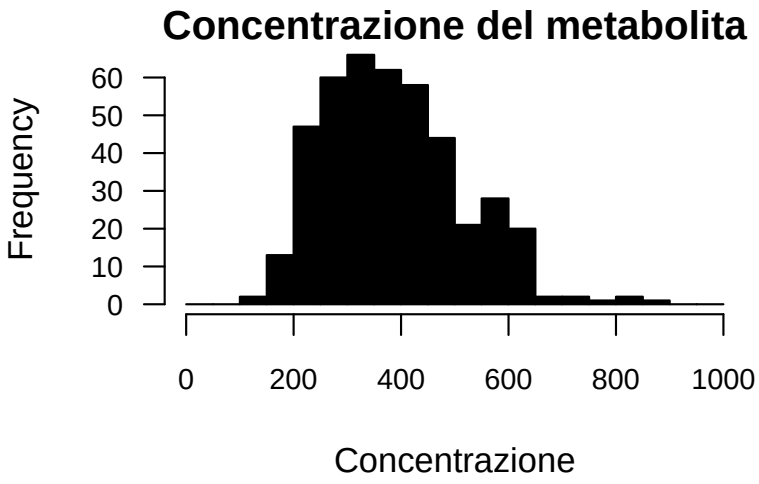
[20] Di default R usa la formula di Sturge per determinare il numero di colonne. Esistono molti modi diversi per calcolare questo numero. Wikipedia ha una sezione abbastanza lunga sull'argomento: `https://en.wikipedia.org/wiki/Histogram`

Per esempio:

```
# Istogramma con 100 colonne
hist(metab$Concentration, col = "black", las = 1,
     br = 100, main = "Concentrazione del metabolita",
     xlab = "Concentrazione")
```




```
# Istogramma con colonne da 0 a 1000 di
# larghezza = 50
hist(metab$Concentration, col = "black", las = 1,
     br = seq(0, 1000, 50), main = "Concentrazione del metabolita",
     xlab = "Concentrazione")
```



Quale istogramma è il migliore?

Sezione 5 - Altri grafici

Ci sono molti altri grafici che R è in grado di produrre, compresi alcuni molto complessi. Alcuni grafici richiedono l'installazione di pacchetti aggiuntivi (come `ggplot2`). Non li copriremo qui, ma ci sono molte risorse online che puoi usare per imparare, se lo desideri.

Cosa fa questo codice? Il grafico risultante è più o meno utile di un boxplot? Perché? Cerca di eseguirlo, leggere la guida di R e cambiare le cose.

```
males <- metab$Concentration[metab$Sex == "M"]
females <- metab$Concentration[metab$Sex == "F"]

mean.M <- mean(males)
mean.F <- mean(females)
sd.M <- sd(males)
sd.F <- sd(females)
# barplot ci dà la coordinata x di ogni
# colonna
bp <- barplot(c(Uomini = mean.M, Donne = mean.F),
    col = c("green", "orange"), ylim = c(0, 600),
    las = 1, ylab = "Concentrazione")
arrows(bp, c(mean.M - sd.M, mean.F - sd.F), bp,
    c(mean.M + sd.M, mean.F + sd.F), angle = 90,
    code = 3)
```

Sezione 6 - Gestione dei dati mancanti

La funzione `complete.cases` può essere molto utile quando si ha a che fare con dati mancanti. Facciamo pratica usandola!

Leggi ed esplora il set di dati `neurons-workshop1.csv`. Questo contiene il numero di neuroni che mostrano l'espressione di una determinata proteina in 5 diverse regioni del cervello. Durante l'elaborazione alcune sezioni del cervello sono andate perse, prima che il conteggio potesse essere fatto. [21]

Inizia esplorando il set di dati. Quanti valori mancanti ci sono? [22]

Ora inizia ad esplorare il set di dati; traccia le varie variabili, vedi se riesci a individuare qualche tendenza o relazione.

Ora proviamo a calcolare il conteggio medio per regione. Dovresti avere tutti gli strumenti necessari per farlo da solo! [23]

[21] Che tipo di dati mancanti è questo?

[22] Suggerimento: usa `summary`! Alternativamente `which(is.na(neurons$counts))` ti dirà quali osservazioni (= righe) sono `NA`.

[23] Una funzione R che renderà molto facile fare è `aggregate`. Vedi se riesci a capire come!

