Progetto DataAnalysis & Statistical Learning

Filippelli Elisabetta,Picarelli Vincenzo, Serianni Francesco

Introduzione

L'OMS ha dichiarato che le malattie cardiovascolari sono la prima causa di morte nel mondo. Il numero delle vittime si attesta intorno ai 17 milioni. Se attraverso lo studio dei dati fosse possibile prevere il rischio di avere in un futuro prossimo una malattia cardiovascolare si potrebbe prevenire in modo da tenere lontano la malattia. Questo set di dati include importanti caratteristiche dei pazienti.

EDA

Per prima cosa importiamo il dataset.

```
data<-read.csv("heart-disease-data.csv")
```

Visualizziamo le prime righe del dataset.

```
kable(head(data),caption="Heart Disease dataset")
```

Table 1: Heart Disease dataset

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
57	1	0	140	192	0	1	148	0	0.4	1	0	1	1

Il dataset contiene:

```
dims = c(dim(data))
names(dims) = c("Numero di righe ", "Numero di colonne")
kable(t(dims), caption="Dimensione del dataset")
```

Table 2: Dimensione del dataset

Numero di righe	Numero di colonne
303	14

Variabili

Gli attributi del dataset nello specifico sono:

- **age**: età in anni
- **sesso**: 1=maschio, 0=femmina
- **cp**: tipo di dolore toracico (valore=0 angina tipica,valore1=angina atipica,valore 2= dolore non anginoso, valore 3=asintomatico)
- **trestbps** : pressione arteriosa a riposo (in mm Hg al ricovero in ospedale)
- **chol** : colesterolo sierico in mg/dl
- **fbs** : glicemia a digiuno >120 mg/dl (1=vero, 0= falso)
- **restecg**: risultati ECG riposo (valore0 =normale, valore 1= anomalia dell'onda ST-T, inversione dell'onda T e/o elevazione o depressione dell'onda ST>0.05 mV, valore 2=mostra una probabile o definita ipertrofia ventricolare sinistra definita secondo il cirterio di Estes)
- **thalach** :frequenza cardiaca massima raggiunta
- **exang**: angina indotta da esercizio (1=si, 0=no)
- **oldpeak** :Depressione ST indotta dall'esercizio rispetto al riposo
- **slope**: pendenza del segmento ST al picco dell'esercizio (valore 0= in salita, valore 1= piatto, valore 2= in discesa)
- **ca** : numero di vasi principali (0-3) colorati dalla flourosopia
- **thal**: 0 = normale; 1 = difetto fisso; 2 = difetto reversibile
- **target**= condizione malattia (0: nessuna malattia, 1 malattia)

```
name_variables = gsub("[_]","",names(data))
summ<-summary(data)[,1:7]
colnames(summ) = name_variables[1:7]

kable(summ,caption="Dettagli delle prime 7 Variabili")
```

Table 3: Dettagli delle prime 7 Variabili

	age	sex	cp	trestbps	chol	fbs	restecg
	Min. :29.00	Min. :0.0000	Min. :0.000	Min. : 94.0	Min. :126.0	Min. :0.0000	Min. :0.0000
	1st Qu.:47.50	1st Qu.:0.0000	1st Qu.:0.000	1st Qu.:120.0	1st Qu.:211.0	1st Qu.:0.0000	1st Qu.:0.0000
	Median :55.00	Median :1.0000	Median :1.000	Median :130.0	Median :240.0	Median :0.0000	Median :1.0000
	Mean :54.37	Mean :0.6832	Mean :0.967	Mean :131.6	Mean :246.3	Mean :0.1485	Mean :0.5281
	3rd Qu.:61.00	3rd Qu.:1.0000	3rd Qu.:2.000	3rd Qu.:140.0	3rd Qu.:274.5	3rd Qu.:0.0000	3rd Qu.:1.0000
	Max. :77.00	Max. :1.0000	Max. :3.000	Max. :200.0	Max. :564.0	Max. :1.0000	Max. :2.0000

```
summ = summary(data)[,8:14]
colnames(summ) = name_variables[8:14]

kable(summ,caption="Dettagli delle ultime 6 Variabili")
```

Table 4: Dettagli delle ultime 6 Variabili

thalach	exang	oldpeak	slope	ca	thal	target
Min. : 71.0	Min. :0.0000	Min. :0.00	Min. :0.000	Min. :0.0000	Min. :0.000	Min. :0.0000
1st Qu.:133.5	1st Qu.:0.0000	1st Qu.:0.00	1st Qu.:1.000	1st Qu.:0.0000	1st Qu.:2.000	1st Qu.:0.0000
Median :153.0	Median :0.0000	Median :0.80	Median :1.000	Median :0.0000	Median :2.000	Median :1.0000
Mean :149.6	Mean :0.3267	Mean :1.04	Mean :1.399	Mean :0.7294	Mean :2.314	Mean :0.5446
3rd Qu.:166.0	3rd Qu.:1.0000	3rd Qu.:1.60	3rd Qu.:2.000	3rd Qu.:1.0000	3rd Qu.:3.000	3rd Qu.:1.0000
Max. :202.0	Max. :1.0000	Max. :6.20	Max. :2.000	Max. :4.0000	Max. :3.000	Max. :1.0000

Si mostra la struttura del dataset:

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : int  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : int  3 2 1 1 0 0 1 1 2 2 ...
##  $ trestbps: int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : int  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalach : int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : int  0 0 2 2 2 1 1 2 2 2 ...
##  $ ca      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thal    : int  1 2 2 2 2 1 2 3 3 2 ...
##  $ target  : int  1 1 1 1 1 1 1 1 1 1 ...
```

Si notano nel dataset diverse variabili categoriali. Verifichiamo la presenza di valori mancanti NaN nel dataset:

```
sum_NaN<-sum(is.na(data))
kable(sum_NaN,caption="Totale valori NaN")
```

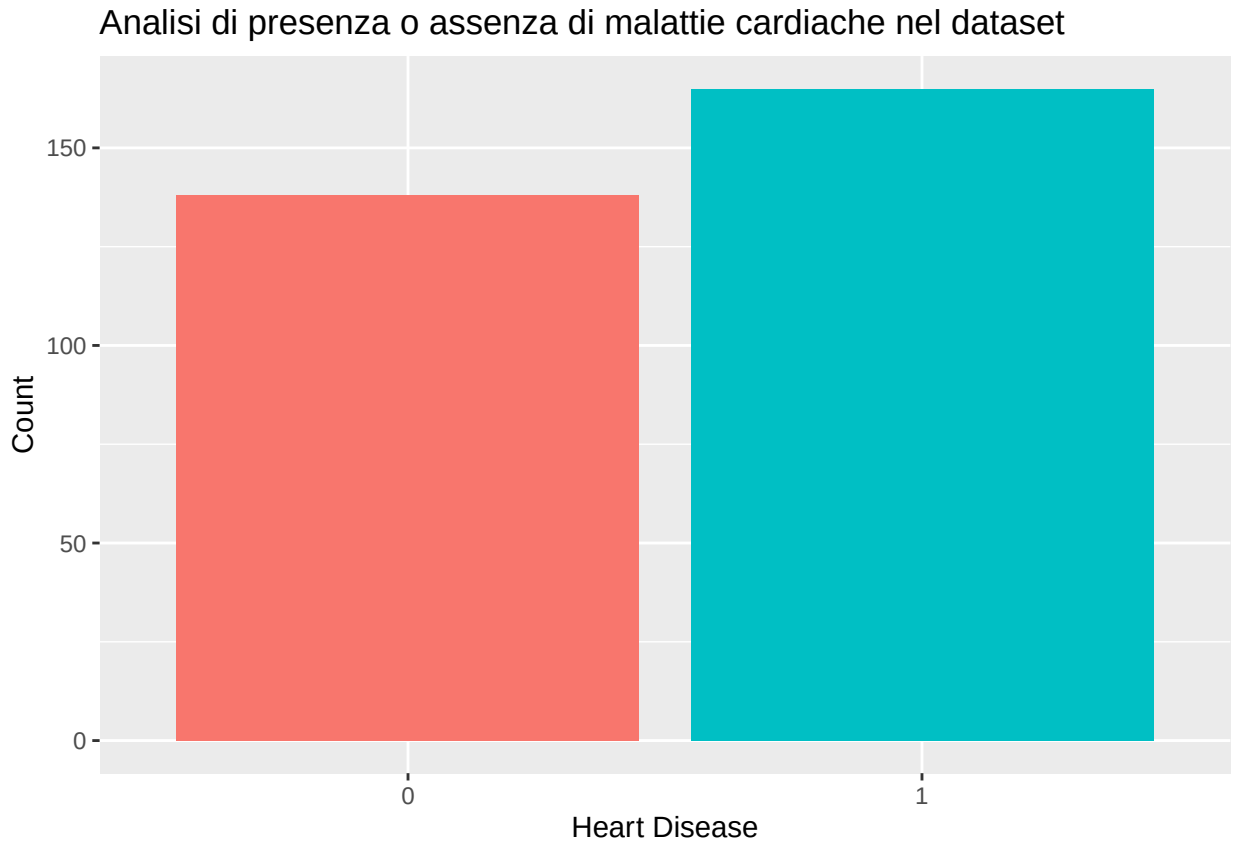
Table 5: Totale valori NaN

x
0

Non sono presenti valori NaN. Il dataset sembra essere abbastanza pulito, ma presenta molte dimensioni.

Data Visualization

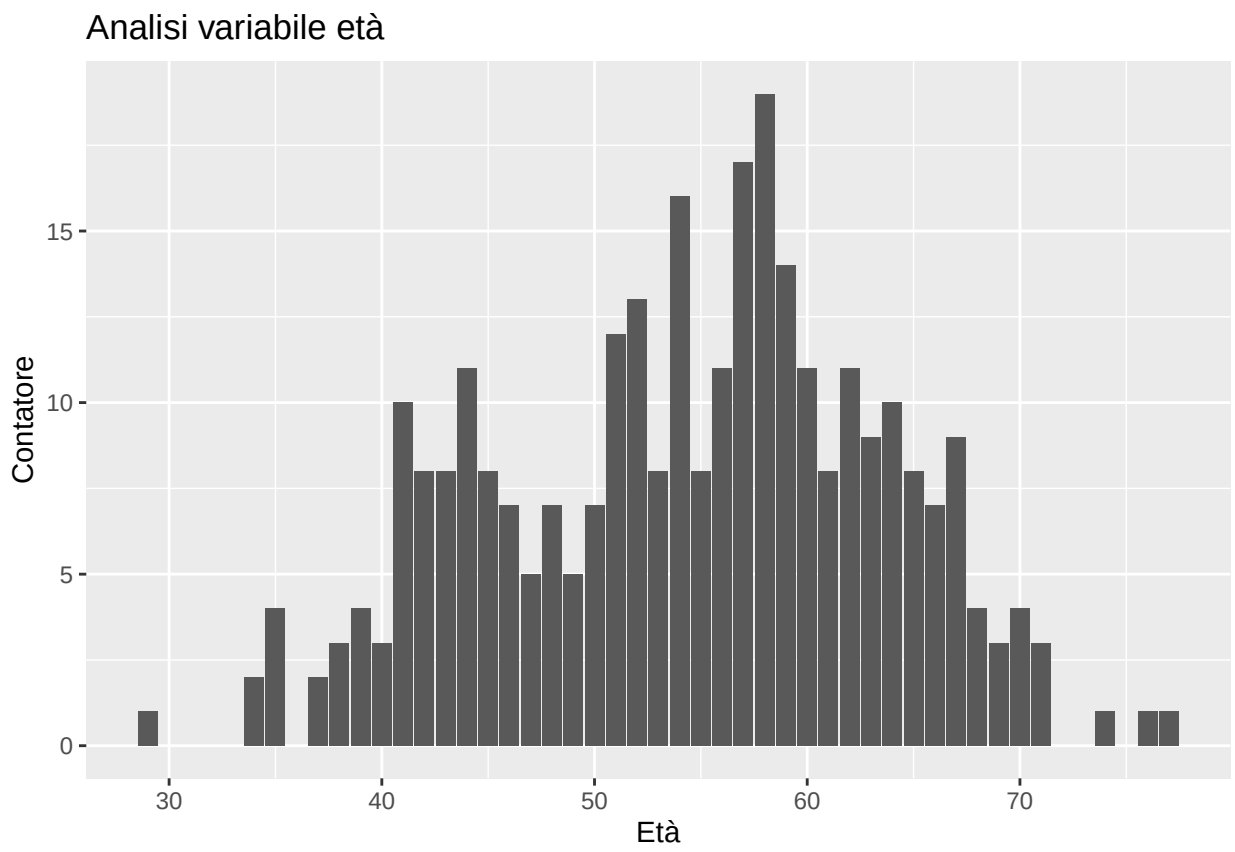
```
ggplot(data, aes(x=as.factor(target), fill=as.factor(target) )) +
  geom_bar( ) +
  xlab("Heart Disease") +
  ylab("Count") +
  ggtitle("Analisi di presenza o assenza di malattie cardiache nel dataset") +
  scale_fill_discrete(name = "Heart Disease", labels = c("Absence", "Presence"))+
  theme(legend.position="none")
```



Dai dati disponibili nel dataset possiamo dire che non è sbilanciato.

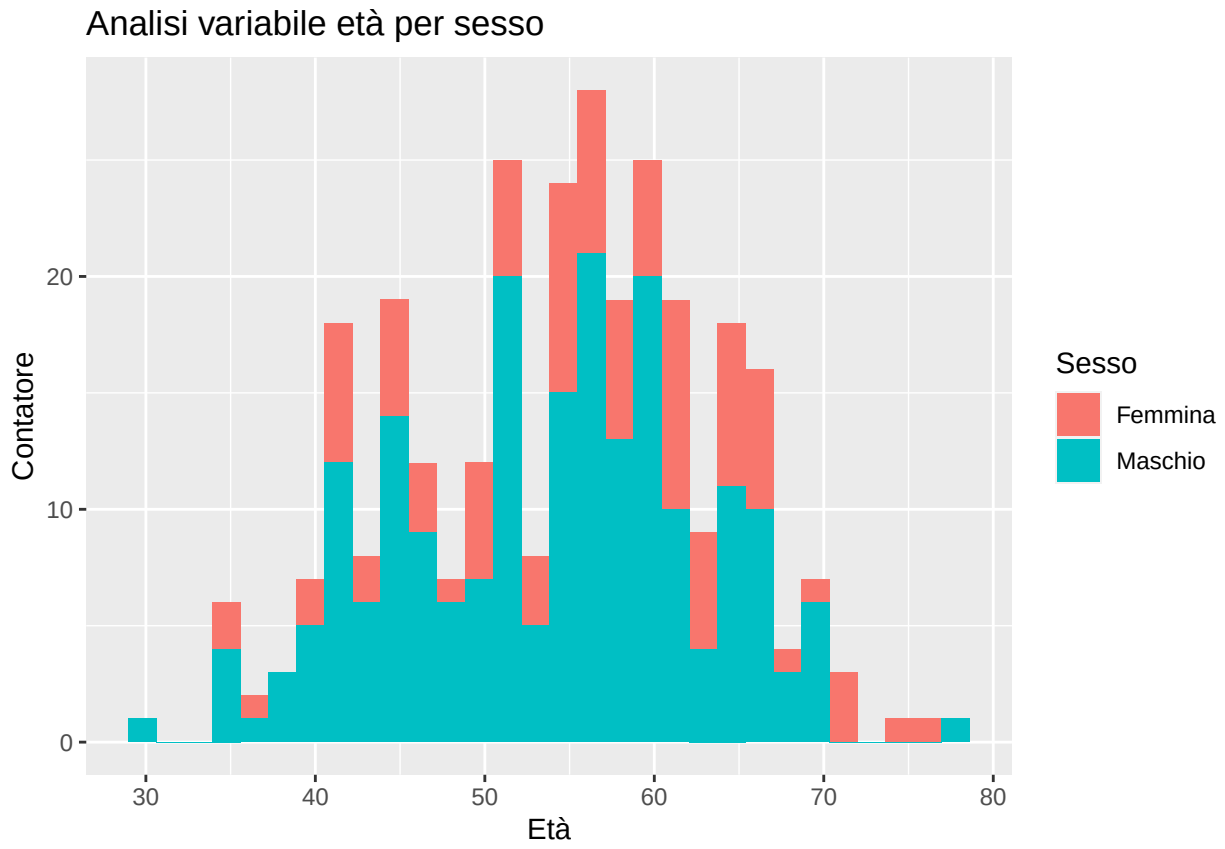
Si mostra l'analisi sulla varaibiale **age**:

```
ggplot(data,aes(x=age))+
  geom_bar()+
  ggtitle("Analisi variabile età") +
  xlab("Età")  +
  ylab("Contatore")
```



Distribuzione di femmine e maschi per età

```
ggplot(data,aes(x=age,fill=as.factor(sex)))+
  geom_histogram(bins = 30)+
  ggtitle("Analisi variabile età per sesso") +
  xlab("Età")  +
  ylab("Contatore")+
  scale_fill_discrete(name="Sesso",labels=c("Femmina","Maschio"))
```



Livelli di colesterolo per maschi e femmine in base all'età:

```
ggplot(data,aes(x=age,y=chol,color=as.factor(sex), size=chol))+
  geom_point(alpha=0.7)+
  xlab("Età") +
  ylab("colesterolo [mg/dl]")+
  labs(title = "Livelli di colesterolo per sesso ed età\n",size="Colesterolo [mg/dl]")+
  scale_color_discrete(name = "Sesso",labels=c("Femmina","Maschio"))
```

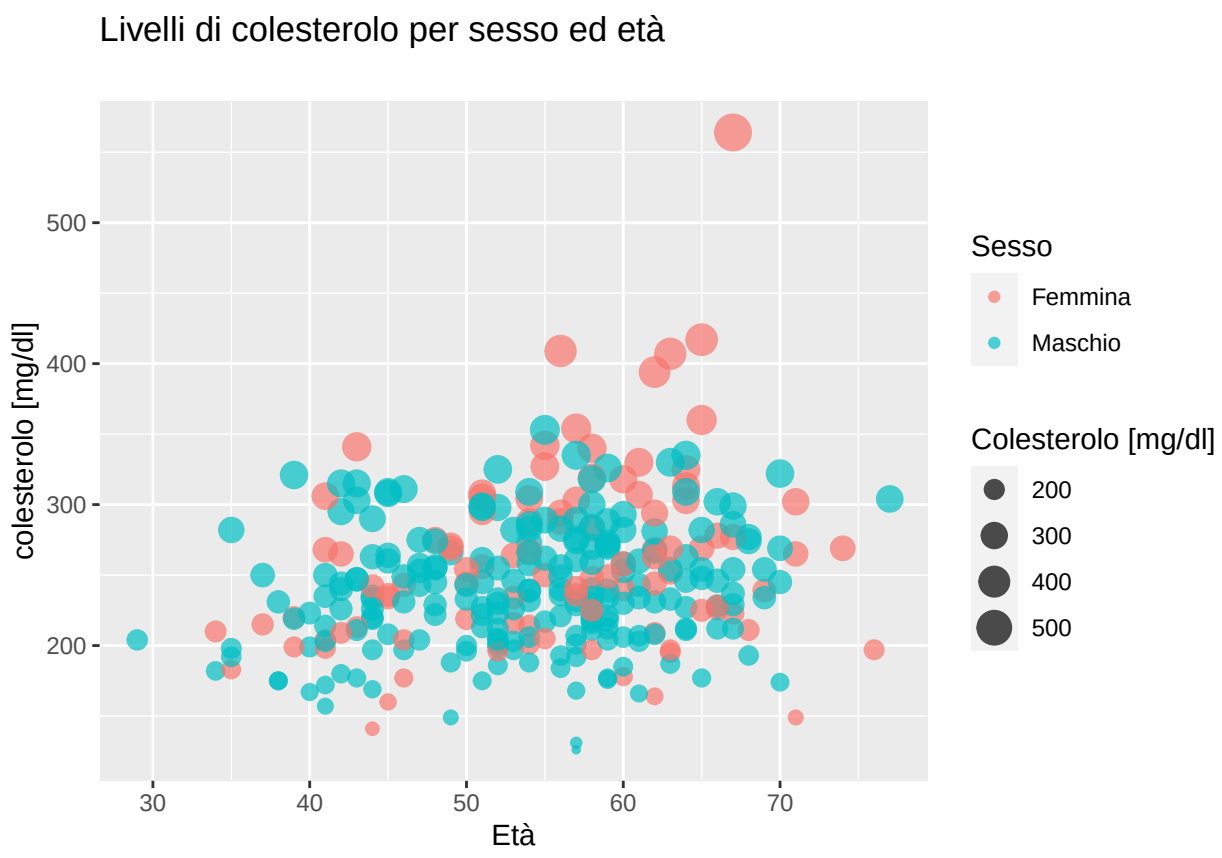
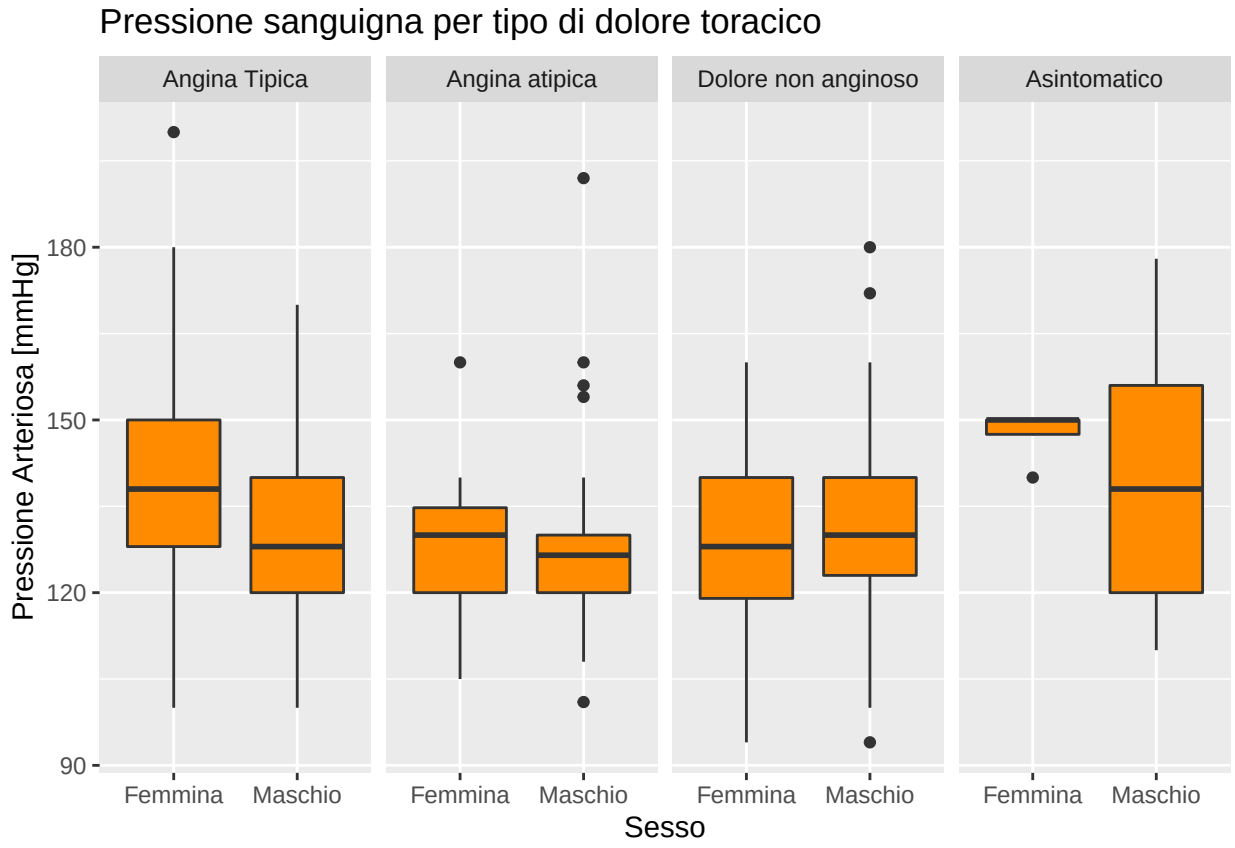


Grafico della pressione sanguigna in base al tipo di dolore toracico per sesso dei pazienti

```
cp.labs <- c("Angina Tipica", "Angina atipica","Dolore non anginoso","Asintomatico")
names(cp.labs) <- c(0,1,2,3)

ggplot(data,aes(x=as.factor(sex),y=trestbps))+
  geom_boxplot(fill="darkorange")+
  labs(title="Pressione sanguigna per tipo di dolore toracico",x="Sex",
       y="Pressione Arteriosa [mmHg]")+
  scale_x_discrete(name="Sesso",labels=c("Femmina","Maschio"))+
  facet_grid(~cp,labeller=labeller(cp=cp.labs))
```



PCA

Analizziamo il dataset mediante la PCA. Nel dataset sono presenti delle variabili categoriali che non possono essere utilizzate per la pca.

```
data_pca<-data[,c(1,4,5,8,10,14)]
```

In questo modo utilizziamo per la PCA solo dati non categoriali. In particolare usiamo le variabili:

- **age**
- **trestbps**
- **chol**
- **thalach**
- **oldpeak**

e riportiamo per ora anche i dati di **num**, la variabile target.

Per evitare problemi legati alla diversa scala delle variabili si standardizza il dataset. Si selezionano le prime 5 variabili, si esclude **target** per la PCA

```
data.new=data_pca[,1:5]
res.pca <- PCA(data.new, graph = FALSE)
```

Gli autovalori misurano la quantità di variazione trattenuta da ciascuna componente principale. Hanno un valore maggiore per le prime PC e piccoli per le PC successive. Le prime PC corrispondono alle direzioni con la massima varianza. Si esaminamo gli autovalori per determinare il numero di componenti principali da considerare.

