Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

ANALISI ESPLORATIVA E PREDITTIVA DI DATI CON R

Relatore:
Chiar.ma Prof.ssa
ELENA LOLI PICCOLOMINI

Presentata da:
GIUSEPPE PIO SALCUNI

Sessione III
Anno Accademico 2020/2021

A nonno Giuseppe

Indice

Introduzione

La Tesi in questione tratta l'argomento relativo all'analisi esplorativa dei dati (Exploratory Data Analysis o EDA). Come già noto essa viene utilizzata per esplorare, analizzare e apprendere dai dati attraverso l'uso di grafici e visualizzazioni per lo studio, ma non per confermare ipotesi statistiche. L'analisi esplorativa dei dati è dunque uno strumento potente per studiare un dataset. In poco tempo riusciremo a farci un'idea realistica di come i dati sono organizzati e potremmo individuare valori anomali ed errori. Questa tipologia di analisi ci permette di comprendere le potenziali relazioni tra le variabili e ci consente di formulare ipotesi interessanti che possono essere verificate in seguito, attraverso l'ausilio di metodi statistici più formali. Inizialmente viene posto lo studio di una variabile alla volta, successivamente se ne studieranno due e, infine, molte contemporaneamente. Una volta completata l'analisi e lo studio, si giunge ad una conclusione sui dati attraverso la creazione di grafici e altri strumenti esplorativi, che permettono di evidenziare e rappresentare il lavoro svolto. Nel caso in cui questi strumenti non dovessero bastare, i dati saranno osservati da un'altra prospettiva.
Per effettuare una corretta analisi dei dati facciamo affidamento su degli strumenti (software) di *data science*:

- **Python**: un linguaggio di programmazione caratterizzato da strutture di dati integrate di alto livello, che lo rendono particolarmente interessante per un rapido sviluppo di applicazioni e per un possibile impiego come linguaggio di scripting, al fine di connettere tra loro le componenti esistenti.

- **R**: un linguaggio di programmazione open source per il calcolo statistico e grafici, supportato da *R foundation for Statistical computing*. Questo linguaggio è ampiamente utilizzato dagli statistici nella formulazione di osservazioni statistiche e analisi dei dati.

Questo lavoro di tesi si occupa in una prima parte di come effettuare una completa analisi dei dati, partendo dalla loro ricerca fino all'ottenimento di risultati da dimostrare. Nella seconda ed ultima parte verrà mostrato un caso studio legato alla previsione del prezzo di uno smartphone.

Capitolo 1

La visualizzazione dei dati

È importante conoscere i vantaggi delle tecniche di visualizzazione e perché dovremmo usarle per l'analisi esplorativa dei dati. La visualizzazione dei dati è fondamentale al fine di comprendere i dati ottenuti, specialmente nel caso in cui essi non siano stati prodotti da noi stessi, e per la scelta dei metodi di analisi più opportuni. Ci sono almeno tre motivazioni per analizzare i dati:

1. Comprendere cosa è successo o cosa stia succedendo;
2. Prevedere cosa potrebbe accadere in futuro o in altre circostanze non ancora verificate;
3. Avere una guida nel processo decisionale.

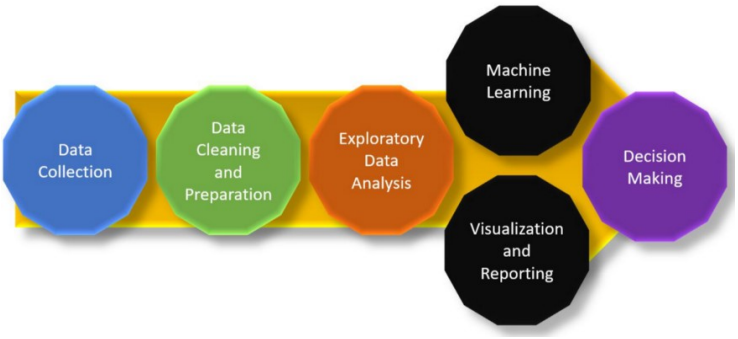


Figura 1.1: Flusso del processo dalla raccolta dei dati al processo decisionale.

1.1 Dati

Con il termine *dati* si fa riferimento ad una raccolta di dettagli, registrati per caratterizzare la fonte come:

- **Entità**, ad esempio un insieme di caratteristiche di aziende concorrenti in un'analisi di marketing;
- **Evento**, ad esempio le caratteristiche demografiche di chi ha votato candidati politici diversi in una data elezione;
- **Processo**, ad esempio i dati operativi di un processo di produzione industriale.

Per poter analizzare i dati si fa riferimento ad un *dataset*, ovvero una collezione di dati strutturati (ad esempio una tabella) creato per essere elaborato da un algoritmo. Il dataset può essere creato sfruttando più tecniche:

1. **Creazione Manuale**: utilizzando software come *Microsoft Excel*, generando tabelle relative ai dati.
2. **Creazione Semi-Automatica**: utilizzando un *DBMS (Database Management System)* si può creare una base di dati in cui potranno essere inseriti periodicamente nuovi dati.
3. **Creazione Automatica**: sempre con l'ausilio di un DBMS, questa volta insieme a un'applicazione web, sarà possibile creare dei moduli di raccolta dati per utenti che, grazie alla loro interazione con essi, forniranno dati che saranno automaticamente memorizzati.

Sono molti gli strumenti disponibili per la creazione e la gestione dei dataset, tra i più famosi è possibile trovare *Microsoft Excel*, *LibreOffice* di *The Document Foundation*, *SAS* di *The SAS Institute*, *Microsoft Azure*, *Google Cloud Platform*.
Solitamente la fase di creazione dei dati può richiedere molto tempo. Un'alternativa

pratica ed efficace potrebbe essere la ricerca online di dataset pronti all'uso. Tra le migliori fonti online per il reperimento dei dataset si ha:

1. *Kaggle*
2. *Google Dataset Search*
3. *Amazon Web Services Open Data*

1.2 Software per l'analisi

Molto importante per l'analisi dei dati è affidarsi ad un software in grado di gestire un grande quantitativo di dati con diversi metodi per rappresentarli. Tra i più famosi ci sono R e Python. R è un linguaggio open source gratuito, preferito in particolar modo per due motivi principali:

1. È caratterizzato da librerie aggiuntive che possono essere gratuite o a pagamento, supportanti metodi di analisi di molti rami della statistica, utili per poter effettuare analisi specifiche.
2. Essendo un linguaggio molto popolare, tende ad essere aggiornato con frequenza, favorendo la crescita del numero di nuove funzionalità.

Python è un software flessibile caratterizzato da grande leggibilità del codice. Ha inoltre un'elevata facilità di implementazione e riproducibilità.
Python è preferibile se:

- Si hanno già esperienze di programmazione.
- Si devono implementare funzioni di matematica computazionale.

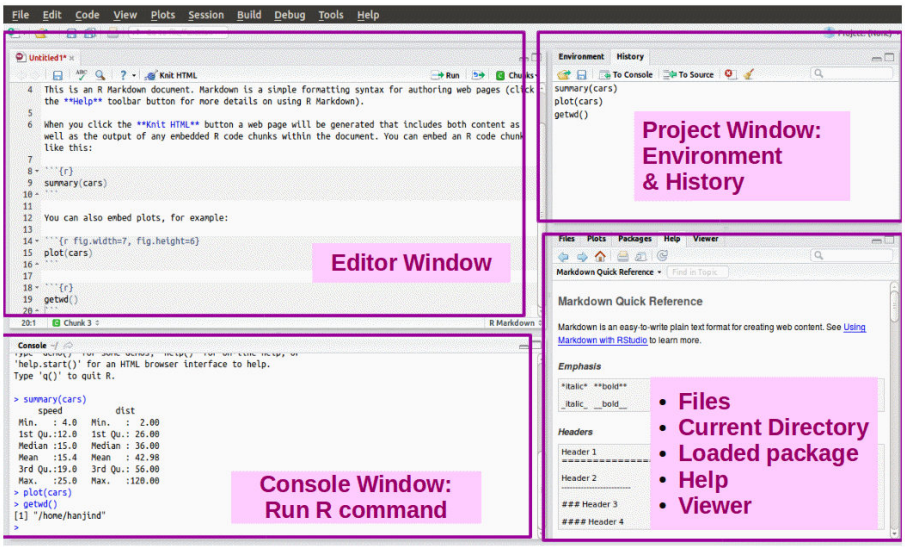


Figura 1.2: Screenshot illustrativo dell'interfaccia di RStudio.

1.3 La struttura di R

Il software che si è deciso di analizzare nel dettaglio è RStudio

Questo linguaggio è arricchito fondamentalmente da tre componenti:

- Pacchetti R di base, supportanti le funzioni standard di statistica di base e funzioni di analisi dei dati.
- Pacchetti consigliati, automaticamente inclusi in RStudio.
- Pacchetti aggiuntivi opzionali, disponibili tramite i *Comprehensive R Archive Network (CRAN)*, i quali permettono di ottenere una vasta personalizzazione.

Una volta scaricato il pacchetto, per poterlo utilizzare, è necessario caricarlo attraverso la funzione *library()*, che funziona in due modi differenti. Specificando tra parentesi il nome di un pacchetto già installato, lo si includerà nel progetto; ad esempio se si volesse includere il pacchetto ***MASS*** si scriverà:

```
1 library(MASS)
```

Se invece si accede a questa funzione senza specificare alcun parametro tra parentesi, verrà aperta una nuova finestra in cui saranno elencati tutti i pacchetti installati.

R è un linguaggio di programmazione caratterizzato da una sintassi semplice. I comandi elementari consistono in espressioni o assegnazioni, ed ogni comando viene inserito nella riga che comincia con il simbolo di *prompt*, solitamente " > ". RStudio fornisce la possibilità di richiamare e rieseguire comandi. Tutte l'entità che R crea e manipola sono conosciute come oggetti, indipendentemente dal fatto che siano variabili, caratteri, stringhe, array di numeri e funzioni. Tutti questi oggetti possono essere permanentemente immagazzinati in un file, per essere riutilizzati in sessioni future. R opera su strutture di dati, e la più semplice struttura è costituita da un vettore numerico, ovvero un'entità formata da un insieme ordinato di numeri; ad esempio:

```
1 x<-c(5.4, 7.8, 10, 3.2, 5, 6.3, 8)
```

Ci sono altre strutture dati più complesse, che non verranno analizzate nel dettaglio. Si cercherà di mostrare gli aspetti essenziali di R omettendo alcune precisazioni e tecniche.