I tuoi calcoli dovrebbero avere come risultato:

Regioni	Conteggi
A	47.8
B	NA
C	9.4
D	26.9
E	NA

I gruppi B ed E hanno dati mancanti, quindi `mean` restituisce`NA` come risultato.

Possiamo usare `complete.cases` per rimuovere quei punti dati permettendo così che si verifichino i calcoli corretti [24].

[24] Cosa stiamo facendo quando usiamo `complete.cases`?

Esiste una correlazione tra numero di neuroni e età dell'animale? Usa la funzione `cor` per calcolarla. Prova a confrontare il risultato della correlazione con l'omissione a coppie e con l'omissione di lista. Come cambia il risultato? Perché?

```
neurons2 <- neurons[complete.cases(neurons), ]
nrow(neurons)
```

```
## [1] 80
```

```
mean(neurons$counts)
```

```
## [1] NA
```

```
nrow(neurons2)
```

```
## [1] 78
```

```
mean(neurons2$counts)
```

```
## [1] 26.21795
```

Notate che possiamo anche usare `complete.cases` assieme a `which` per sapere quali sono le osservazioni mancanti. [25]

[25] Questo funziona perché `complete.cases` restituisce `TRUE` se l'osservazione è completa, e `FALSE` se mancano dati.

```
which(complete.cases(neurons) == FALSE)
```

```
## [1] 32 45
```

Questo ci dice che mancano le osservazioni 32 e 45. In effetti possiamo verificarlo tramite:

```
neurons[32, ]
```

```
##    mouse sex age region counts
## 32     7   F  94      B     NA
```

```
neurons[45, ]
```

```
##    mouse sex age region counts
## 45     9   M 112      E     NA
```

Sezione 7 - Conversione del formato dei dati

L'ultima parte di questa lezione riguarda la formattazione dei dati. Molto spesso dovrete trattare con dati disordinati, non formattati in un modo con cui è facile lavorare (ad esempio, i dati potrebbero essere in formato "largo"). Ciò potrebbe essere dovuto al fatto che qualcun altro ha registrato i dati o perché spesso i dati di grandi dimensioni sono più facili da consultare in software come Excel. Leggi il file `lizard.csv`. Questo file contiene i conteggi per il numero di tre diverse specie di lucertole trovate in 5 luoghi diversi in una regione. Se ispezioniamo i dati, vediamo che è in un formato "lungo".

```
lizard <- read.csv("lizard-workshop1.csv")
head(lizard)
```

```
##     Lizard Location1 Location2 Location3
## 1 Species1         18         22         15
## 2 Species2          0          1          6
## 3 Species3         25         22         30
##   Location4 Location5
## 1         25         10
## 2          4          3
## 3         11         10
```

Ogni riga rappresenta una specie e ogni colonna una posizione. Questo non è un formato utile da usare in R, poiché molte funzioni saranno molto difficili da eseguire. Per esempio, se volessimo tracciare il numero di lucertole di ogni specie per posizione, non abbiamo un modo semplice per farlo [26].

Fortunatamente R ci viene in soccorso! Per rimodellare i dati abbiamo bisogno di installare un pacchetto extra, chiamato `reshape2` [27]. Per installare il pacchetto possiamo semplicemente eseguire:

```
install.packages("reshape2")
```

È necessario farlo solo una volta, se il pacchetto è già installato, la riga sopra non è necessaria. Una volta che il pacchetto è installato, dobbiamo dire a R di usarlo; lo facciamo usando il comando `library` (questa linea dovrai usarla ogni volta).

```
library(reshape2)
```

Ora possiamo usare tutte le funzioni fornite da `reshape2`, in particolare vogliamo la funzione`melt`, che converte i dati da "larghi" a "lunghi".

```
lizard.long <- melt(lizard, id.vars = "Lizard",
    variable.name = "Location", value.name = "Counts")
```

[26] Ci sono modi per tracciare dati "larghi", ovviamente, ma per esempio funzioni come `lm` vogliono un formato di dati "lungo".

[27] Ci sono anche altri pacchetti come`tidyr` che possono eseguire lo stesso compito, sentiti libero di provarlo se lo desideri.

Questo dice a R di usare la colonna Lizard come nostro `ID`, che è l'identificatore del soggetto che viene misurato. I parametri `variable.name` specificano il nome della variabile (o delle variabili) che stiamo cambiando, in questo caso la posizione (questo è solo il nome della nuova colonna nel set di dati lungo). Infine, value.name è il nome della colonna per ciò che stiamo misurando attualmente.