```
eig.val <- get_eigenvalue(res.pca)
kable(eig.val,caption="Autovalori per componenti principali")
```

Table 6: Autovalori per componenti principali

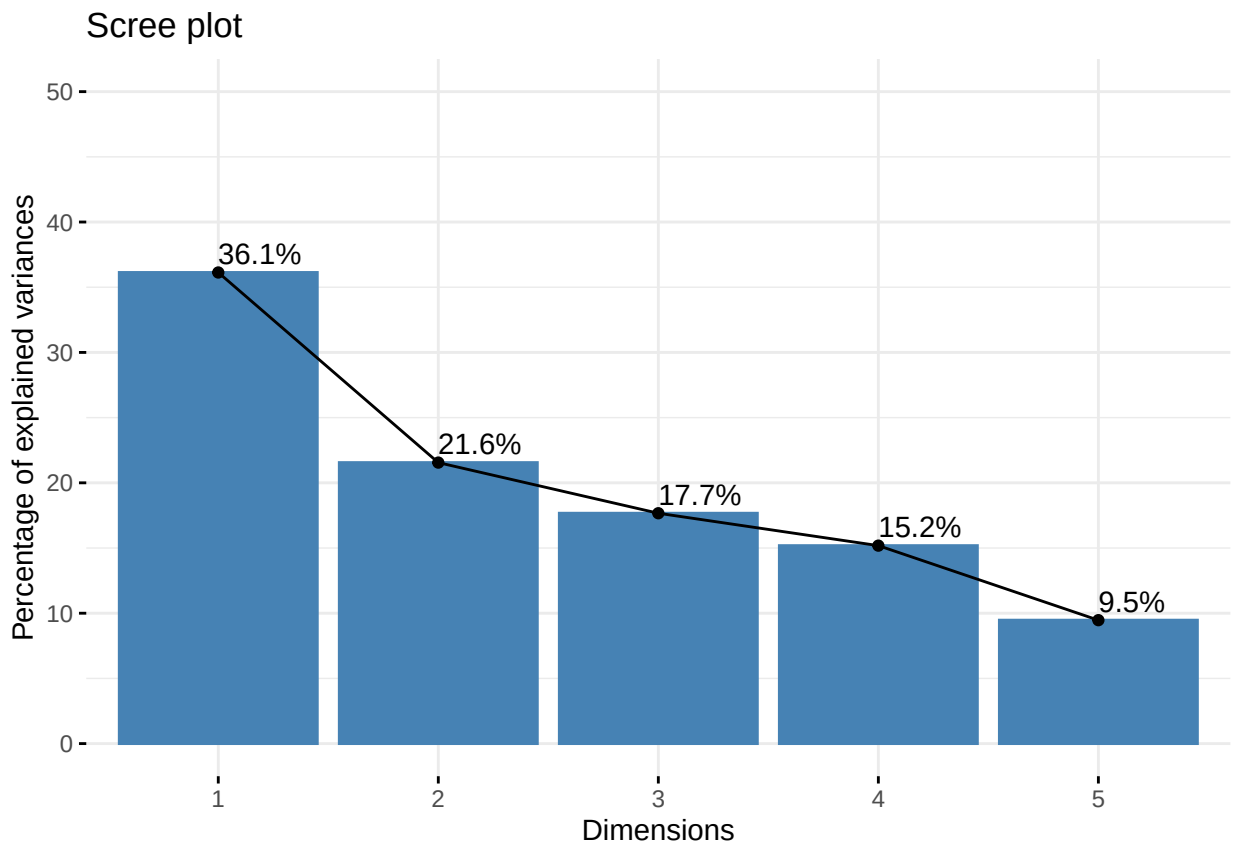
	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	1.8065754	36.131509	36.13151
Dim.2	1.0775475	21.550951	57.68246
Dim.3	0.8833766	17.667531	75.34999
Dim.4	0.7591714	15.183429	90.53342
Dim.5	0.4733290	9.466581	100.00000

Nella tabella vengono indicati per le diverse componenti principali:

- Prima colonna: il valore dell'autovalore stesso,
- Seconda colonna: la porzione di variabile spiegata da quella determinata PC,
- Terza colonna: la percentuale cumulativa ottenuta sommando le successive porzioni di variazione spiegate.

Per la scelta delle componenti principali uno dei metodi utilizzati è il metodo grafico dello Scree plot. Lo scree plot rappresenta le PC e le percentuali di varianza spiegata di volta in volta dalle componenti principali. L'identificazione di un gomito nell'andamento delle percentuali della varianza permettono di scegliere il numero di componenti principali.

```
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 50))
```



Andare ad identificare un gomito in questo grafico non è immediato. Si opta ,quindi, per un altro approccio. Basandoci sulla regola di Kaiser scegliamo gli autovalori con valore maggiore di 1. Quindi potremmo scegliere le prime 2 componenti principali. Per avere maggiore informazione si decide di scegliere una percentuale di varianza cumulativa del 75% circa quindi si prendono in cosiderazione le prime 3 PC.

Correlation Plot

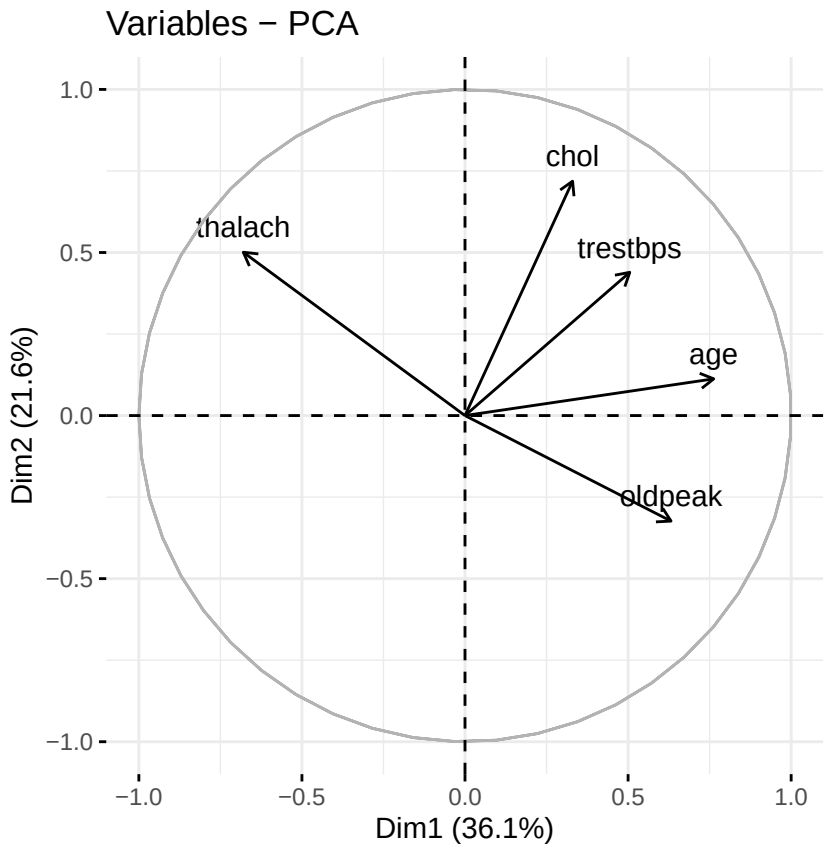
Si esamina ora la correlazione tra le variabli e le componenti principali.

```
var <- get_pca_var(res.pca)
kable(var$coord,caption="Coordinate per le varaibili")
```

Table 7: Coordinate per le varaibili

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
age	0.7626348	0.1125733	-0.1676556	-0.4530729	0.4151288
trestbps	0.5051709	0.4392683	0.6950120	-0.1253828	-0.2303979
chol	0.3298126	0.7186249	-0.4997870	0.3294728	-0.1283061
thalach	-0.6796796	0.5008213	0.2804304	0.1510828	0.4309831
oldpeak	0.6316844	-0.3235342	0.2092802	0.6378066	0.2137873