Capitolo 2

L'esplorazione di un Dataset

Gli scopi del'esplorazione vanno ben oltre la conoscenza dei dati stessi e la corretta predisposizione del dataset. In questa fase si possono conoscere molti aspetti di tutto il processo di analisi in cui si sta operando. Per poter correttamente esplorare un dataset è sicuramente utile rispettare i punti seguenti:

1. Valutare le caratteristiche generali del dataset rispondendo alle domande:
    - Quanti record possediamo?
    - Quante variabili?
    - Quali sono i nomi delle variabili? Sono significativi?
    - Che tipo di variabili possediamo? (esempio: numerica, logica, ecc.)
    - Qual è il valore che si verifica più frequentemente e con quale frequenza?
    - Ci sono delle osservazioni mancanti? Se sì, con quale frequenza si verifica?
2. Esaminare le statistiche descrittive per ogni variabile;
3. Effettuare un'analisi esplorativa per delle variabili di particolare interesse dove possibile;
4. Localizzare dove sia possibile applicare tutte le procedure indicate dal software per identificare eventuali anomalie dei dati;

5. Osservare le relazioni tra le variabili chiave utilizzando dei grafici come supporto;

6. Infine, riassumere questi risultati sotto forma di un dizionario di dati al fine di poterli utilizzare come base per le analisi successive e per poter spiegare i risultati ottenuti.

Un buon punto di inizio per compiere una corretta esplorazione dei dati si basa sull'analisi delle caratteristiche dei dati. In R iniziamo questa analisi partendo col leggere il nostro dataset, ad esempio:

```
ticket <- read.csv("D:\\Giuseppe\\Download\\AnnualTicketSales.csv")
```

Successivamente, attraverso il comando *summary* o *str* saremo in grado di analizzare la struttura di un *dataframe*, ad esempio:

```
str(ticket)
```

```
> str(ticket)
'data.frame':   27 obs. of  6 variables:
 $ ï..YEAR                          : int  2021 2020 2019 2018 2017 2016 2015 2014 2013 2012 ...
 $ TICKETS.SOLD                     : chr  "42,37,74,881" "22,36,38,958" "1,22,85,41,629" "1,31,15,36,128" ...
 $ TOTAL.BOX.OFFICE                 : chr  "$3,881,777,912" "$2,048,534,616" "$11,253,443,955" "$11,948,096,650" ...
 $ TOTAL.INFLATION.ADJUSTED.BOX.OFFICE: chr  "$3,881,777,912" "$2,048,534,616" "$11,253,444,050" "$12,013,670,952" ...
 $ AVERAGE.TICKET.PRICE             : chr  "$9.16" "$9.16" "$9.16" "$9.11" ...
 $ X                                : logi  NA NA NA NA NA NA ...
```

Figura 2.1: Output del comando *str*

2.1 Sistemare i dati

Come possiamo notare dall'esempio, ci sono alcuni valori mancanti (NA) che potrebbero compromettere il nostro dataframe. Per procedere nelle analisi esplorative è fondamentale ripulire e sistematizzare i dati. Bisognerà, dunque, analizzarli al fine di verificare che la loro codifica sia corretta ed eventualmente controllare che la base dei dati non contenga errori rilevanti. Questa pulizia trasforma i dati tecnicamente corretti in dati consistenti. I dati sono tecnicamente corretti se :

- sono caricati in un dataframe (una collezione di oggetti generici del dataset), i cui nomi di colonne siano corretti e significativi

- ogni colonna è del tipo corretto. (Ad esempio: dati numerici in vettori, dati testuali in character ecc.)

Uno svantaggio sostanziale della pulizia dei dati è legato al fatto che non è facilmente automatizzabile, e dunque replicabile. Per ripulire le righe, l'approccio ottimale potrebbe essere quello di aprire il file con un editor di testo e andare a modificare tutte le eventuali virgole erroneamente interpretate come separatori di campo. Una volta ripulito, sarà sufficente ricaricare il dataset per verificare se il problema è stato risolto. Per ovviare al problema dei dati non disponibili (NA), invece, è necessario comprendere la causa dell'errore. Nella maggioranza dei casi, esso è dovuto ad una lacuna nel dataset. Se questi valori indisponibili non sono di grande importanza per la nostra analisi, sarà possibile ometterli attraverso il comando *na.omit*. Se l'assenza dei dati si verifica nell'intera colonna, essa potrà essere eliminata senza problemi.

2.2 Analisi Descrittiva

Una volta ripulito il datset si potrà procedere con l'analisi descrittiva. Questo processo analizza gli eventi passati al fine di poter interpretare i nuovi dati. Le finalità sono:

- avere una prima visione qualitativa delle variabili raccolte;
- controllare se sono presenti errori;
- evidenziare gli outlier e le anomalie;
- esprimere un giudizio in merito alla qualità delle ipotesi e determinare qualitativamente le relazioni tra variabili;
- individuare l'entità e la direzione delle relazioni fra le variabili;
- attuare i modelli statistici più appropriati.

In base alla tipologia delle variabili (categoriali, ordinali, quantitative) ed in base al numero di variabili prese in considerazione (univariate, bivariate, multivariate), vengono utilizzati strumenti diversi per l'analisi.

2.2.1 Tipi di variabili

Variabili categoriali

Le variabili categoriali sono le cosiddette variabili *qualitative* in quanto non derivano da misurazioni, ma da operazioni di classificazione e confronto. Attraverso l'uso delle tabelle di contingenza potremo rappresentare la distribuzione di frequenza di variabili categoriali e di fattori. Nel caso specifico di R, si usa la funzione *table(variabile)* per la rappresentazione univariata, e *table(variabile1, variabile2)* per la rappresentazione bivariata.

Tra i metodi grafici più comuni di rappresentazione di frequenza di una variabile categoriale o ordinale sono presenti i *barplot*.

```
1 barplot(table(ticket$TICKETS.SOLD, ticket$YEAR))
```

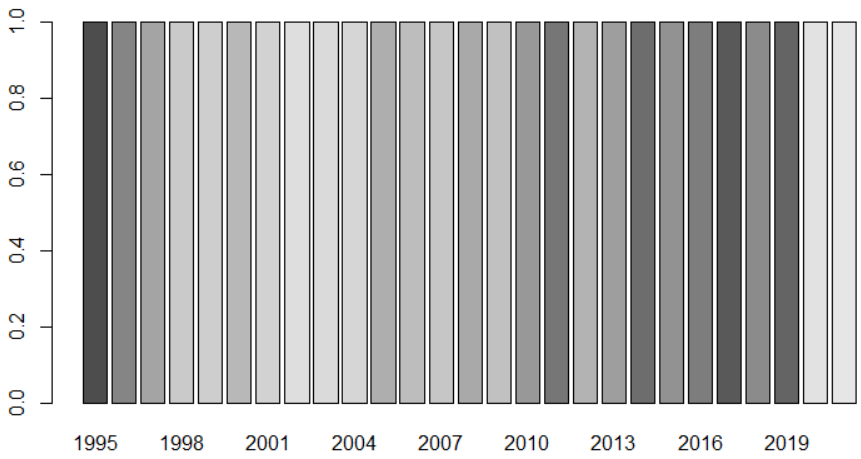


Figura 2.2: Esempio della funzione con il grafico relativo

Si potrebbe quindi procedere con il calcolo della moda. In R non esiste una funzione per calcolare la moda. Per calcolarla, si ricerca il valore massimo della tabella delle

frequenze.

Esempio:

```
# si crea la variabile nominale di 3 livelli
nominale <-factor(sample (c("rosso","verde","blu"), 100,replace = TRUE)
    )
# tabella delle frequenze
(frequenza<-table(nominale))

# nominale
#  blu rosso verde
#   31    38    31

# viene calcolata la moda
moda<-which(frequenza == max(frequenza))
names(frequenza)[moda]
barplot(frequenza)
```

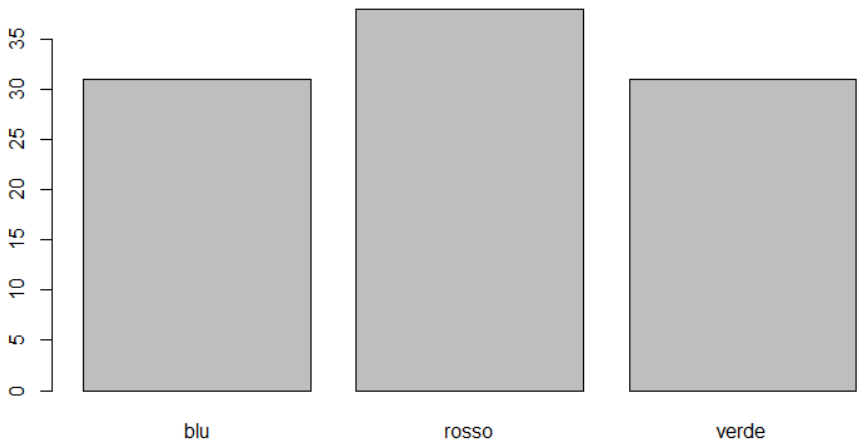


Figura 2.3: boxplot variabili categoriali

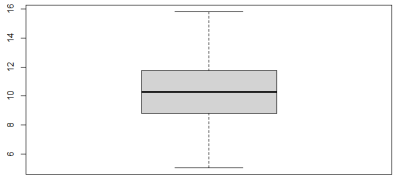
Variabili ordinali

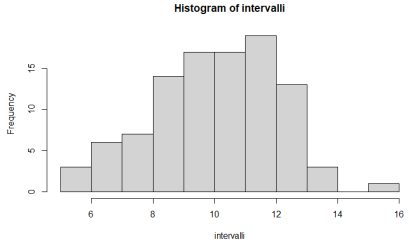
Le variabili ordinali sono rappresentate quando il carattere statistico assume stati discreti ma ordinabili. Un esempio di variabile ordinale è *Titolo di studio*, con tre modalità disposte in ordine crescente: licenza media inferiore, diploma e laurea. Gli indici che possono essere calcolati con questo tipo di variabile sono, oltre al numero di livelli e alla moda, anche il minimo, il massimo, la mediana, i quartili ed il range interquantile. In R vengono rappresentate come fattori ordinati.

Variabili a intervalli

Le variabili a intervalli possono assumere valori numerici che consentono confronti solo per differenza tra le modalità che le unità assumono. Oltre a mediana, moda, massimo, minimo e quartili possiamo calcolare la media e la varianza/deviazione standard. Per quanto riguarda il lato grafico è possibile utilizzare diverse funzioni come la funzione boxplot e la funzione *hist*, necessaria per rappresentare l'istogramma delle frequenze. Esempio:

```
# viene generata una variabile numerica con distribuzione normale
# 100 osservazioni, media=10, ds=2
intervalli <- rnorm(100, mean=10, sd=2)
summary(intervalli)
#    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   5.039   8.784  10.273  10.109  11.742  15.826
sd(intervalli)
## [1] 2.065185
boxplot(intervalli)
hist(intervalli)
```





2.2.2 La media

La funzione media è già integrata in R.
La media:

- viene applicata senza richiedere nessun parametro aggiuntivo;
- supporta parametri opzionali che la rendono più flessibile;
- non dovrebbe essere applicata quando un numero consistente di dati tende a discostarsi dagli altri;

Tra i parametri opzionali di più rilievo abbiamo *na.rm*, una variabile logica che specifica come i valori mancanti dovrebbero essere gestiti. Di default questo parametro booleano è *false*. Se così non fosse, la funzione calcolerebbe la media tenendo conto dei valori NA, restituendo quindi risultati non corretti.