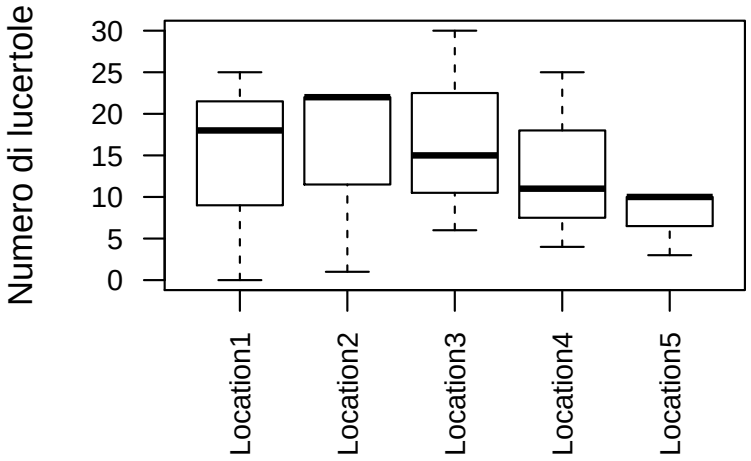
Vediamo cosa abbiamo ottenuto!

```
head(lizard.long)
```

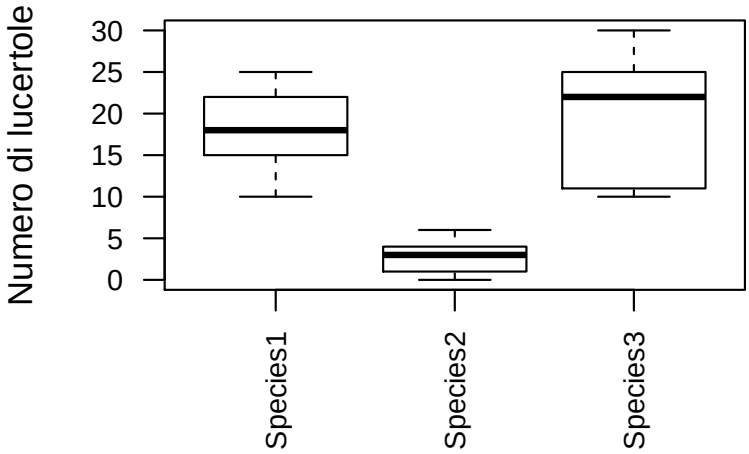
```
##     Lizard  Location Counts
## 1 Species1 Location1     18
## 2 Species2 Location1      0
## 3 Species3 Location1     25
## 4 Species1 Location2     22
## 5 Species2 Location2      1
## 6 Species3 Location2     22
```

Ora possiamo facilmente fare:

```
boxplot(Counts ~ Location, lizard.long, las = 2,
    ylab = "Numero di lucertole")
```




```
boxplot(Counts ~ Lizard, lizard.long, las = 2,
    ylab = "Numero di lucertole")
```



A volte le conversioni dai dati "larghi" a quelli "lunghi" sono un po' più complicate, ma `melt` è una funzione molto potente e il più delle volte sarai in grado di rimodellare i tuoi dati.

Ora prova ad aprire il file `BP-workshop1.csv`. Contiene i dati di un esperimento in cui partecipanti di sesso ed età differenti hanno assunto due diversi farmaci. I dati sono stati resi anonimi, in modo che ogni partecipante sia rappresentato da un ID (2 lettere e 3 numeri). La pressione sanguigna di ogni partecipante è stata misurata prima e dopo l'assunzione di ciascuno dei farmaci, e il relativo cambiamento della pressione sanguigna è stato calcolato. Il set di dati contiene la suscettibilità calcolata per ciascun partecipante.

- In quale formato sono i dati (ad esempio, largo, lungo, ...)?
- Se necessario, convertire i dati in formato lungo
- Quanti partecipanti erano nello studio? Quanti uomini e quante donne?
- Quanti partecipanti avevano più di 50 anni? Quante di quelle erano donne?
- La risposta alle droghe è diversa tra uomini e donne? E tra partecipanti più giovani e più anziani?
- Traccia la risposta ai due farmaci usando un istogramma, un boxplot o un barplot. Quale rappresentazione è più utile? Perché?
- Guarda i dati per il paziente OV019. C'è qualcosa di sorprendente in questo paziente? Cosa pensi che dovresti fare se continuassi e analizzassi questi dati?