```
fviz_pca_var(res.pca, col.var = "black")
```



Indipendentemente dal numero di PC che si vuole estrarre nel grafico si ha come assi le prime due PC. Ogni variabile è indicata da un vettore che parte dall'origine. L'angolo che si crea tra ogni coppia di variabili può essere usata come approssimazione della correlazione che esiste tra le variabili. In questo modo si possono identificare gruppi di variabili che sono correlate positivamente tra di loro(hanno un piccolo angolo tra loro). La lunghezza indica quanto bene le due PC rappresentano la variabile in questione.

Si mostra il contributo che hanno le diverse variabili rispetto alle PC.

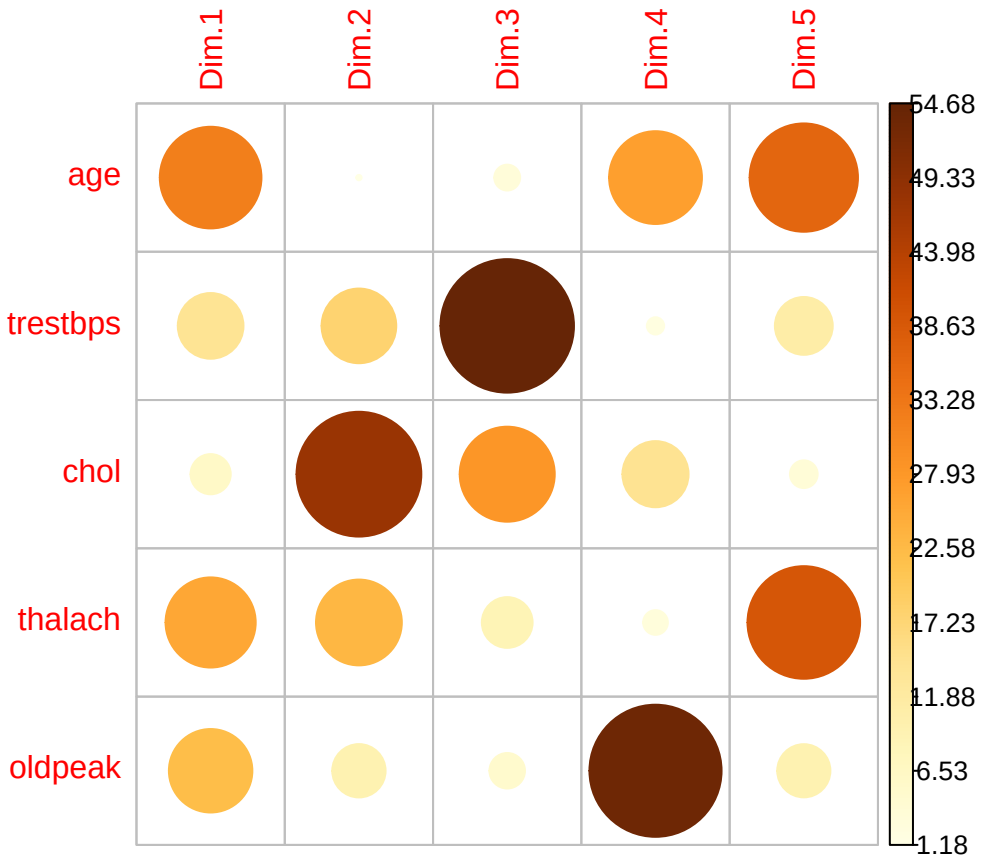
```
kable(round(var$contrib,digits = 3),caption="Contributo rispetto a PC")
```

Table 8: Contributo rispetto a PC

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5
age	32.194	1.176	3.182	27.039	36.408
trestbps	14.126	17.907	54.681	2.071	11.215
chol	6.021	47.926	28.276	14.299	3.478
thalach	25.571	23.277	8.902	3.007	39.243
oldpeak	22.087	9.714	4.958	53.584	9.656

La matrice di correlazione è data da:

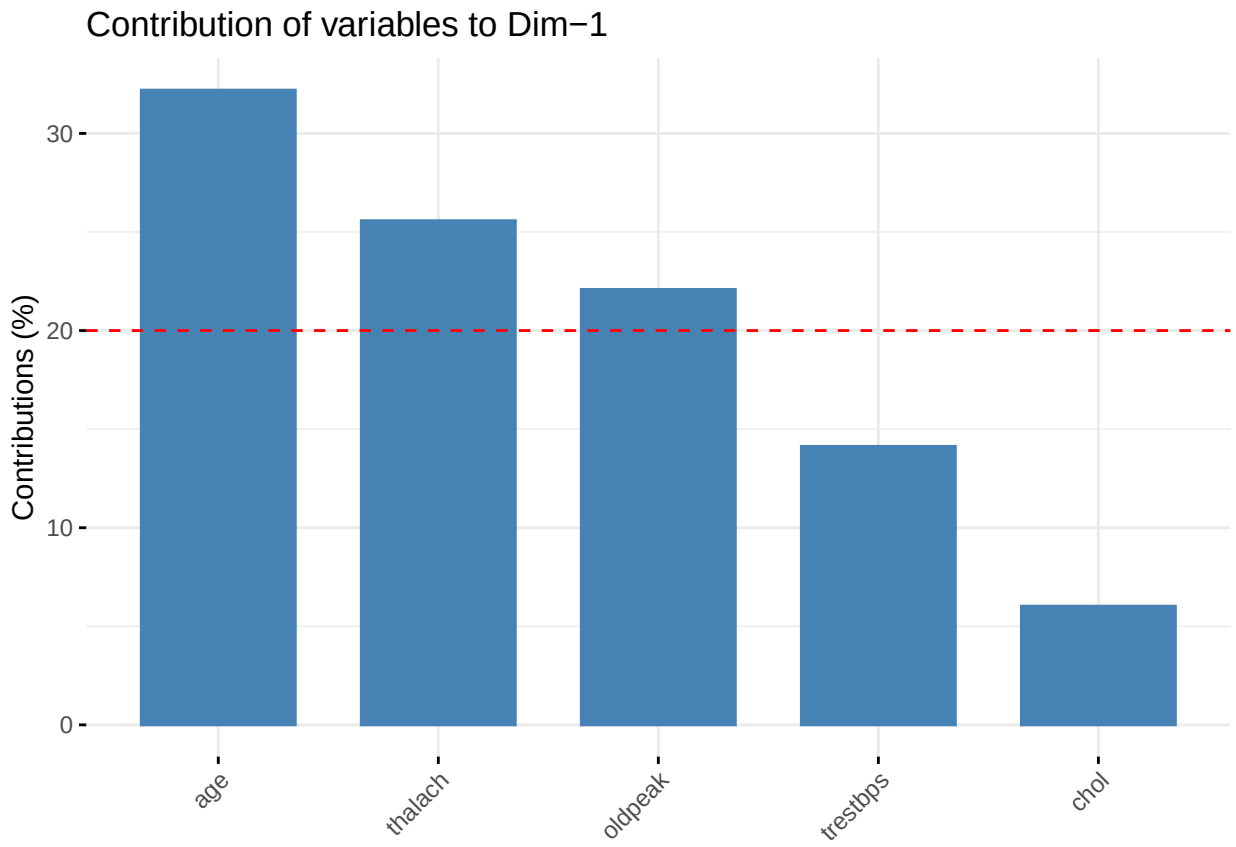
```
corrplot(var$contrib, is.corr=FALSE)
```



Per ogni variabile in analisi si ottengo dei pallini che sono tanto più grandi e tanto più scuri quanto più essi sono importanti per quella componente principale.

Nello scpecifico il contributo che hanno le variabili per la PC1. Visualizziamo per le 5 variaibli il contributo della componente principale PC1.

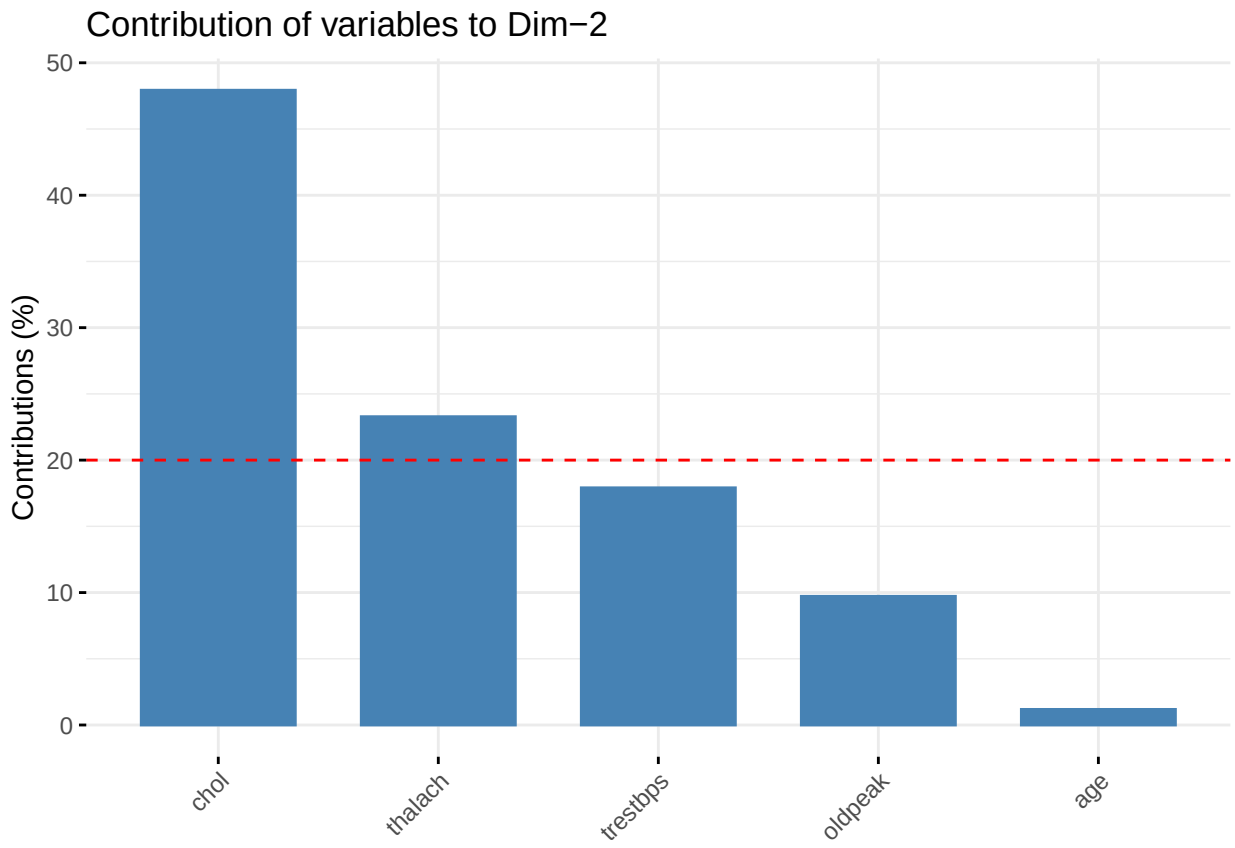
```
fviz_contrib(res.pca, choice = "var", axes = 1)
```



La linea rossa rappresenta il contributo che ci si aspetterebbe se tutte le variabili contribuissero a spiegare la variabilità della PCA nella stessa maniera. Quindi posso ritenere le prime 3 variabili utili per spiegare la prima PC.

Contributi alle variabili da parte della componente principale PC2.

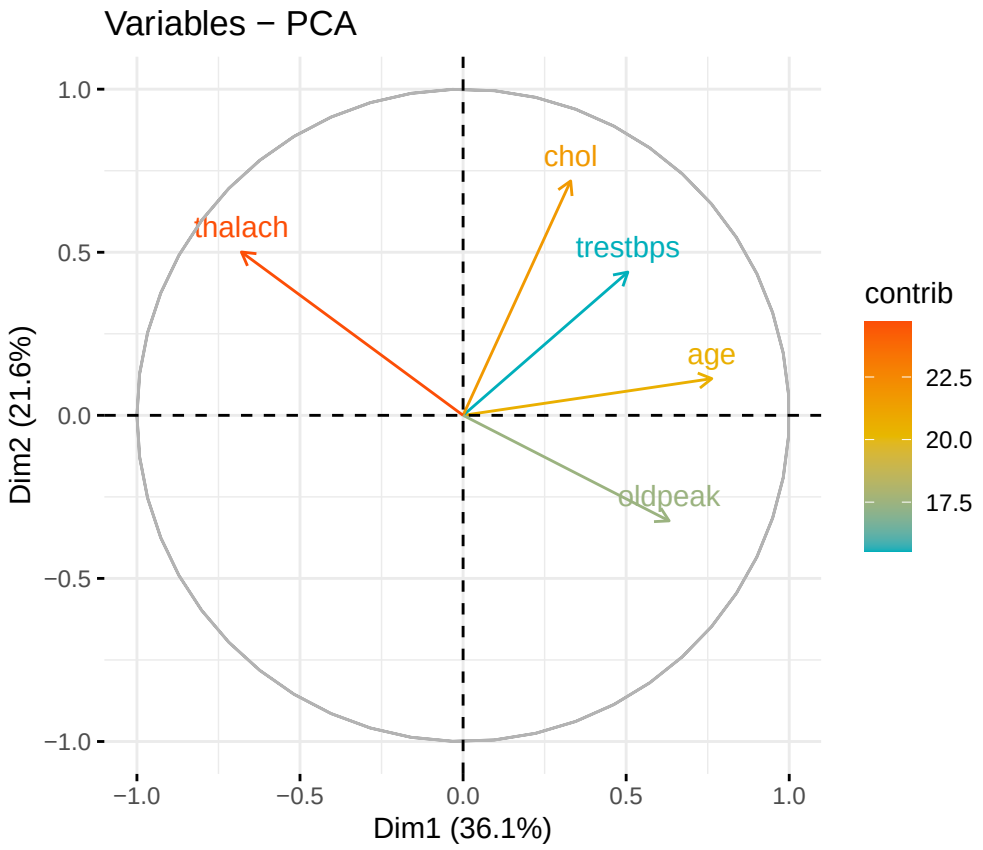
```
fviz_contrib(res.pca, choice = "var", axes = 2)
```



Il contributo delle prime 2 variabiali è molto alto per la PC2.

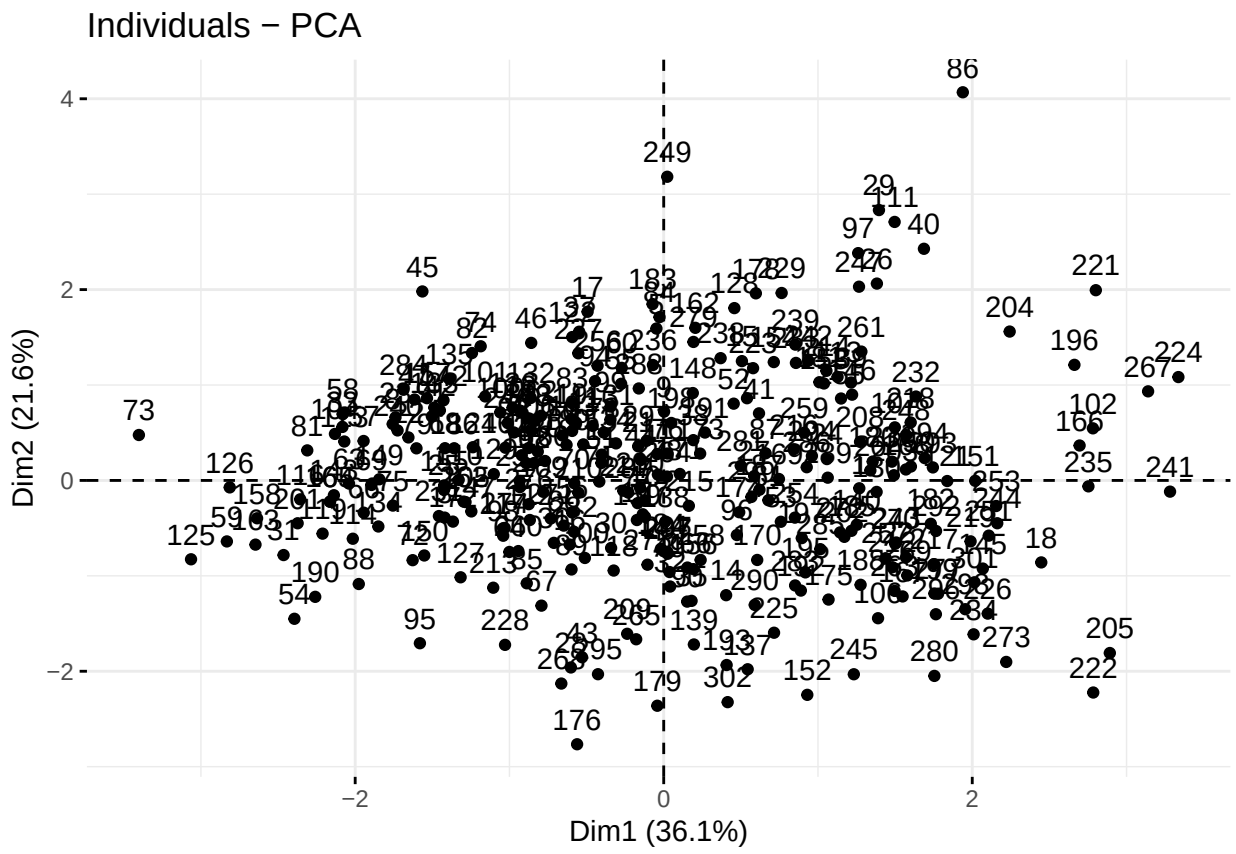
In questo grafico si evidenzia mediante l'uso dei colori il contributo che danno le variabili alle prime due componenti principali. I vettori delle variabili più tendono al rosso-arancione più sono importanti per le prime due PC.

```
fviz_pca_var(res.pca, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")
)
```



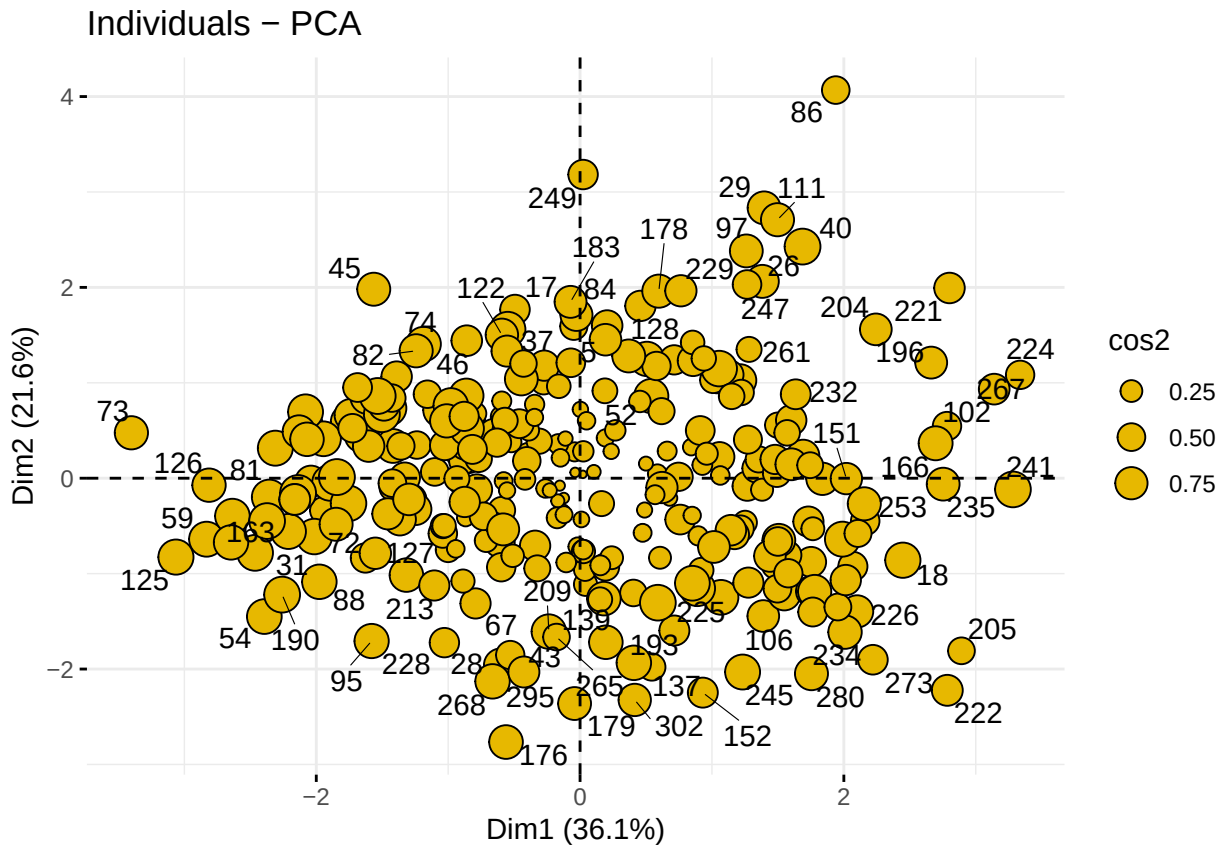
A partire dal risultato della PCA, per le prime due PC si plottano le osservazioni e l'importanza che ogni osservazione ha rispetto a quella PC e viene misurata dal coseno al quadrato che indica il contributo che quell'osservazione ha avuto nella determinazione della PC. Più sono distanti dal centro, più è alto il coseno al quadrato più quell'osservazione è stata importante per quella PC. Più sono vicine meno importante è l'osservazione di quella componente principale. Più è vicina al centro meno contribuisce a spiegare la PC.