2.2.3 La deviazione standard

Come la media, anche la deviazione standard, o scarto quadratico medio, è di semplice calcolo. Essa non fa altro che misurare quanto lontane le unità statistiche siano dalla media. In altre parole, sintetizza le deviazioni della media. Quest'ultima non è in grado di descrivere autonomamente la distribuzione di una variabile quantitativa in maniera adeguata. Sfortunatamente, la deviazione standard è ancora più sensibile ai valori anomali rispetto alla media. Come parametro opzionale possiede solo na.rm con la stessa interpretazione di default della moda, ovvero prendendo in considerazione i valori NA.
Esempio:

```
summary(Posti.letto)
# Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 1.00   19.00   49.50   60.25   90.00  314.00

# calcolo della deviazione standard
 sd(turismo20$Posti.letto, na.rm=TRUE)
#[1] 54.60374
```

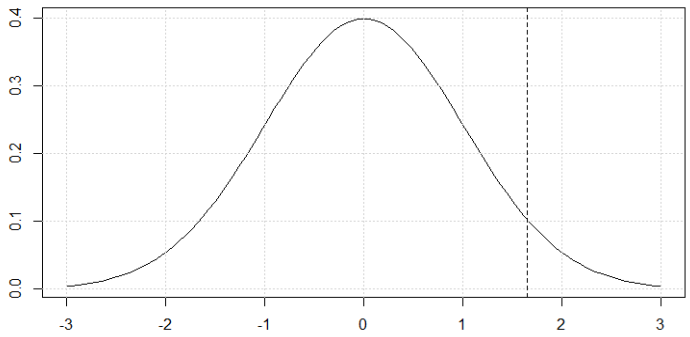
2.2.4 La distribuzione gaussiana

La distribuzione gaussiana o normale è una delle distribuzioni di probabilità più diffuse nei problemi. Essa aiuta a descrivere il comportamento delle variabili, ed è determinata da media e deviazione standard. Il vantaggio della distribuzione gaussiana è che permette di rendere determinate osservazioni più attendibili. Nonostante la popolarità di questa distribuzione, non è sempre ragionevole applicarla, per diversi motivi. La questione è di fatto semplice: come possiamo stabilire se l'ipotesi gaussiana è ragionevole oppure no? Esistono tre grafici che ci consentono di dare una risposta:

1. Istogrammi, probabilmente i più conosciuti ma i meno efficaci;
2. Grafici di densità, ovvero degli istogrammi smussati che permettono la rappresentazione della distribuzione;
3. Grafici quantili-quantili (QQ), sicuramente meno conosciuti ma più efficaci.

Esempio di distribuzione gaussiana:

```
# Il grafico che segue rappresenta una distribuzione normale, mentre la
    retta indica il quantile 1,65
curve(dnorm(x), -3, +3,
       ylab = "", xlab = "")
grid()
abline(v = 1.65, lty = 2)
```



2.3 Anomalie nei dati

All'interno dei dati, durante la visualizzazione, possono essere presenti delle osservazioni in grado di creare problemi, in quanto possono influenzare parametri e stime. Esse vengono identificate come *outlier* e possono essere generate da diverse cause, in base a come i dati sono trattati durante l'analisi:

- Non fanno parte dell'insieme dei dati che si vogliono analizzare, ad esempio un errore di misura, oppure fanno parte dei dati dell'esperimento ma sono creati da un dispositivo malfunzionante. In questo caso è necessario eliminare gli outlier prima di procedere con l'analisi dei dati.
- Possono essere errori tipografici o di scrittura; in questo caso vengono individuati dopo una rilettura del file, corretti e successivamente inseriti nell'insieme da analizzare.
- Potrebbero indicare una tendenza non prevista dell'oggetto da analizzare: ci andrebbero a segnalare un comportamento nuovo, a volte molto interessante. In questi casi l'outlier viene considerato un elemento importante e quindi viene assunto nell'insieme dei dati da analizzare al fine di contribuire all'analisi.

2.3.1 Rilevamento dei valori anomali

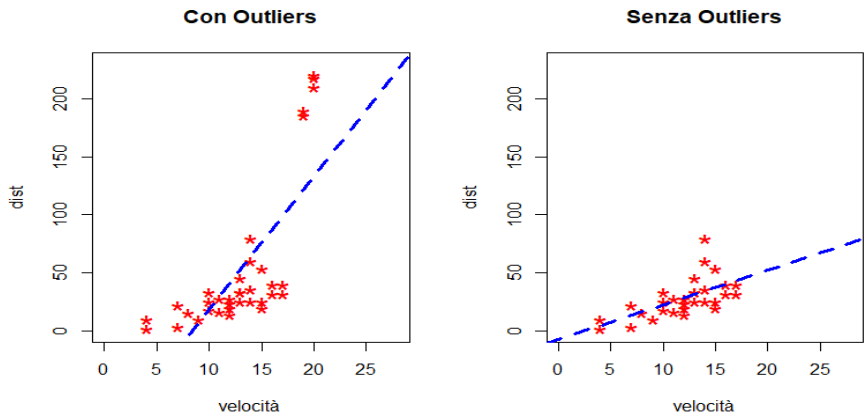
Per quanto riguarda la rilevazione degli outlier, i grafici della dispersione sono utili al fine di individuarli, a meno che non si tratti di outlier gravi e non frequenti. In questo caso potrebbero essere difficili da individuare. Prima di effettuare qualsiasi scelta, bisogna sempre analizzare gli outlier per poter valutare se vi sono motivi per considerarli errori oppure osservazioni da collocare all'interno dei nostri dati. Per definizione:

- Si definisce outlier **potenziale** un'osservazione che si trova ad una distanza dal centro superiore a 1.5 volte l'ampiezza dell'intervallo $[h_L; h_U]$, cioè che appartiene all'intervallo $[h_L - 1.5(h_U - h_L), h_U + 1.5(h_U - h_L)]$.

- Si definisce outlier **sospetto** un'osservazione che si trova ad una distanza dal centro superiore a 3 volte rispetto all'ampiezza dell'intervallo$[h_L; h_U]$, cioè che appartiene all'intervallo $[h_L - 3(h_U - h_L), h_U + 3(h_U - h_L)]$.

Di seguito è riportato un esempio di outlier: [7]

```
# vengono introdotti gli outlier nel dataframe
# dati originali
cars1 <- cars[1:30,]
# vengono aggiunti gli outliers.
cars_outliers <- data.frame(speed=c(19,19,20,20,20), dist=c(190, 186,
    210, 220, 218))
# si uniscono ai dati originali gli outliers attraverso la funzione
    rbind.
cars2 <- rbind(cars1, cars_outliers)
# viene eseguito il plot dei dati con gli outliers
par(mfrow=c(1, 2)) # serve per inserire piu' grafici nello stessa area
plot(cars2$speed, cars2$dist, xlim=c(0, 28), ylim=c(0, 230), main="Con
    Outliers", xlab="speed", ylab="dist", pch="*", col="red", cex=2)
abline(lm(dist ~ speed, data=cars2), col="blue", lwd=3, lty=2)
# viene eseguito il plot dei dati originali senza gli outliers.
plot(cars1$speed, cars1$dist, xlim=c(0, 28), ylim=c(0, 230), main="
    Senza Outliers \n A much better fit!", xlab="speed", ylab="dist",
    pch="*", col="red", cex=2)
abline(lm(dist ~ speed, data=cars1), col="blue", lwd=3, lty=2)
```



Nella figura è possibile notare un netto miglioramento della pendenza (angolo) della linea dopo aver rimosso gli outlier. Se avessimo usato invece questi valori anomali, le nostre previsioni sarebbero state esagerate per i valori della velocità a causa di una maggiore pendenza.

2.4 Visualizzazione grafica dei dati

Come accennato in precedenza, per una corretta analisi dei dati, è necessario utilizzare strumenti in grado di rappresentare correttamente le analisi svolte. I grafici costituiscono una rappresentazione visiva di un insieme di informazioni. Uno dei vantaggi della rappresentazione grafica è dato dallo sfruttamento di determinate caratteristiche del sistema visivo:

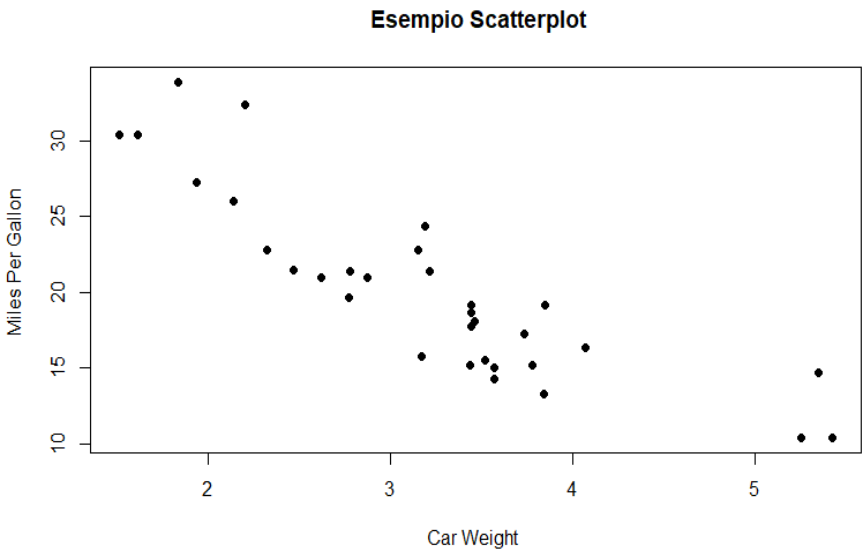
- la capacità di elaborare un'ampia quantità dei dati;
- la capacità di identificare distribuzioni e relazioni fra i dati;
- la capacità di percepire l'allineamento, orizzontale e verticale, degli elementi grafici;
- la capacità di stimare, anche se approssimativamente, differenze di lunghezza, colore e di forma degli elementi.

Un grafico ottimo è quello che riesce a trasmettere una giusta quantità di informazioni in maniera adeguata e di facile comprensione. Permette di far emergere distribuzioni, pattern, relazioni tra variabili, presenza di outlier e relazioni fra osservazioni; inoltre consente di stimare, seppur approssimativamente, il valore delle singole osservazioni.

2.4.1 Scatterplot

Per visualizzare le relazioni tra variabili, la migliore rappresentazione è sicuramente il grafico di dispersione, anche noto come *scatterplot*.
Esempio:

```
# attraverso attach si accede alle variabili presenti nel dataframe
    contenuti in R
attach(mtcars)
plot(wt, mpg, main="Esempio Scatterplot",
    xlab="Car Weight ", ylab="Miles Per Gallon ", pch=19)
```



Con *pch* si specifica il simbolo da utilizzare nel grafico, mentre si assegna l'etichetta sugli assi con *xlab* per l'asse orizzontale e *ylab* per l'asse verticale.

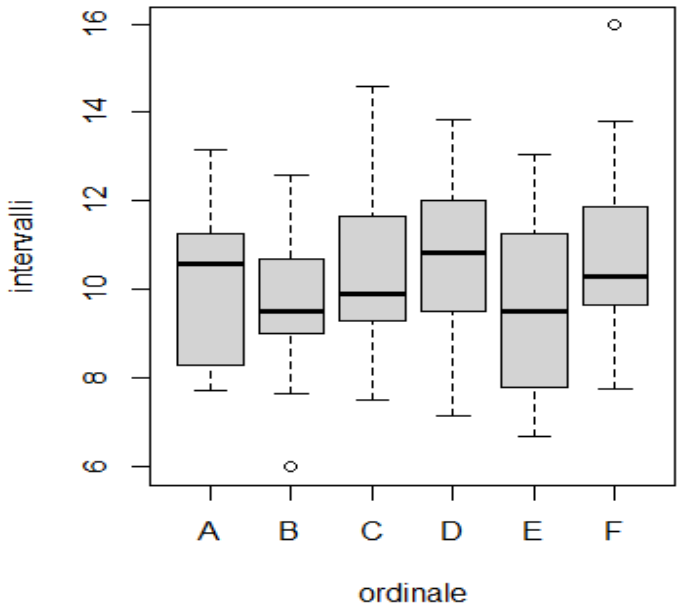
2.4.2 Boxplot

Il boxplot, insieme all'istogramma, è uno dei tipi di grafico più utilizzati per la rappresentazione delle variabili quantitative. Permette di effettuare il confronto tra una variabile

ad intervalli e una variabile categoriale (nominale o ordinale). Esso confronta la forma di più distribuzioni, ma soprattutto ci consente di identificare in modo rapido e preciso gli outlier.

Esempio:

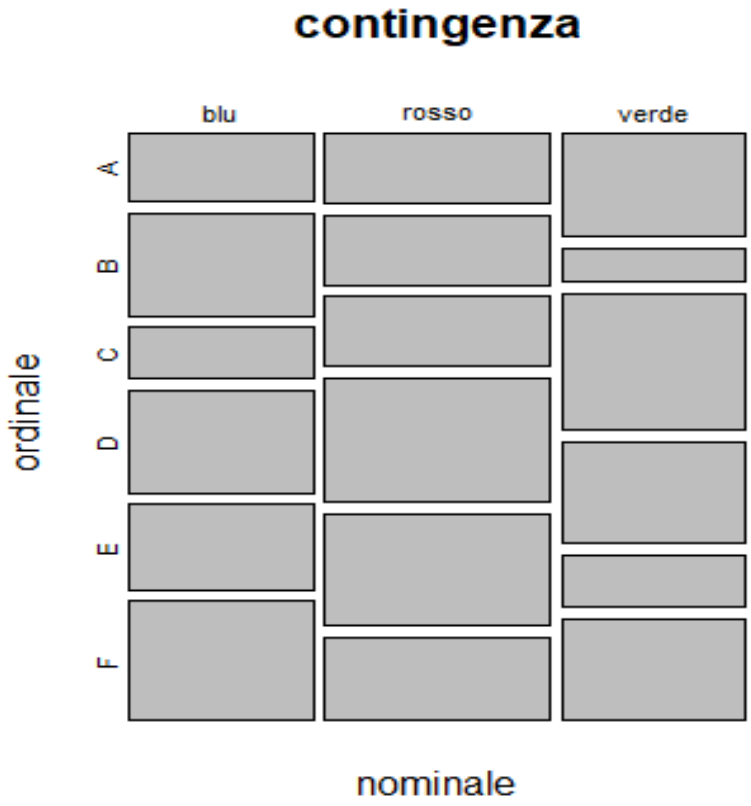
```
# viene generata una variabile numerica con distribuzione normale
# 100 osservazioni, media=10, sd=2
intervalli <- rnorm(100, mean=10, sd=2)

# viene creata una variabile ordinale con 6 livelli
cat_ord <- c("A","B","C","D","E" "F")
ordinale <- factor(sample (cat_ord, 100,replace = TRUE),levels = cat_
    ord)

# viene eseguito il plot
boxplot(intervalli ~ ordinale)
```



2.4.3 Mosaicplot

Il *mosaicplot* serve principalmente per rappresentare il rapporto tra due variabili categoriali (nominali o ordinali).
Esempio:

```
# viene creata variabile nominale
nominale <-factor(sample (c("rosso","verde","blu"), 100,replace = TRUE)
    )

# viene creata variabile ordinale con 6 livelli
cat_ord <- c("A","B","C","D","E" "F")
ordinale <- factor(sample (cat_ord, 100,replace = TRUE),levels = cat_
    ord)

# viene creata la tabella
(tabcontingenza <- table(nominale, ordinale))
#         ordinale
#nominale A B C D E F
#    blu   4 6 3 6 5 7
#    rosso 5 5 5 9 8 6
#    verde 6 2 8 6 3 6

# viene eseguito il plot
mosaicplot(tabcontingenza)
```

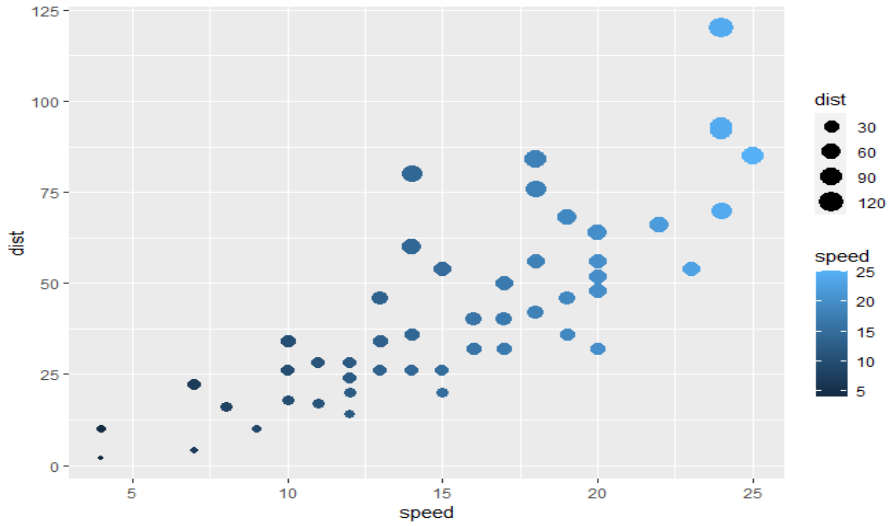


2.4.4 Libreria ggplot2

Ggplot2 è una libreria di R per la produzione di grafici relativa ai dati. A differenza della libreria dei grafici di base, ggplot2 prevede componenti separati che possono essere combinati in diversi modi. Per iniziare ad usare ggplot2 è necessario ricorrere alla funzione *qplot()* per creare e personalizzare i primi grafici "standard". Il funzionamento di qplot() è simile a quello di plot(). I grafici, in questo caso, sono completamente personalizzabili in modo estremamente flessibile.

Tuttavia, a differenza dei grafici standard, i ggplot2 potrebbero essere complicati da realizzare, e non sempre vale la pena utilizzarli. Infatti, a differenza dei grafici standard, ggplot2 utilizza propri parametri e una propria sintassi. Il punto di forza di questa libreria è una maggiore flessibilità nella creazione dei grafici, i quali risultano anche più gradevoli dal punto di vista estetico. Ad esempio, i grafici a bolle generati con ggplot2, sono in pratica una variazione dei grafici a dispersione (scatterplot) prodotti con `geom_point()`; Di seguito il codice:

```
# viene eseguito il plot
ggplot(cars,aes(speed, dist)) + geom_point(aes(size = dist, col = speed
    ))
```



Capitolo 3

La regressione

In questo capitolo si introduce uno degli argomenti più importanti dal punto di vista applicativo, ovvero il modello di regressione lineare. Esso si pone l'obiettivo di costruire un modello attraverso cui prevedere i valori di una variabile dipendente o quantitativa a partire dai valori di una o più variabili indipendenti e successivamente effettuare, attraverso una correlazione, lo studio dell'associazione tra queste variabili.

3.1 Regressione lineare semplice

[5] Il modello di regressione lineare semplice è definito come

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i, i = 1, ..., n,$$

dove (x_i, y_i) rappresentano le coppie di osservazioni facenti parte del campione, β_0 e β_1 sono le incognite che si suppone uniscano le variabili X e Y nella popolazione, mentre ϵ_i corrisponde all'errore casuale che intuitivamente descrive l'informazione relativa a Y, spiegabile attraverso una funzione lineare di X. Questo modello è definito anche dalle seguenti ipotesi:

1. Le osservazioni x_i sono costanti o realizzazioni di una variabile aleatoria X non correlata con le componenti aleatorie di errore ϵ_i. In questo caso l'inferenza è svolta condizionatamente ai valori osservati di X.

2. Gli errori aleatori ϵ_i sono variabili aleatorie con media 0 e varianza σ^2 costante per ogni $i = 1, ..., n$
3. Gli errori aleatori ϵ_i non sono correlati tra loro
4. Gli errori aleatori ϵ_i sono distribuiti secondo una distribuzione normale

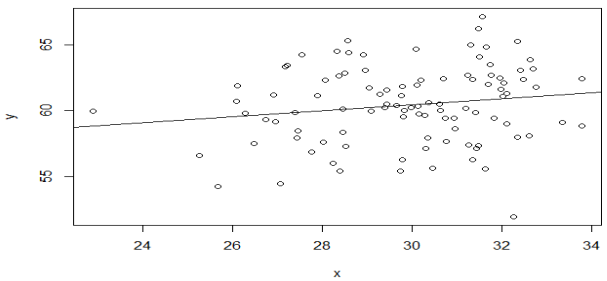
La funzione per stimare un modello di regressione lineare in R è *lm() (linear model)* . La sintassi usata è dunque $lm(yx)$, dove y è la variabile dipendente e x è la variabile indipendente. Una volta chiamata la funzione, verrà restituita una lista contenente un gran numero di risultati relativi al modello scelto. Questa funzione produce un output composto dalle seguenti parti:

- La prima riga (*Call*) riporta il codice che è stato eseguito;
- La seconda parte (*Residuals*) mostra una sintesi dei residui del modello;
- La terza parte (*Coefficients*) fornisce le stime dei coefficienti del modello insieme ai *p-value* relativi;
- La quarta ed ultima parte contiene alcuni indici, utili per valutare la bontà complessiva del modello. In particolare, la stima di σ (indicata come *Residual standard error*), l'indice di R^2 (riportato come *Multiple R-squared*), ed infine il test F.