```
fviz_pca_ind(res.pca)
```



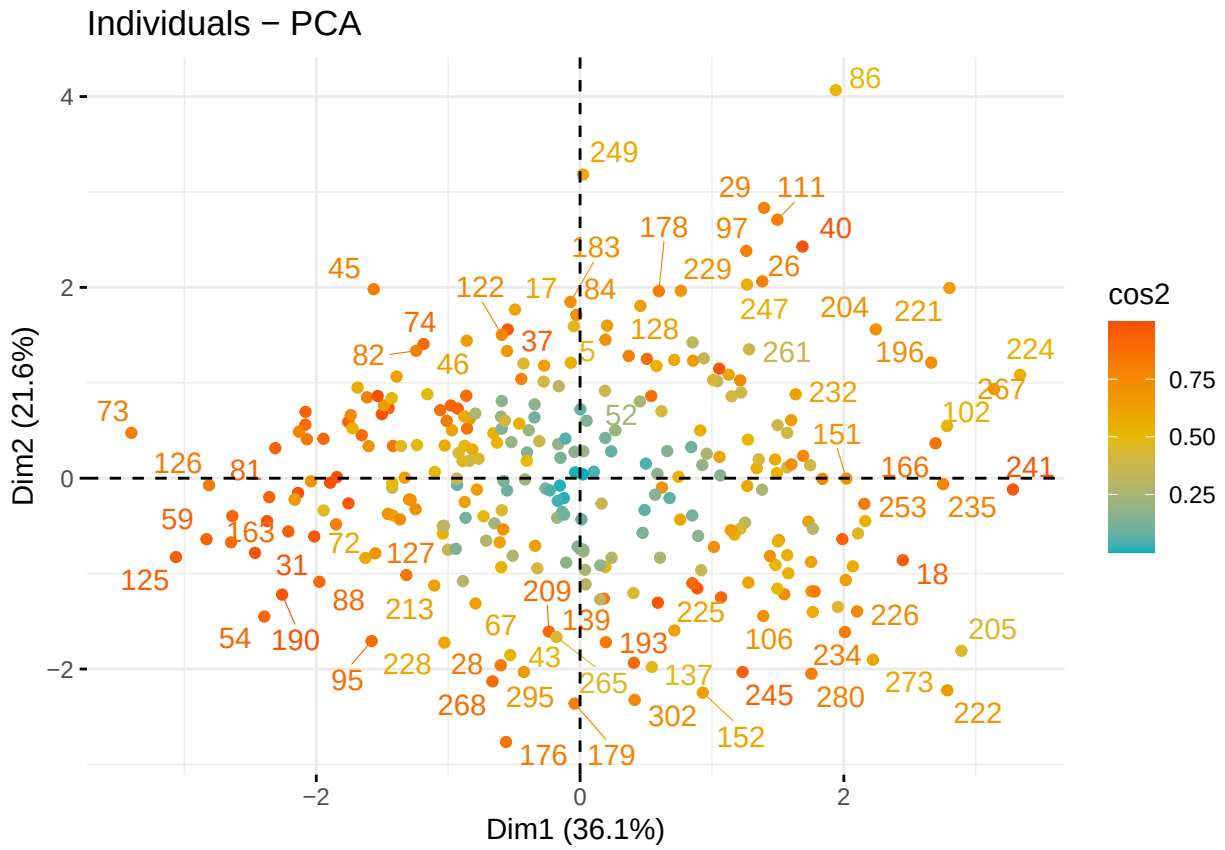
I pallini sono proporzionali al valore di coseno al quadrato.

```
fviz_pca_ind(res.pca, pointsize = "cos2",
             pointshape = 21, fill = "#E7B800",
             repel = TRUE
)
```



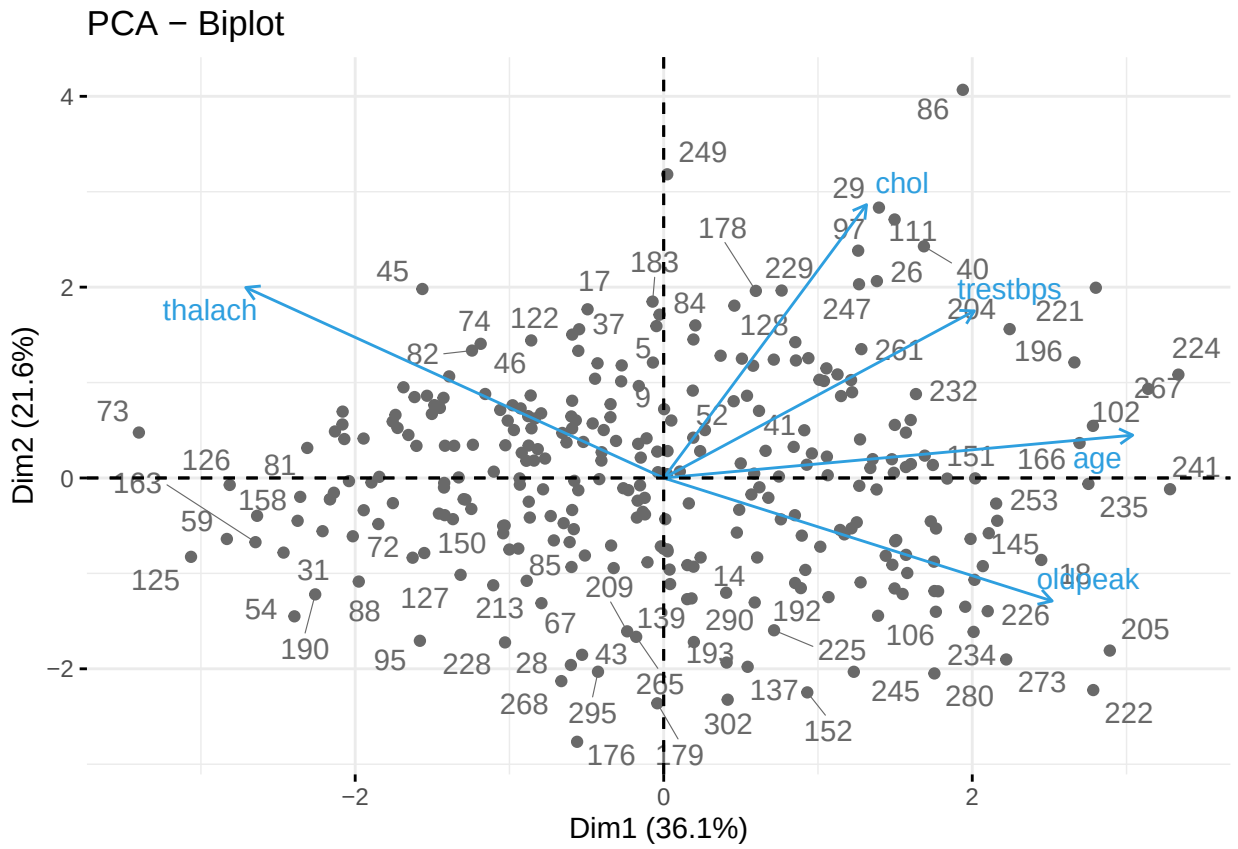
In questo grafico invece si va a colorare l'osservazione in base al valore di coseno al quadrato.

```
fviz_pca_ind(res.pca, col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE
)
```



Nel biplot vengono rappresentate le informazioni per le 2 PC delle variaibili e delle osservazioni. Il grafico a causa dell'elevato numero di osservazioni e di variabili non risulta tanto leggibile. Un'unità statistica che si trova vicino alla variabile ha un contributo alto per quella variabile, mentre se l'unità statistica si trova dal lato opposto a quella variabile ha un contributo basso rispetto a quella variabile.

```
fviz_pca_biplot(res.pca, repel = TRUE,
                col.var = "#2E9FDF",
                col.ind = "#696969"
)
```



Clustering gerarchica

Un altro metodo per semplificare i dati è quello della cluster gerarchica, che verrà utilizzata per capire se è possibile identificare dei gruppi di unità simili all'interno del dataset.

Prendiamo in considerazione il dataset utilizzato per la PCA e iniziamo a standardizzare i dati.

```
uns_df <- scale(data.new)
```

Calcoliamo la matrice di dissimilarità considerando la distanza euclidea

```
res.dist <- dist(uns_df, method = "euclidean")
```

Per avere una visualizzazione più semplice delle distanze appena calcolate, sistemiamo i dati in una matrice.

```
round(as.matrix(res.dist)[1:6, 1:6],digits = 2)
```

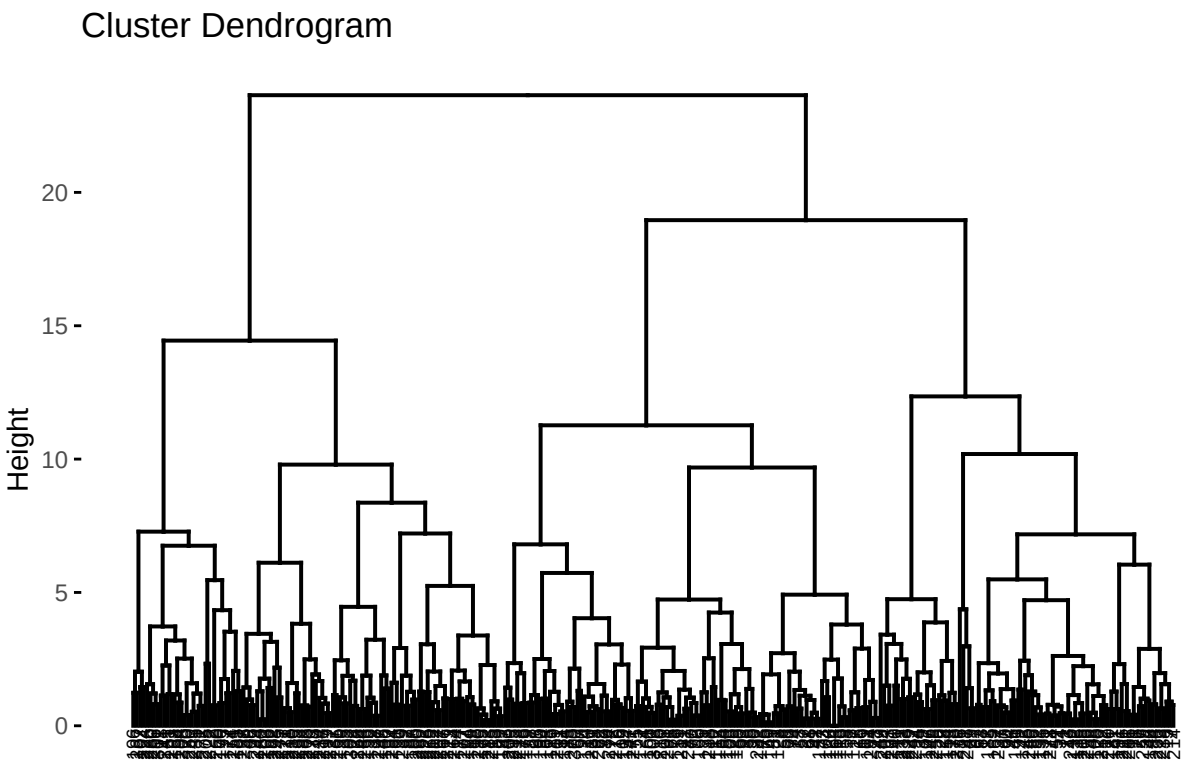
```
##      1    2    3    4    5    6
## 1 0.00 3.57 2.90 2.41 3.22 1.96
## 2 3.57 0.00 2.16 3.22 4.07 4.06
## 3 2.90 2.16 0.00 1.94 3.53 2.31
## 4 2.41 3.22 1.94 0.00 2.38 1.97
## 5 3.22 4.07 3.53 2.38 0.00 3.40
## 6 1.96 4.06 2.31 1.97 3.40 0.00
```

Utilizziamo i risultati della matrice di dissimilarità per applicare il clustering gerarchico agglomerativo con il metodo di Ward, perché è il metodo che generalmente performa meglio rispetto gli altri metodi.

```
res.hc <- hclust(d = res.dist, method = "ward.D2")
```

Per rappresentare graficamente i risultati della cluster gerarchica, costruiamo il dendogramma. In questo caso, sembra suggerire 2 cluster.

```
fviz_dend(res.hc, cex = 0.5)
```



Per misurare la bontà di clustering, calcoliamo le distanze cofenetiche.

```
res.coph <- cophenetic(res.hc)
```

E calcoliamo la correlazione tra le distanze cofenetiche e le distanze originali.

```
cor(res.dist, res.coph)
```

```
## [1] 0.4229219
```

Il risultato non è ottimale. Eseguiamo il clustering gerarchico con il metodo del legame medio, ottenendo in questo modo una correlazione più alta.