Per disegnare la retta di regressione, si passa il risultato di lm() alla funzione *abline* che disegna una linea con parametri a e b.
Esempio in R:

```
# vengono generate due variabili casuali indipendenti
x <- rnorm (100,30,2)
y <- rnorm (100,60,3)

# viene applicata la formula
modello <- lm(y~x)

# vengono mostrate le informazioni
summary(modello)
```

```
10
11 ##Call:
12 #lm(formula = y ~ x)
13
14 #Residuals:
15 #     Min       1Q   Median       3Q      Max
16 #-9.0489 -2.0605   0.0593   1.8578   6.3735
17
18 #Coefficients:
19 #             Estimate Std. Error t value Pr(>|t|)
20 #(Intercept)  53.6739     4.1334   12.985   <2e-16 ***
21 #x             0.2259     0.1380    1.638    0.105
22 #---
23 #Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .
   0.1          1
24
25 #Residual standard error: 2.896 on 98 degrees of freedom
26 #Multiple R-squared:  0.02664,  Adjusted R-squared: 0.01671
27 #F-statistic: 2.682 on 1 and 98 DF,  p-value: 0.1047
28
29 # vengono mostrate le informazioni
30 summary(modello)
31
32 # viene eseguito il plot
33 plot (x,y)
34 abline (modello)
```



3.2 Overfitting e split dei dati

L'esempio appena descritto adotta l'utilizzo di pratiche standard per valutare la bontà del modello ed è costituito da dati semplici. Un problema legato alla regressione è l'*overfitting*. Esso si verifica quando il nostro modello offre una precisione del dataset elevata, ma presenta prestazioni scadenti su un dataset non osservato. Per cercare di risolvere questo problema si deve eseguire una suddivisione dei dati. Principalmente esso consiste nell'utilizzare un sottoinsieme di dati per costruire il nostro modello e un altro, diverso ma simile, per valutarlo. In generale l'approccio corretto è quello di suddividere casualmente il nostro dataset in tre sottoinsiemi che si escludono a vicenda:

- un *Train set* utilizzato per l'adattamento dei parametri al modello;
- un *Validation set* utilizzato per prendere decisioni inerenti alla struttura del modello;
- un *Test set* utilizzato per la valutazione del modello finale.

3.2.1 Train set

Nel cosiddetto Train set verranno inseriti dei dati che saranno utilizzati esclusivamente per "allenare" il modello, ciò sta a significare che il nostro modello imparerà tutte le relazioni rappresentate dalle nostre variabili. Il funzionamento generale, dunque, consiste nell'osservare, analizzare e studiare questi dati, per poi effettuare delle previsioni in base allo studio necessario. Esso avviene attraverso il confronto tra il risultato della predizione reale con quello ottenuto da questo modello, aggiornando i vari iperparametri al fine di ridurre al minimo l'errore rispetto all'iterazione precedente.

3.2.2 Validation set

Per avere una reale capacità predittiva e quindi risolvere problematiche relative all'overfitting, facciamo riferimento al validation set, in quanto si occupa di validare i risultati ottenuti nel training set. Costruiamo questo modello inserendo dei dati che non sono mai stati osservati e iniziamo la previsione. Essi devono necessariamente contenere le informazioni nello stesso formato adottato dal train set. Infine si andranno a confrontare i dati predetti con quelli reali per vedere quanto il nostro modello riesce a prevedere con un'ottima approssimazione la nostra variabile di output. Se le performance ottenute non sono soddisfacenti, si dovrà andare a modificare i parametri del modello per poi ripartire con il train test, finchè non si otterrà un risultato ottimale.

3.2.3 Test set

Quando il nostro modello riuscirà ad ottenere delle buone performance sia sul train test che sul validation test, esso potrà essere testato su altre osservazioni che il modello non ha mai osservato attraverso il train test. Questo set serve solamente per visualizzare le prestazioni del nostro modello e il relativo funzionamento.

3.3 Calcolare le previsioni di un modello di regressione lineare

Il modello della regressione lineare, come detto in precedenza, si basa sul calcolo di previsioni in relazione a scenari non necessariamente analizzati nel campione. In R, utilizzando la funzione *predict()*, sarà possibile effettuare tale calcolo.
I parametri richiesti dalla funzione sono quattro:

- **object**, l'oggetto restituito dalla funzione lm() popolato al suo interno dai risultati della stima;

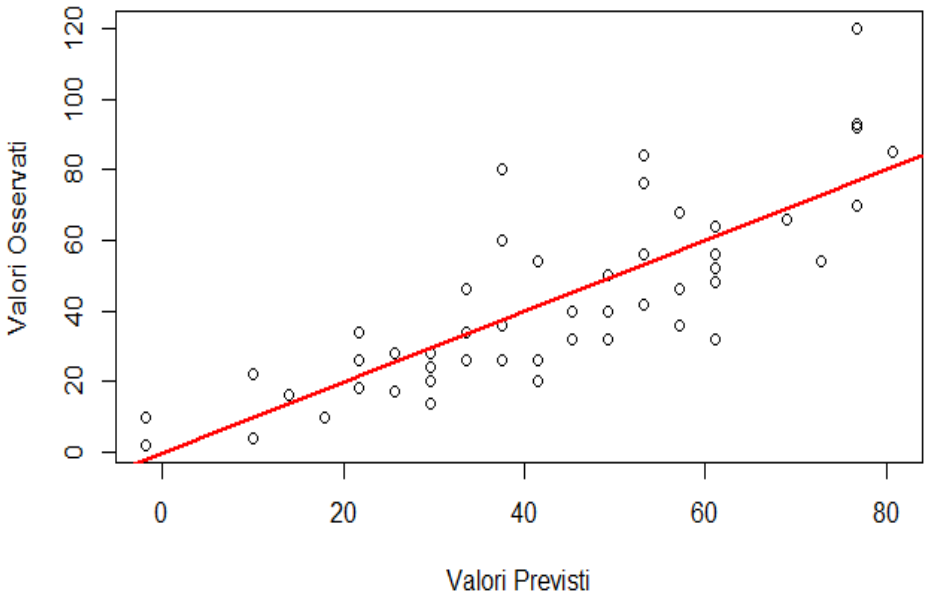
- **newdata**, che mostra un nuovo dataframe contenente tutti i valori delle variabili indipendenti da utilizzare per il calcolo delle previsioni;
- **interval**, che può assumere tre valori:
  - confidence, se si desidera calcolare gli intervalli di confidenza per la previsione del valore;
  - prediction, se si desidera calcolare gli intervalli di previsione per i valori individuali;
  - none, se non si vuole calcolare nessun intervallo.
- **level**, attraverso cui viene specificato il livello di confidenza per il calcolo o la previsione degli intervalli di confidenza.

L'output che viene mostrato rappresenta una matrice con due righe che stanno ad indicare gli scenari considerati e colonne contenenti rispettivamente le previsioni (*fit*), e i limiti degli intervalli di previsione (*lwr* e *upr*).
Esempio di previsione in R:

```
# viene creato il modello di regressione lineare, per semplicita'
   utilizzo il dataset "cars" contenuto in R
lreg <- lm(dist ~ speed, data = cars)
lreg

#Call:
#lm(formula = dist ~ speed, data = cars)

#Coefficients:
#(Intercept)        speed
#    -17.579        3.932

# si prova a prevedere la distanza di arresto per un nuovo valore della
    velocita'

# viene creato un nuovo dataframe contenente tre nuovi valori di
   velocita' di esempio
```

```r
new_speeds <- data.frame(speed = c(12, 18, 23))

# viene utilizzata la funzione predict
predict(lreg, newdata = new_speeds)
#    1        2        3
#29.60981 53.20426 72.86631
```

```r
# viene generato il plot per analizzare i valori predetti attraverso i
    valori osservati
plot(predict(lreg),
     cars$dist,
     xlab = "Valori Previsti",
     ylab = "Valori Osservati")
abline(a = 0,
       b = 1,
       col = "red",
       lwd = 2)
```



3.4 R-quadro

Dopo aver impostato un modello di regressione lineare, è necessario determinare come esso si adatti ai dati. Per questa analisi dovremo utilizzare il coefficiente di determinazione noto come R-quadro. Esso misura quanto le singole osservazioni si allontanino dalla retta di regressione sullo stesso dataset. R-quadro è dunque la variazione percentuale della variabile dipendente illustrata da un modello lineare. [3]

$$R^2 = \frac{Varianza\ illustrata\ dal\ modello}{Varianza\ totale}$$

R-quadro è sempre compreso tra 0 e 100%:

- 0% rappresenta un modello che non riesce a spiegare nessuna delle variazioni all'interno della variabile ,ovvero indica che non esiste alcuna correlazione
- 100% rappresenta un modello che riesce a spiegare tutte le variazioni della variabile, ovvero indica che esiste una correlazione perfetta.

Solitamente, più è grande il valore di R-quadro, più il modello di regressione si adatta alle nostre osservazioni. Non è possibile, però, utilizzare R-quadro per determinare se le stime dei coefficienti e le previsioni siano irregolari, motivo per cui è necessario valutare e analizzare i grafici dei residui. R-quadro non indica dunque se un modello di regressione fornisce un adattamento conforme ai dati. Non necessariamente un buon modello deve avere un valore R-quadro elevato, di conseguenza un modello buono potrebbe avere un valore R-quadro basso.

3.4.1 Visualizzazione grafica di R-quadro

Per dimostrare graficamente i valori di R-quadro è possibile tracciare le osservazioni con la retta adattata che rappresenta l'equazione di regressione. Le relazioni più forti indicano una dispersione più bassa. A differenza di un coefficiente di correlazione, R-quadro non indica la direzione della relazione. Per determinare i punti, bisogna controllare i coefficienti di regressione. [3]

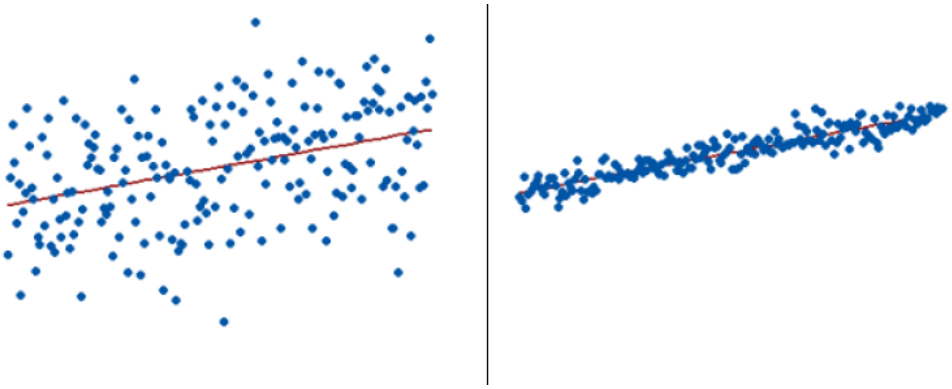


Figura 3.1: i due tipi di regressione

Entrambi i grafici utilizzano la stessa scala in modo da poter confrontare la dispersione. Per il modello sinistra il valore di R-quadro è del 15% mentre per quello a destra il valore è pari al 85%. Quando un modello di regressione tiene conto di una parte maggiore della varianza, i punti dati sono più vicini alla retta di regressione. Dunque per un R-quadro del 100%, i valori adattati sono uguali ai valori dei dati e, conseguentemente, tutte le osservazioni combaciano esattamente sulla retta di regressione.

Capitolo 4

Caso studio: Previsione del prezzo di uno smartphone

In questo capitolo è stato effettuato uno studio su un dataset presente su Kaggle in merito al prezzo di uno smartphone. Innanzitutto è necessario porsi una domanda: **Quali sono le caratteristiche più importanti per prevedere il prezzo di uno smartphone?** L'obiettivo è dunque scoprire quali sono le relazioni tra le caratteristiche di uno smartphone (ad esempio: RAM, memoria interna) e il suo prezzo di vendita. In seguito vengono applicate tutte le analisi.

Descrizione dei dati

Di seguito la descrizione dei dati:

- ID: Identificativo
- battery_power: Capienza totale della batteria in mAh
- blue: Possiede o no il *Bluetooth*
- clock_speed: velocità di clock del processore
- dual_sim: Dispone di due slot SIM o meno

- fc: Megapixel della fotocamera frontale
- four_g: Dispone di connettività 4G
- int_memory: Memoria interna espressa in Gigabyte
- m_dep: Profondità in centimetri
- mobile_wt: Peso del dispositivo
- n_cores: Numero di core del processore
- pc: Megapixel della fotocamera principale
- px_height: Risoluzione in altezza dello schermo
- px_width: Risoluzione in larghezza dello schermo
- ram: RAM espressa in Megabyte
- sc_h: Altezza del telefono espressa in centimetri
- sc_w: Larghezza del telefono espressa in centimetri
- talk_time: Durata massima della batteria durante una chiamata
- three_g: Dispone o no di connettività 3G
- touch_screen: Dispone o no di uno schermo touchscreen
- wifi: Dispone o no di un'antenna Wi-Fi

La varibile **target** indica la fascia di prezzo caratterizzata dai seguenti valori:

- 0 - costo basso.
- 1 - costo medio.
- 2 - costo elevato.
- 3 - costo molto elevato.

4.1 Analisi esplorativa dei dati

4.1.1 Preelaborazione dei dati

Come prima cosa vengono importate le librerie necessarie per le analisi e successivamente verrà caricato il dataset:

```
library(ggplot2)
library(rpart)
library('class')
library(RColorBrewer)
library(e1071)
# leggo il dataset
mobile <- read.csv("D:\\Giuseppe\\Universita\\3 anno\\TESI\\dataset
   mobile price\\mobile.csv")
```

Vengono eseguiti i comandi *dim* e *summary* per iniziare a familiarizzare con i dati.

```

dim(mobile)# viene osservata la dimensione del dataset
#[1] 2000   21
summary(mobile)# viene analizzata la struttura del dataframe

# battery_power          blue         clock_speed
# Min.   : 501.0   Min.   :0.000   Min.   :0.500
# 1st Qu.: 851.8   1st Qu.:0.000   1st Qu.:0.700
# Median :1226.0   Median :0.000   Median :1.500
# Mean   :1238.5   Mean   :0.495   Mean   :1.522
# 3rd Qu.:1615.2   3rd Qu.:1.000   3rd Qu.:2.200
# Max.   :1998.0   Max.   :1.000   Max.   :3.000
#    dual_sim            fc              four_g
# Min.   :0.0000   Min.   : 0.000   Min.   :0.0000
# 1st Qu.:0.0000   1st Qu.: 1.000   1st Qu.:0.0000
# Median :1.0000   Median : 3.000   Median :1.0000
# Mean   :0.5095   Mean   : 4.309   Mean   :0.5215
# 3rd Qu.:1.0000   3rd Qu.: 7.000   3rd Qu.:1.0000
# Max.   :1.0000   Max.   :19.000   Max.   :1.0000
#   int_memory        m_dep           mobile_wt
```

```
21 # Min.   : 2.00   Min.   :0.1000   Min.   : 80.0
22 # 1st Qu.:16.00   1st Qu.:0.2000   1st Qu.:109.0
23 # Median :32.00   Median :0.5000   Median :141.0
24 # Mean   :32.05   Mean   :0.5018   Mean   :140.2
25 # 3rd Qu.:48.00   3rd Qu.:0.8000   3rd Qu.:170.0
26 # Max.   :64.00   Max.   :1.0000   Max.   :200.0
27 #    n_cores            pc           px_height
28 # Min.   :1.000   Min.   : 0.000   Min.   :   0.0
29 # 1st Qu.:3.000   1st Qu.: 5.000   1st Qu.: 282.8
30 # Median :4.000   Median :10.000   Median : 564.0
31 # Mean   :4.521   Mean   : 9.916   Mean   : 645.1
32 # 3rd Qu.:7.000   3rd Qu.:15.000   3rd Qu.: 947.2
33 # Max.   :8.000   Max.   :20.000   Max.   :1960.0
34
35 #    px_width           ram            sc_h
36 # Min.   : 500.0   Min.   : 256   Min.   : 5.00
37 # 1st Qu.: 874.8   1st Qu.:1208   1st Qu.: 9.00
38 # Median :1247.0   Median :2146   Median :12.00
39 # Mean   :1251.5   Mean   :2124   Mean   :12.31
40 # 3rd Qu.:1633.0   3rd Qu.:3064   3rd Qu.:16.00
41 # Max.   :1998.0   Max.   :3998   Max.   :19.00
42 #      sc_w          talk_time        three_g
43 # Min.   : 0.000   Min.   : 2.00   Min.   :0.0000
44 # 1st Qu.: 2.000   1st Qu.: 6.00   1st Qu.:1.0000
45 # Median : 5.000   Median :11.00   Median :1.0000
46 # Mean   : 5.767   Mean   :11.01   Mean   :0.7615
47 # 3rd Qu.: 9.000   3rd Qu.:16.00   3rd Qu.:1.0000
48 # Max.   :18.000   Max.   :20.00   Max.   :1.0000
49 #  touch_screen        wifi        price_range
50 # Min.   :0.000   Min.   :0.000   Min.   :0.00
51 # 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.75
52 # Median :1.000   Median :1.000   Median :1.50
53 # Mean   :0.503   Mean   :0.507   Mean   :1.50
54 # 3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:2.25
55 # Max.   :1.000   Max.   :1.000   Max.   :3.00
```

4.1.2 Pulizia dei dati

Adesso è necessario ripulire il dataset di tutti quei valori mancanti che potrebbero successivamente causare problemi. Inoltre viene effettuato un controllo ulteriore attraverso la funzione *duplicated* al fine di verificare che non ci siano dati duplicati dopo la pulizia.

```
mobile = na.omit(mobile)
duplicated(mobile)
```

Questa funzione restituirà un elenco di valori settati uguale a *false* per indicare che non esiste alcun duplicato per quel valore.

4.1.3 Visualizzazione dei dati

Innanzitutto, per rispondere al quesito è necessario analizzare la relazione tra le caratteristiche e la variabile target che in questo caso è la fascia di prezzo.

```
# viene visualizzata la relazione tra le features e la nostra variabile
cor(mobile)

# viene eseguito il plot della matrice di correlazione
corrplot(cor(mobile), type='upper', order='hclust', col=brewer.pal(n=8,
    name="RdBu"))
```

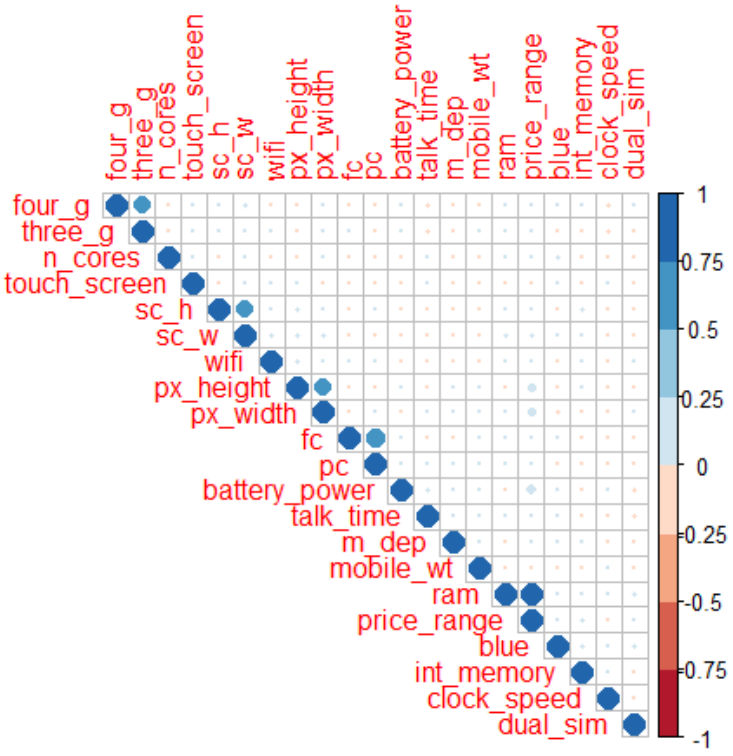
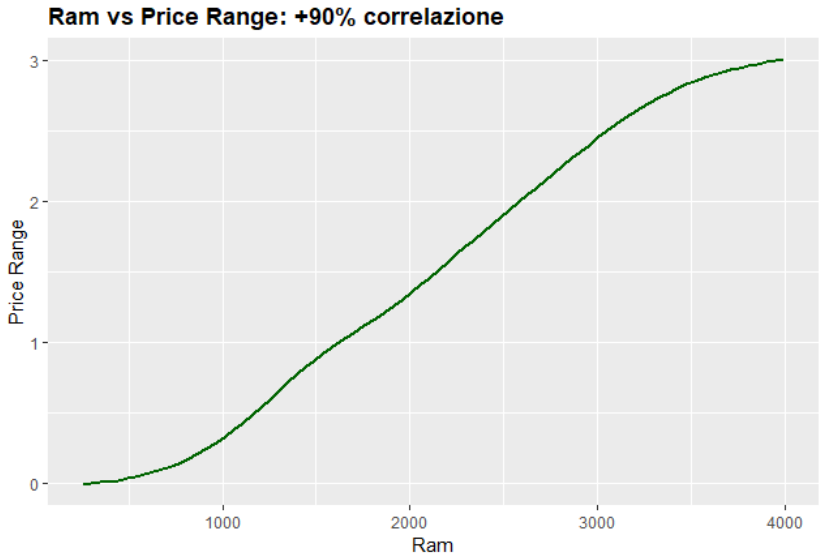