```
res.hc2 <- hclust(res.dist, method = "average")
cor(res.dist, cophenetic(res.hc2))
```

```
## [1] 0.7016001
```

Tagliamo il dendogramma in due gruppi

```
grp <- cutree(res.hc, k = 2)
```

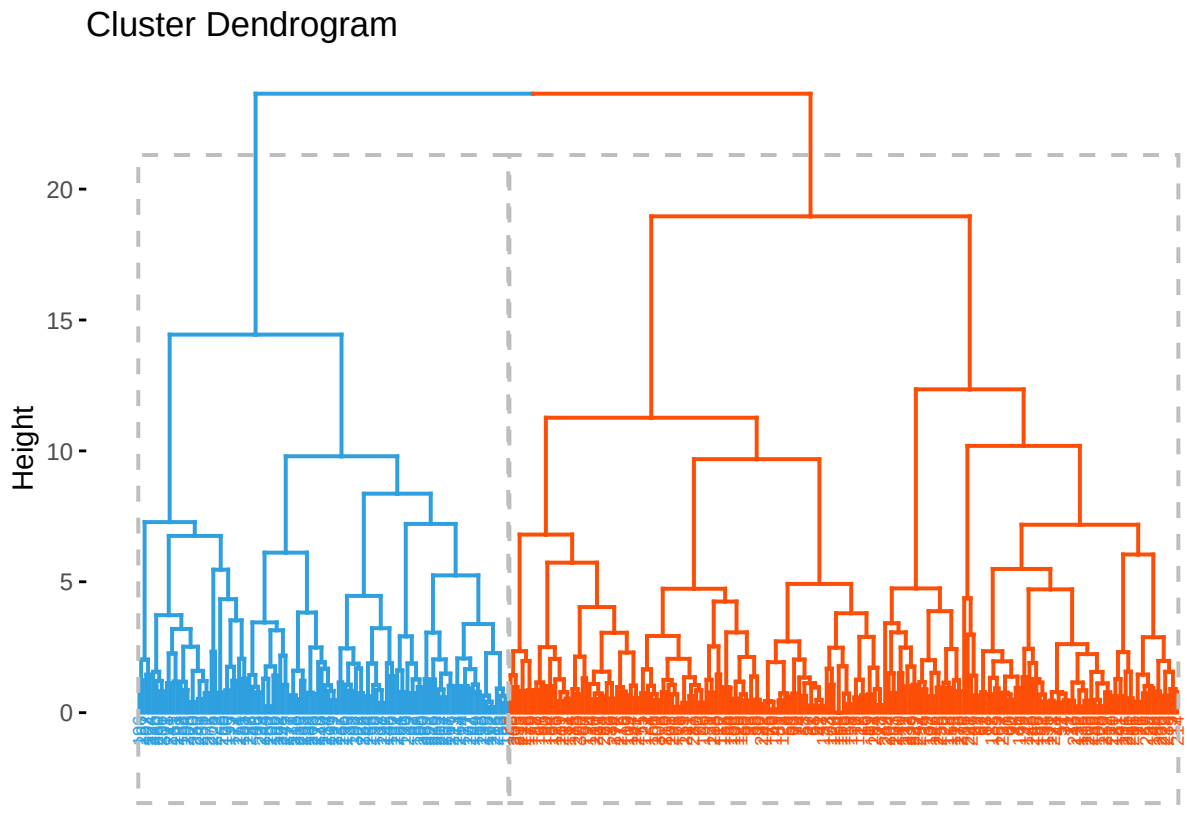
Vediamo la numerosità di ciascun gruppo

```
table(grp)
```

```
## grp
##   1   2
## 108 195
```

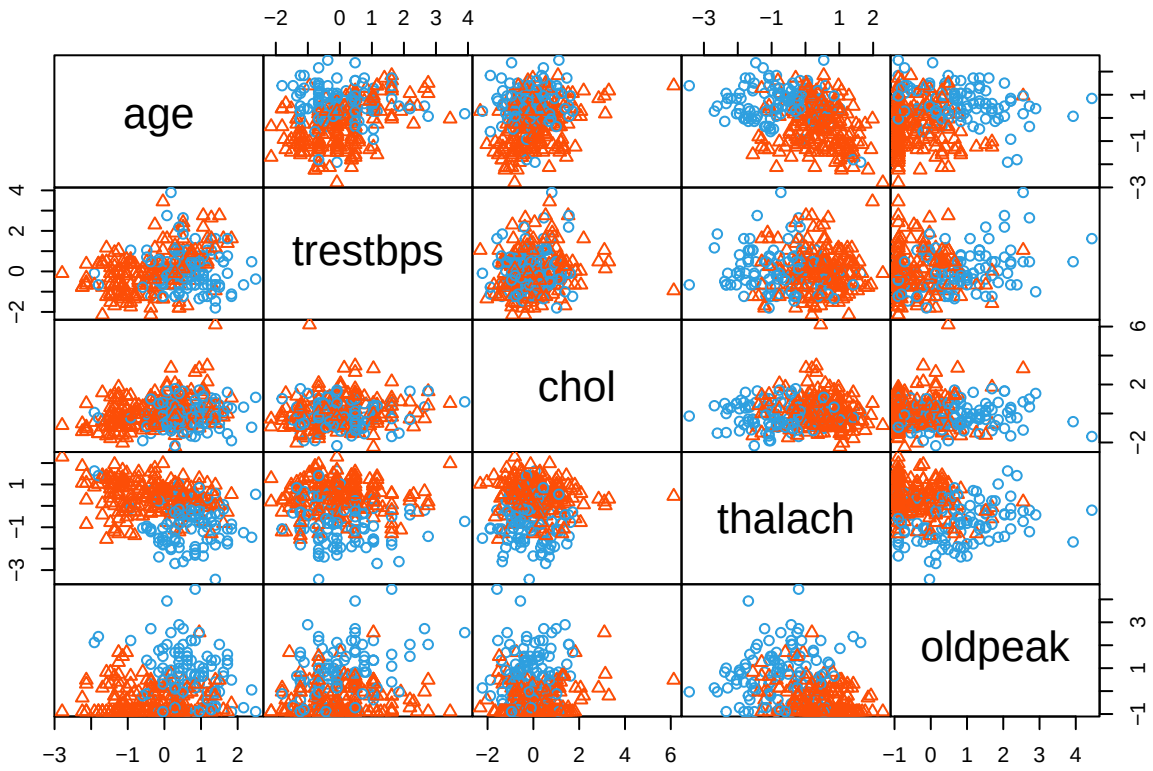
Creiamo nuovamente il dendogramma, distinguendo con un colore diverso ciascun gruppo

```
fviz_dend(res.hc, k = 2,
          cex = 0.5,
          k_colors = c("#2E9FDF", "#FC4E07"),
          color_labels_by_k = TRUE,
          rect = TRUE
)
```



Di seguito invece visualizziamo i risultati tramite una matrice di grafici a dispersione.

```
pairs(uns_df, gap=0, pch=grp, col=c("#2E9FDF", "#FC4E07")[grp])
```



K-means

Si utilizza il dataset **data.new** dove è stata rimossa la colonna **num**, ovvero la variabile target. Come primo passo si standardizzano i dati nel dataset per renderli comparabili.

```
uns_df <- scale(data.new)
kable(head(uns_df),caption="Dati standardizzati")
```

Table 9: Dati standardizzati

age	trestbps	chol	thalach	oldpeak
0.9506240	0.7626941	-0.2559104	0.0154173	1.0855423
-1.9121497	-0.0925846	0.0720803	1.6307737	2.1190672
-1.4717230	-0.0925846	-0.8154238	0.9758995	0.3103986
0.1798773	-0.6627704	-0.1980297	1.2378492	-0.2063639
0.2899839	-0.6627704	2.0786111	0.5829750	-0.3786180
0.2899839	0.4776012	-1.0469466	-0.0718993	-0.5508722

Per utilizzare K-means bisgona definire il numero di cluster che vogliamo trovi nel dataset.Si sceglie k=2 e come valore di nstart=25.

```
set.seed(123)
res.km <- kmeans(uns_df, 2, nstart = 25)
```

La variaible **res.km** mostra il gruppo per ogni variabile.
Si visualizzano per esempio le prime 5 righe.

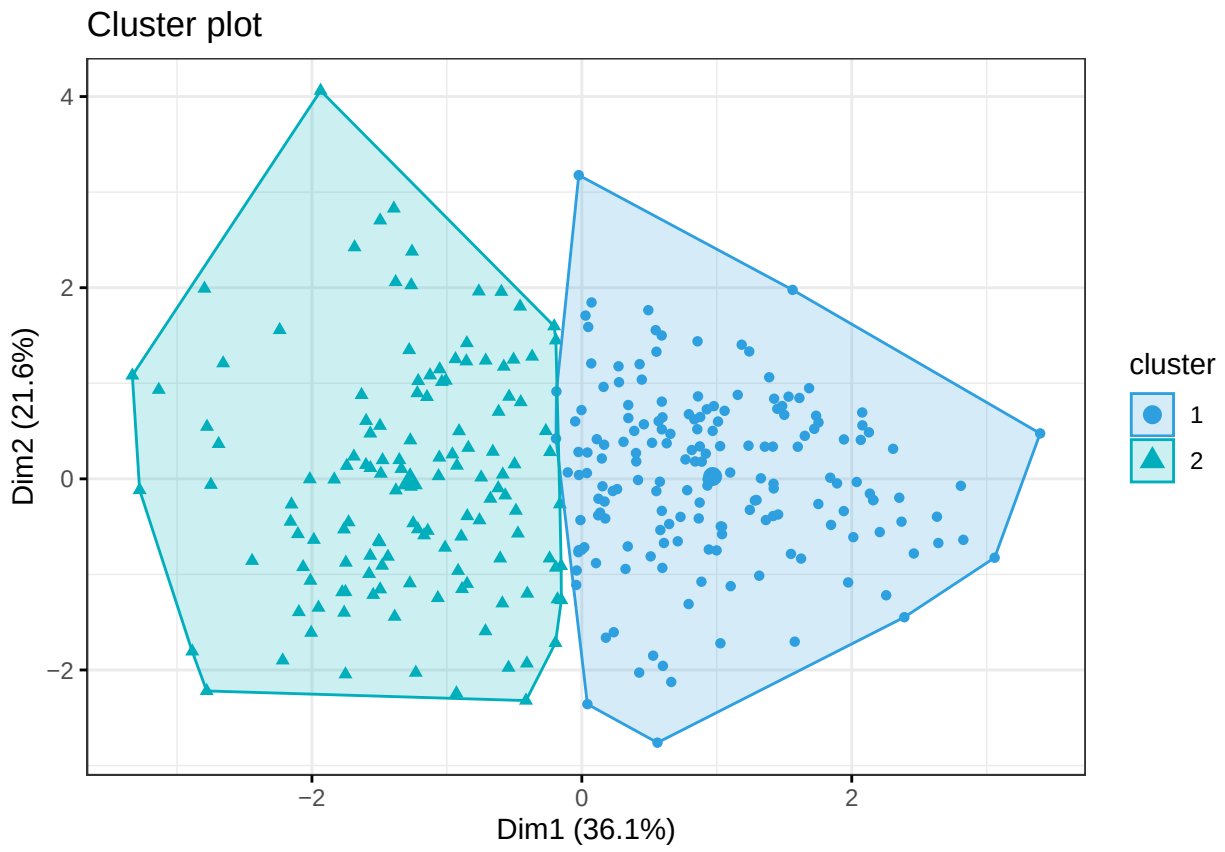
```
kable(head(res.km$cluster),caption="Risultato cluster k-means")
```

Table 10: Risultato cluster k-means

x
2
1
1
1
1
1

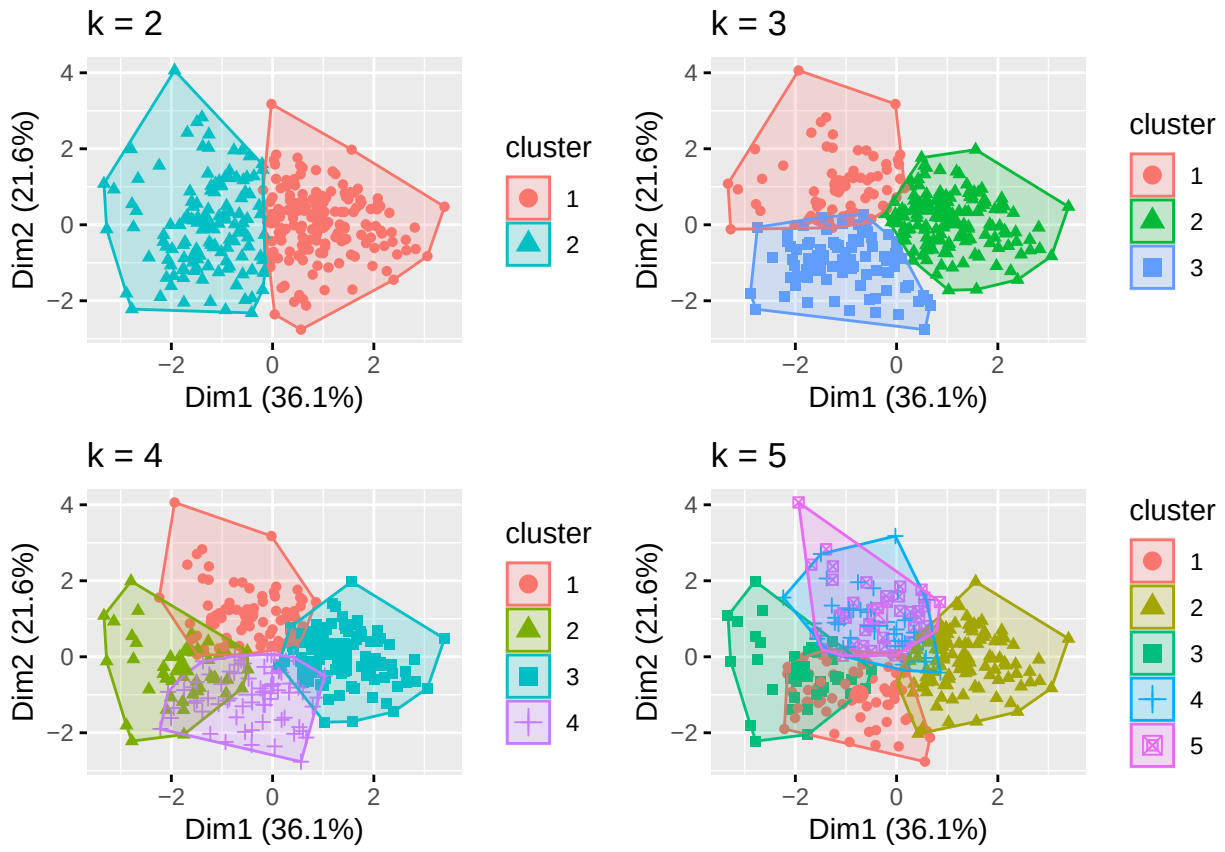
Si utilizza la funzione **fviz_cluster()** per visualizzare in maniera più immediata i cluster prodotti da k-means.