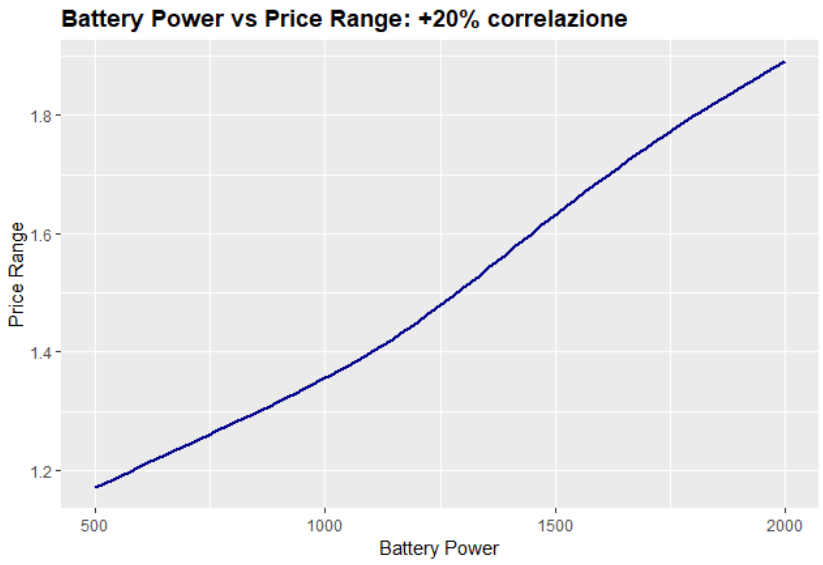
Figura 4.1: Matrice di correlazione

Da questa visualizzazione all'interno della matrice di correlazione si evince che le due caratteristiche altamente correlate sono la **fascia di prezzo** e la **RAM**. Possiamo notare che anche la potenza della batteria e l'altezza dei pixel sono leggermente correlate alla fascia di prezzo. Viene eseguito il plot dei grafici con le tre variabili della correlazione:

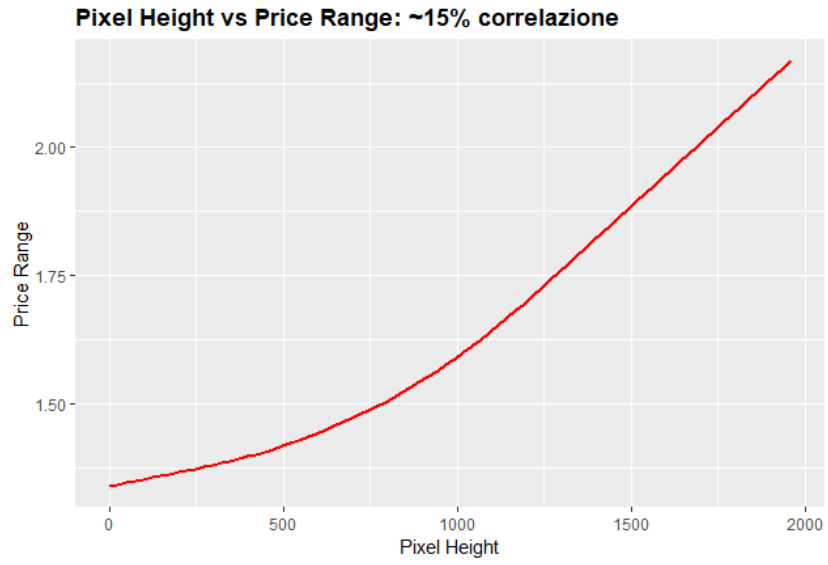
```r
# con la RAM
ggplot(data = mobile, mapping=aes(x=ram, y=price_range)) +
  geom_smooth(se=FALSE, color='darkgreen', method='gam', formula = y ~
     s(x, bs = "cs")) +
  ggtitle('Ram vs Price Range: +90% correlazione') +
  theme(plot.title = element_text(face="bold")) +
  labs(x = 'Ram', y='Price Range')
```




```r
# con la batteria
ggplot(data = mobile, mapping=aes(x=battery_power, y=price_range)) +
  geom_smooth(se=FALSE, color='darkblue', method='gam', formula = y ~ s
     (x, bs = "cs")) +
  ggtitle('Battery Power v. Price Range: +20% correlazione') +
  theme(plot.title = element_text(face="bold")) +
  labs(x = 'Battery Power', y='Price Range')
```




```
# con l'altezza dei pixel
ggplot(data = mobile, mapping=aes(x=px_height, y=price_range)) +
  geom_smooth(se=FALSE, color='red', method='gam', formula = y ~ s(x,
      bs = "cs")) +
  ggtitle('Pixel Height v. Price Range: ~15% correlation') +
  theme(plot.title = element_text(face="bold")) +
  labs(x = 'Pixel Height', y='Price Range')
```



```r
# viene converitita la varibile price range in fattore al fine di
    raggruppare le  osservazioni per la fascia di prezzo
mobile$price_range <- as.factor(mobile$price_range)

# boxplot
p1 <- ggplot(mobile, aes(x=price_range, y = ram, color=price_range)) +
  geom_boxplot(outlier.colour="red", outlier.shape=8,
               outlier.size=4) +
  labs(title = "RAM vs Price Range")

p2 <-  ggplot(mobile, aes(x=price_range, y = battery_power, color=price
    _range)) +
  geom_boxplot(outlier.colour="red", outlier.shape=8,
               outlier.size=4) +
  labs(title = "Battery Power vs Price Range")

p3 <- ggplot(mobile, aes(x=price_range, y = px_height, color=price_
    range)) +
  geom_boxplot(outlier.colour="red", outlier.shape=8,
               outlier.size=4) +
  labs(title = "Pixel Height vs Price Range")
grid.arrange(p1, p2, p3, nrow = 1)
```

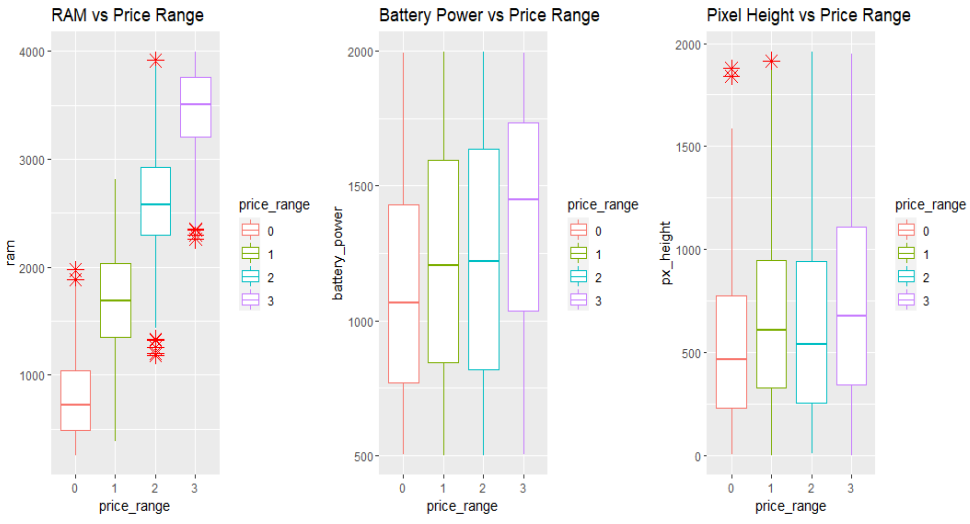


Figura 4.2: boxplot

4.2 Regressione Lineare

Prima di effettuare la regressione lineare è necessario evitare l'overfitting eseguendo lo split dei dati suddividendoli in train set e test set; vengono divisi affinchè gli intervalli dei valori per la nostra variabile y abbiano la stessa proporzione in entrambi gli insiemi.

```
# viene convertita la variabile price range nuovamente in numerica
mobile$price_range <- as.numeric(mobile$price_range)

# split dei dati per evitare overfitting
set.seed(125)#per garantire la replicabilit  dei risultati
split<- sample(c(rep(0, 0.7 * nrow(mobile)), rep(1, 0.3 * nrow(mobile))
   ))
split
table(split)

# split
#   0    1
#1400  600

# possiamo notare che 1400 osservazioni sono state assegnate al train
   set mentre 600 al test set
mobile.train <- mobile[split == 0,]
mobile.test <- mobile[split == 1,]
summary(mobile.train)
str(mobile.train)
dim(mobile.train)

#[1] 1400   21
```

Attraverso i comandi *summary* e *str* vengono rianalizzate le osservazioni proseguendo con l'utilizzo della regressione lineare al fine di tentare di prevedere la fascia di prezzo. In primo luogo verrà utilizzata la variabile più correlata alla fascia di prezzo, ovvero la RAM.

```
lreg = lm(price_range~ram, data=mobile.train)
summary(lreg)
```

```
3
4  #Call:
5  lm(formula = price_range ~ ram, data = mobile.train)
6
7  #Residuals:
8  #     Min       1Q   Median       3Q      Max
9  #-1.35678 -0.27604 -0.00749  0.25758  1.37866
10
11 #Coefficients:
12 #              Estimate Std. Error t value Pr(>|t|)
13 #(Intercept) -5.262e-01  2.652e-02  -19.84   <2e-16 ***
14 #ram          9.539e-04  1.117e-05   85.36   <2e-16 ***
15 #---
16 #Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .
   0.1          1
17
18 #Residual standard error: 0.4517 on 1398 degrees of freedom
19 #Multiple R-squared:  0.839,  Adjusted R-squared:  0.8389
20 #F-statistic:  7286 on 1 and 1398 DF,  p-value: < 2.2e-16
```