```
fviz_cluster(res.km, data = data.new,
             palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
             )
```



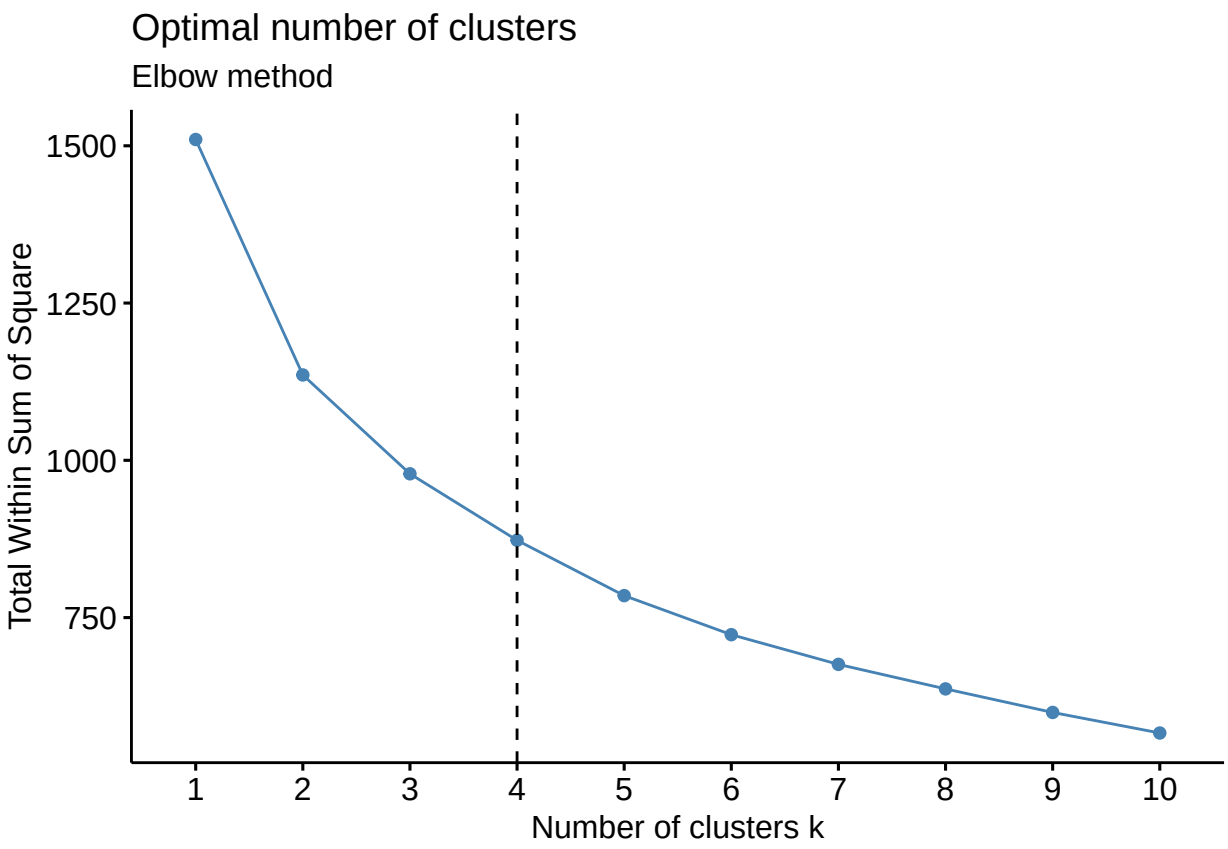
Si prova ora comparare i risultati con un numero di cluster pari a 2,3,4,5.

```
set.seed(123)
k3<- kmeans(uns_df, 3, nstart = 25)
k4<- kmeans(uns_df, 4, nstart = 25)
k5<- kmeans(uns_df, 5, nstart = 25)

p1<-fviz_cluster(res.km, data = data.new,geom = "point" )+
  ggtitle("k = 2")

p2<-fviz_cluster(k3, data = data.new,geom = "point")+
  ggtitle("k = 3")

p3<-fviz_cluster(k4, data = data.new,geom = "point")+
  ggtitle("k = 4")

p4<-fviz_cluster(k5, data = data.new,geom = "point")+
  ggtitle("k = 5")

grid.arrange(p1,p2,p3,p4, nrow = 2)
```



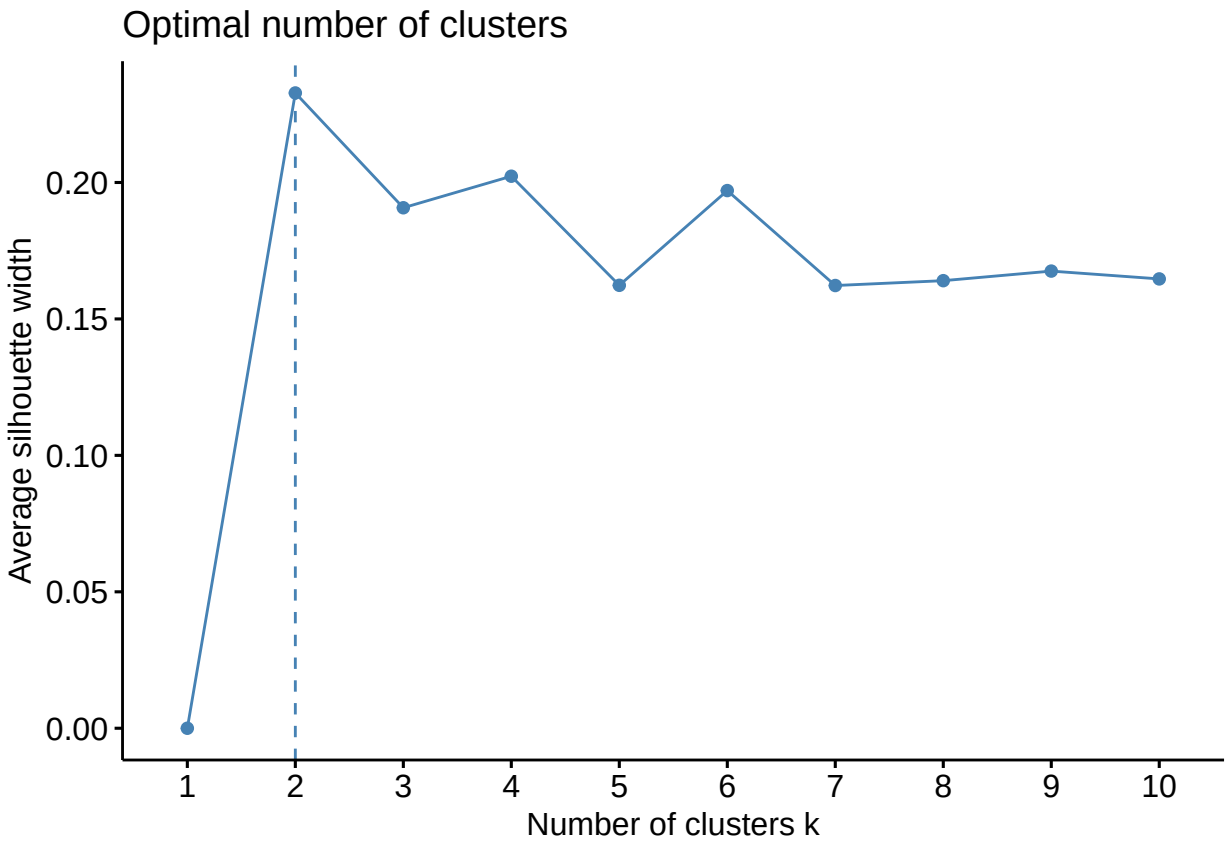
Mediante la funzione **fviz_nbclust()** si va a stimare il numero ottimale di cluster.

```
fviz_nbclust(uns_df, kmeans, nstart = 25, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```



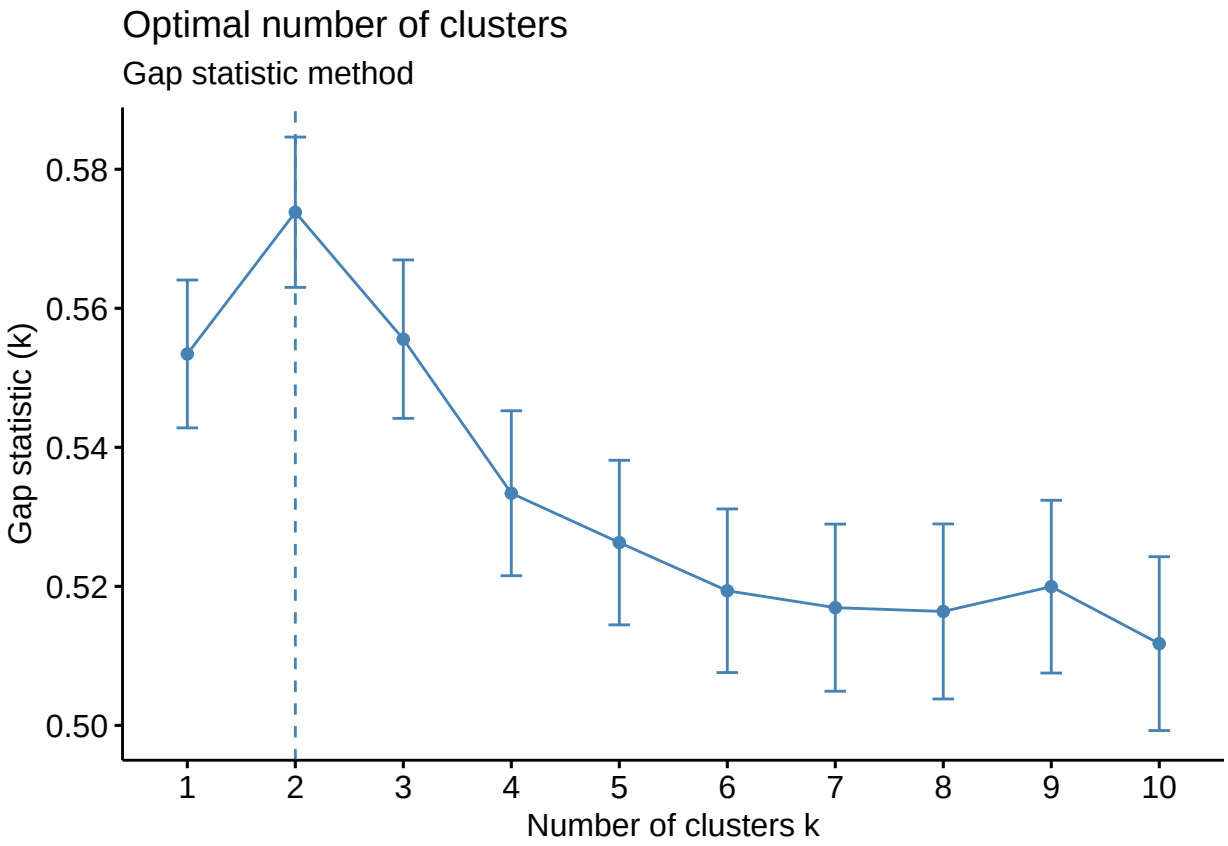
Utilizzando "Elbow method" ci viene suggerito un numero ottimale di cluster pari a 4.

```
fviz_nbclust(uns_df, kmeans, method = "silhouette")
```



Utilizzando “Silhouette method” ci viene suggerito un numero ottimale di cluster pari a 2.

```
set.seed(123)
fviz_nbclust(uns_df, kmeans, nstart = 25,  method = "gap_stat", nboot = 50)+
  labs(subtitle = "Gap statistic method")
```



Il Gap statistic method ci suggerisce un numero di cluster pari a 2.

Quindi in accordo con le osservazioni e per la regola della maggioranza possiamo dire che il numero di cluster ottimale è 2.

Si ripropone quindi l'algoritmo k-means con k=2 come suggerito dai metodi precedentemente utilizzati.

```
set.seed(123)
final <- kmeans(uns_df, 2, nstart = 25)
kable(head(final$cluster,3),caption="Risultato k=2")
```

Table 11: Risultato k=2

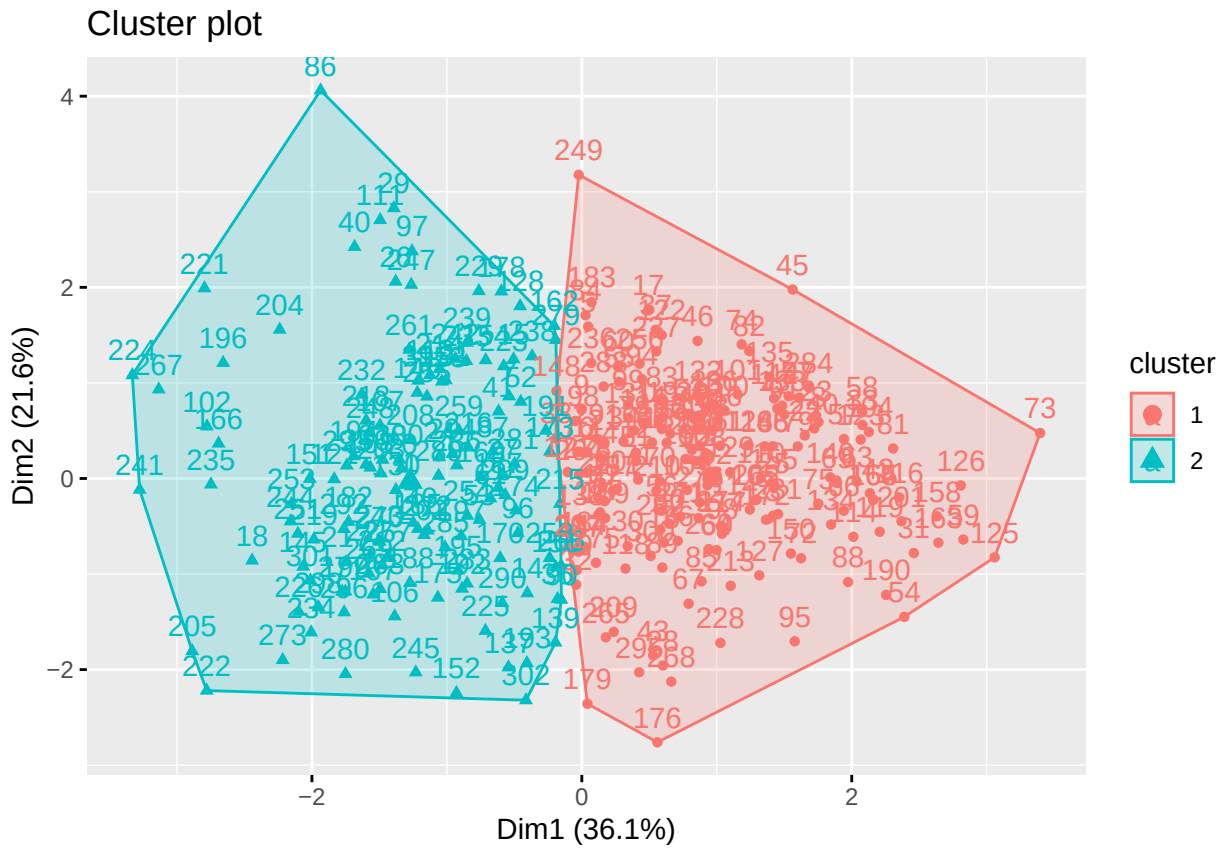
x
2
1
1

```
kable(head(final$centers),caption="Cluster means")
```