Possiamo notare come i risultati della regressione siano molto significativi, circa l'84%. Per migliorare la percentuale ottenuta è necessario provare ad aggiungere le altre variabili correlate. Nel nostro caso viene aggiunta la capacità della batteria:

```
1  set.seed(1)
2
3  lreg2 = lm(price_range~ram+battery_power, data=mobile.train)
4  summary(lreg2)
5
6  #Call:
7  #lm(formula = price_range ~ ram + battery_power, data = mobile.train)
8
9  #Residuals:
10 #     Min       1Q   Median       3Q      Max
11 #-1.05952 -0.28163 -0.01029  0.25767  1.24176
12
13 #Coefficients:
14 #                Estimate Std. Error t value Pr(>|t|)
```

```
#(Intercept)    -1.152e+00   3.739e-02   -30.81    <2e-16 ***
#ram             9.523e-04   9.717e-06    98.01    <2e-16 ***
#battery_power   5.087e-04   2.392e-05    21.27    <2e-16 ***
#---
#Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .
   0.1         1

#Residual standard error: 0.3927 on 1397 degrees of freedom
#Multiple R-squared:  0.8784, Adjusted R-squared:  0.8782
#F-statistic:  5045 on 2 and 1397 DF,  p-value: < 2.2e-16
```

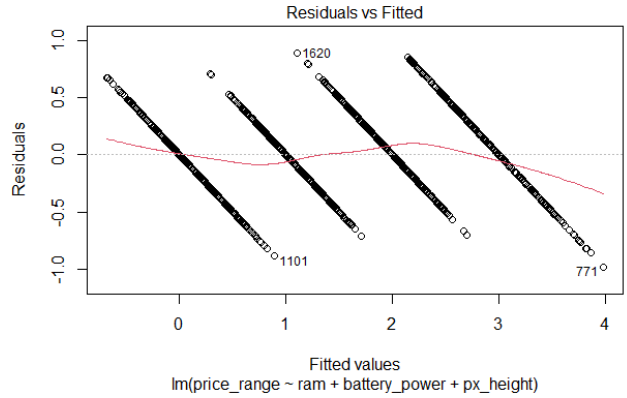
Si può notare che il modello è migliorato notevolmente, circa l'88%. Infine viene aggiunta anche la terza variabile correlata maggiormente, ovvero l'altezza dei pixel:

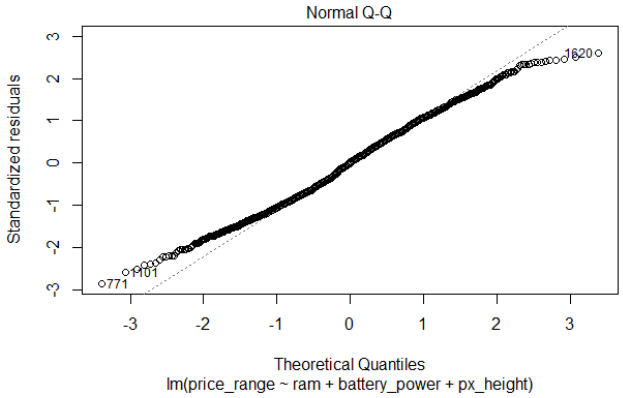
```
set.seed(1)

lreg3 = lm(price_range~ram+battery_power+px_height, data=mobile.train)
summary(lreg3)

#Call:
#lm(formula = price_range ~ ram + battery_power + px_height, data =
   mobile.train)

#Residuals:
#     Min       1Q   Median       3Q      Max
#-0.98084 -0.26204  0.00366  0.24717  0.88972

#Coefficients:
#                Estimate Std. Error t value Pr(>|t|)
#(Intercept)    -1.419e+00   3.522e-02   -40.29    <2e-16 ***
#ram             9.542e-04   8.512e-06   112.10    <2e-16 ***
#battery_power   4.987e-04   2.096e-05    23.80    <2e-16 ***
#px_height       4.193e-04   2.034e-05    20.61    <2e-16 ***
#---
#Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .
   0.1         1

#Residual standard error: 0.344 on 1396 degrees of freedom
```

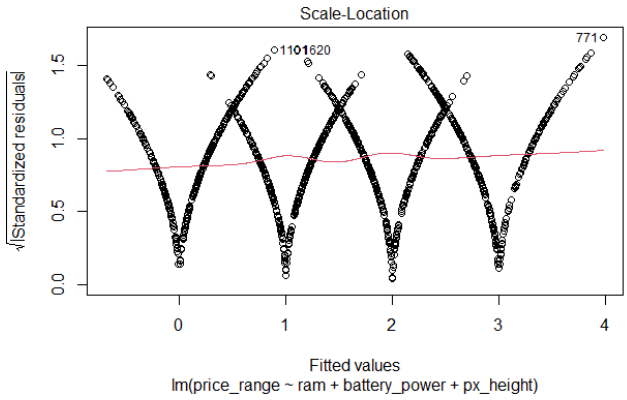
```
#Multiple R-squared:  0.9068, Adjusted R-squared:  0.9066
#F-statistic:  4525 on 3 and 1396 DF,  p-value: < 2.2e-16
```

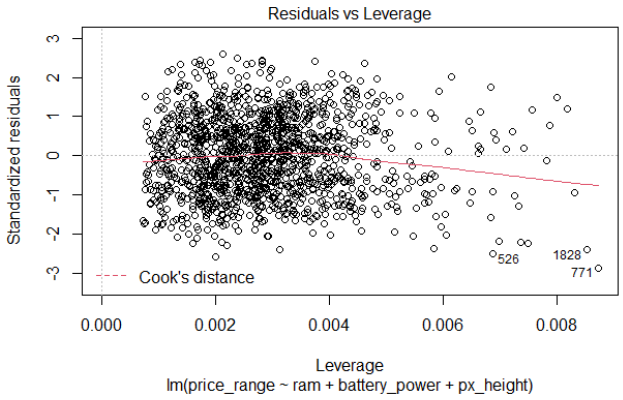
Aggiungendo la terza variabile correlata possiamo notare come il modello migliori ancora, circa il 91%.

4.2.1 Visualizzazione grafica dei residui









Queste rappresentazioni grafiche approvano ulteriormente il buon rendimento del nostro modello di regressione lineare.

Calcolo MSE

Infine viene effettuato il calcolo del *MSE* (Mean Squared Error), ovvero l'errore quadratico medio, che servirà per calcolare la distanza di un insieme di punti dalla retta di regressione. Il valore del MSE potrà differire tra il train set e il test set.

```
predict_lreg3_train <- predict(lreg3, mobile.train)
mean(mobile.train$price_range - predict_lreg3_train)^2
#[1] 1.937971e-28
predict_lreg3_test <- predict(lreg3, mobile.test)
mean(mobile.test$price_range - predict_lreg3_test)^2
#[1] 0.0002091202
```

Conclusioni

In conclusione, siamo in grado di rispondere alla domanda iniziale: **Quali sono le caratteristiche più importanti per prevedere il prezzo di uno smartphone?** Innanzitutto, osservando la matrice di correlazione, ci rendiamo conto che la RAM è la variabile più correlata alla fascia di prezzo, la capacità della batteria è la seconda variabile più correlata e, infine l'altezza dei pixel è la terza varibile più correlata alla fascia di prezzo.

Guardando il riepilogo del nostro modello di regressione lineare, osserviamo che tutte e tre le variabili (RAM, capacità della batteria e altezza dei pixel) sono caratterizzate da *t-value* molto elevati; ciò significa che sono tutte molto importanti nella previsione della nostra variabile target. In particolare, la RAM ha il t-value più elevato, a seguire la capacità della batteria e l'altezza dei pixel. Quest'ordine nel t-value corrisponde conseguentemente ai risultati ottenuti dalla nostra matrice di correlazione.

Di conseguenza, si può affermare che le caratteristiche più importanti nella previsione del prezzo di uno smartphone sono la **RAM**, la **capacità della batteria** e l'**altezza dei pixel**.

Ringraziamenti

Inizio ringraziando la mia relatrice la Prof.ssa Piccolomini, per essere stata sempre disponibile e cordiale lungo questo percorso. Si cresce anche grazie ai professori e lei mi ha aiutato moltissimo. Grazie davvero!

Vorrei ringraziare in modo particolare i miei genitori, che hanno sempre creduto in me e mi hanno sempre sostenuto, non facendomi mai mancare nulla. Senza di loro non sarei mai arrivato a questo punto! Ringrazio mia sorella Conny, alla quale devo molto. In primis perchè mi ha sempre ispirato a dare il meglio di me, costituendo un esempio da seguire e inoltre, nonostante la distanza, mi ha sempre supportato e mi ha sempre dimostrato la sua vicinanza nei momenti no. Vorrei ringraziare mio zio Michele, anche lui, seppur lontano, avendo avuto esperienze universitarie pregresse, mi ha sempre confortato e rasserenato anche quando credevo che il mio percorso universitario fosse compromesso.

Ringrazio Michela, che mi ha sempre sostenuto dal primo giorno, non facendomi sentire il peso di tutte le difficoltà che avrei dovuto affrontare.

Vorrei ringraziare tutti i miei colleghi di studio con cui ho condiviso tanto, seppur in un periodo particolare, e mi hanno dato la giusta carica per proseguire nel raggiungimento dei miei obiettivi. Inoltre vorrei ringraziare anche i miei conquilini Marco, Peppe, e Antonio per avermi "sopportato" e per aver condiviso momenti meravigliosi, che resteranno per sempre con me.

E infine, un grazie lo dedico a me, specialmente nei momenti più negativi, ho avuto la forza di non arrendermi mai e di continuare imperterrito sapendo che con i dovuti sacrifici sarei riuscito a realizzare tutto questo.

Bibliografia

[1] Ronald K. Pearson. *Exploratory Data Analysis Using R* .

[2] Stefano Bussolon. *Analisi dei dati con R.*

[3] Jim Frost. *Regression Analysis An Intuitive Guide for Using and Interpreting Linear Models.*

[4] Elena Loli Piccolomini, Antonio Messina, *Statistica e calcolo con R.*

[5] Sergio Venturini. *Manuale R*, `https://sergioventurini.github.io/manuale_R/`.

[6] Rahul Pandey. *How to ace Exploratory Data Analysis*, `https://medium.com/analytics-vidhya/how-to-ace-exploratory-data-analysis-d3821011532b`.

[7] Selva Prabhakaran. *Outlier detection and treatment with R*, `https://www.r-bloggers.com/2016/12/outlier-detection-and-treatment-with-r/`.

[8] Agnese Vardanega. *Ricerca sociale con R*, `https://www.agnesevardanega.eu/wiki/r/start`.