Table 12: Cluster means

age	trestbps	chol	thalach	oldpeak
-0.5415776	-0.3355766	-0.2416645	0.5202796	-0.4537289
0.7110790	0.4406044	0.3173000	-0.6831152	0.5957356

```
fviz_cluster(final, data = uns_df)
```



Cluster Validation

Per misurare la bontà dei risultati ottenuti dal clustering gerarchico e kmeans e soprattutto determinare il numero ottimale di cluster, prendiamo in considerazione le seguenti statistiche di validazione del clustering.

```
rownames(uns_df) <- rownames(data)
```

```
clmethods <- c("hierarchical","kmeans")
intern <- clValid(uns_df, nClust = 2:6,
                  clMethods = clmethods, validation = "internal")
```

Dalle misure di validazione interna, vediamo che il numero ottimale di cluster è 2.

```
summary(intern)
```

```
##
## Clustering Methods:
##  hierarchical kmeans
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2        3        4        5        6
##
## hierarchical Connectivity    2.9290  13.8306  16.7595  19.7996  30.3246
##              Dunn            0.4246   0.2168   0.2168   0.2164   0.2048
##              Silhouette      0.5508   0.3682   0.2901   0.1924   0.1630
## kmeans       Connectivity   72.4024 115.7575 147.9873 162.7841 173.9298
##              Dunn            0.0614   0.0638   0.0378   0.0605   0.0732
##              Silhouette      0.2418   0.2161   0.1857   0.2060   0.1933
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 2.9290 hierarchical 2
## Dunn         0.4246 hierarchical 2
## Silhouette   0.5508 hierarchical 2
```

Calcoliamo anche le misure di stabilità, e possiamo dire che i risultati sono fortemente stabili

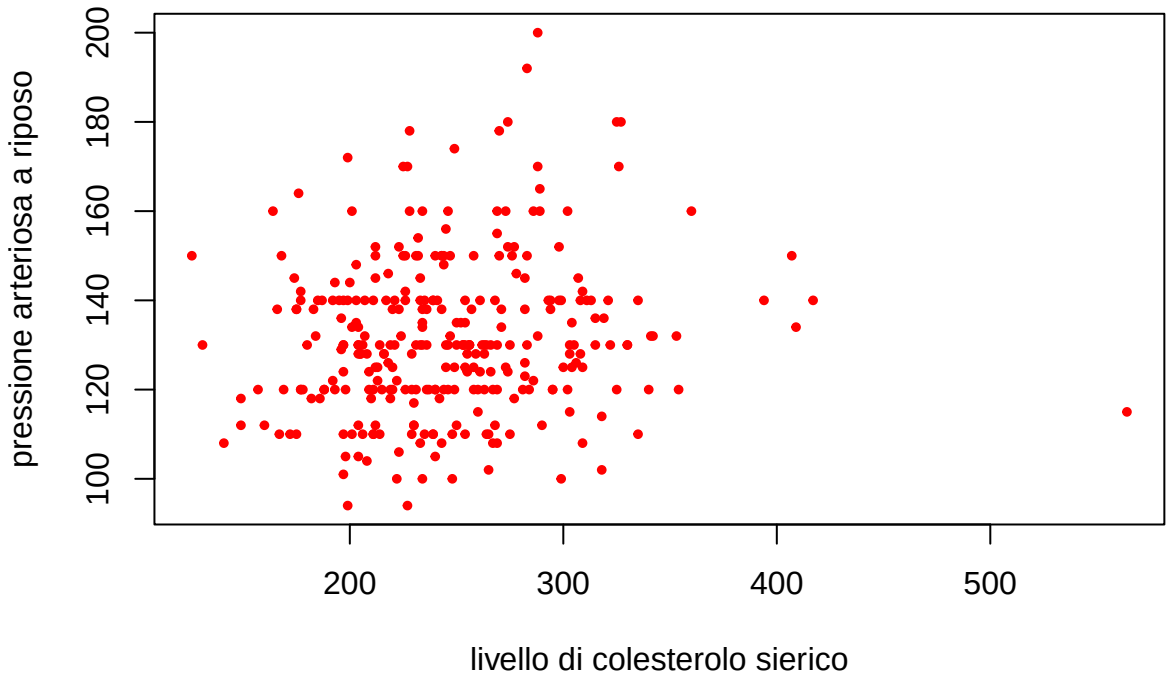
```
clmethods <- c("hierarchical","kmeans")
stab <- clValid(uns_df, nClust = 2:6, clMethods = clmethods,
                validation = "stability")
```

```
summary(stab)
```

```
##
## Clustering Methods:
##  hierarchical kmeans
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                         2      3      4      5      6
##
## hierarchical APN  0.0233 0.0200 0.0494 0.1353 0.2177
##              AD   2.9362 2.8821 2.8661 2.8556 2.8158
##              ADM  0.0974 0.1209 0.1845 0.3522 0.5247
##              FOM  0.9998 0.9991 0.9971 0.9894 0.9908
## kmeans       APN  0.1968 0.2638 0.3642 0.3510 0.4291
##              AD   2.6630 2.5309 2.5013 2.3705 2.3563
##              ADM  0.5206 0.6373 0.9388 0.8253 0.9310
##              FOM  0.9590 0.9572 0.9544 0.9535 0.9495
##
## Optimal Scores:
##
##     Score  Method       Clusters
## APN 0.0200 hierarchical 3
## AD  2.3563 kmeans       6
## ADM 0.0974 hierarchical 2
## FOM 0.9495 kmeans       6
```

FRM

Utilizzando i *mixture regression models*, si mette in relazione la pressione arteriosa a riposo con il colesterolo sierico nel sangue.



Si può notare dal grafico che c'è presenza di eteroschedasticità. Facendo una regressione lineare:

```
z <- lm(trestbps ~  chol)
summary(z)
```

```
##
## Call:
## lm(formula = trestbps ~ chol)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.821 -11.259  -1.946   9.513  66.637
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 121.35976    4.87052  24.917   <2e-16 ***
## chol          0.04168    0.01935   2.153   0.0321 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.43 on 301 degrees of freedom
```

```
## Multiple R-squared:  0.01517,    Adjusted R-squared:  0.0119
## F-statistic: 4.637 on 1 and 301 DF,  p-value: 0.03208
```

Si nota come il coefficiente della variabile del colesterolo sierico sia statisticamente significativo e abbia un effetto positivo sulla pressione arteriosa a riposo.

```
sick.fmr<-stepFlexmix(trestbps ~ chol ,  k=2, nrep=10)
```

```
## 2 : * * * * * * * * * *
```

```
summary(sick.fmr)
```

```
##
## Call:
## stepFlexmix(trestbps ~ chol, k = 2, nrep = 10)
##
##        prior size post>0  ratio
## Comp.1 0.856  276    302 0.9139
## Comp.2 0.144   27    302 0.0894
##
## 'log Lik.' -1281.949 (df=7)
## AIC: 2577.898   BIC: 2603.894
```

```
parameters(sick.fmr)
```

```
##                        Comp.1      Comp.2
## coef.(Intercept) 120.84415674 109.6399314
## coef.chol          0.02684755   0.1907219
## sigma             13.51866959  15.9511329
```

```
post.pres.chol.fmr<-posterior(sick.fmr)
head(post.pres.chol.fmr)
```

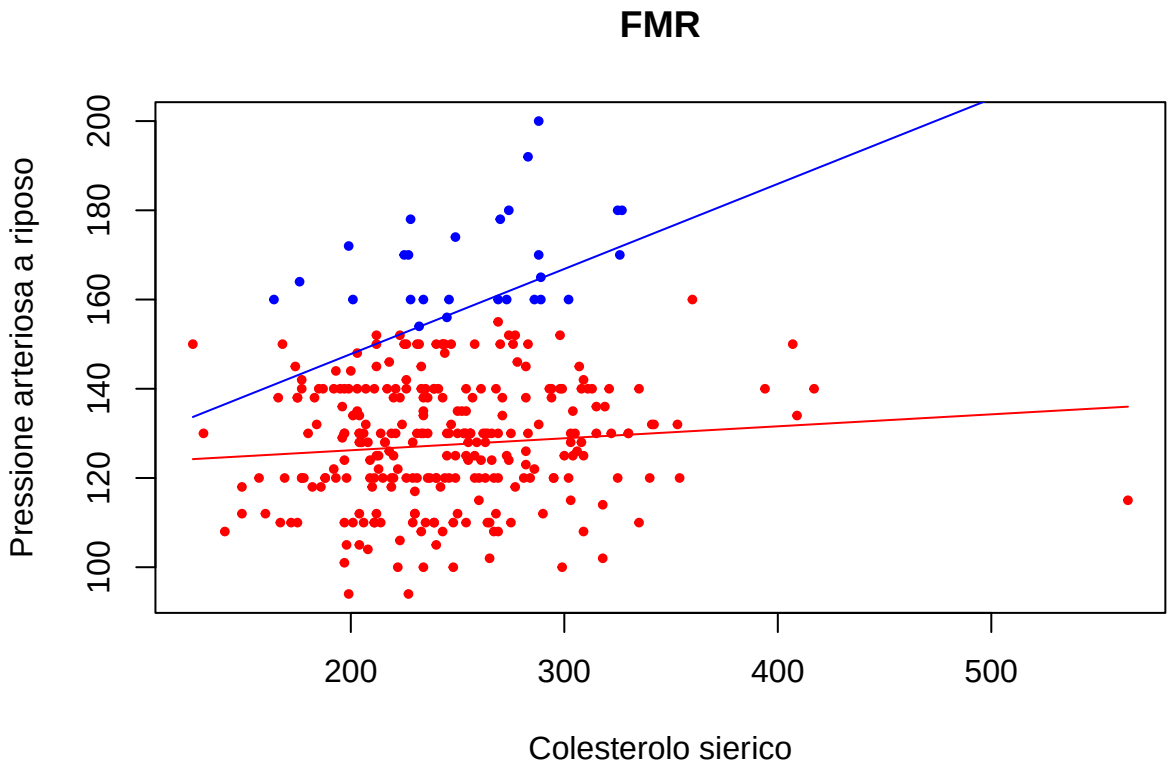
```
##           [,1]         [,2]
## [1,] 0.7730993 0.2269006697
## [2,] 0.9674228 0.0325772475
## [3,] 0.9295345 0.0704655024
## [4,] 0.9846163 0.0153837174
## [5,] 0.9996864 0.0003135791
## [6,] 0.8147602 0.1852398205
```

```
pred.pres.chol.fmr<-clusters(sick.fmr)
pred.pres.chol.fmr
```

```
##   [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
## [149] 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [223] 1 2 1 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1
## [260] 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1
## [297] 1 2 1 1 1 1 1
```

Da ciò emerge, dopo aver ripetuto l'algoritmo dieci volte e aver individuato due gruppi, in relazione anche a quanto trovato tramite k-means, che un gruppo è molto più numeroso del secondo.

```
xmin <- min(chol)
xmax <- max(chol)
FMR1<-parameters(sick.fmr)[1:2,1]
FMR2<-parameters(sick.fmr)[1:2,2]
plot(chol,trestbps, col=c("red","blue")[as.numeric(pred.pres.chol.fmr)], cex=0.8, pch=20,
     xlab="Colesterolo sierico", ylab="Pressione arteriosa a riposo", main="FMR")
curve(FMR1[1]+FMR1[2]*x,col="red",add=TRUE)
curve(FMR2[1]+FMR2[2]*x,col="blue",add=TRUE)
```



Per il gruppo più numeroso, il colesterolo sierico ha un effetto positivo bassissimo sulla pressione arteriosa, al contrario di ciò che succede nel secondo gruppo, anche se quest'ultimo presenta poche osservazioni.

Anche pesando la covariata, si ottengono gli stessi risultati:

```
sick.fmrc<-stepFlexmix(trestbps ~ chol ,  k=2, nrep=10, concomitant = FLXPmultinom(~ chol))
```

```
## 2 : * * * * * * * * * *
```

```
summary(sick.fmrc)
```

```
##
## Call:
## stepFlexmix(trestbps ~ chol, concomitant = FLXPmultinom(~chol),
##     k = 2, nrep = 10)
##
##        prior size post>0 ratio
## Comp.1 0.221   44    302 0.146
## Comp.2 0.779  259    302 0.858
##
## 'log Lik.' -1281.738 (df=8)
## AIC: 2579.475   BIC: 2609.185
```

```
parameters(sick.fmrc)
```

```
##                      Comp.1       Comp.2
## coef.(Intercept) 97.3651567 118.54265985
## coef.chol         0.2206794   0.03243361
## sigma            16.9319645  13.06760760
```

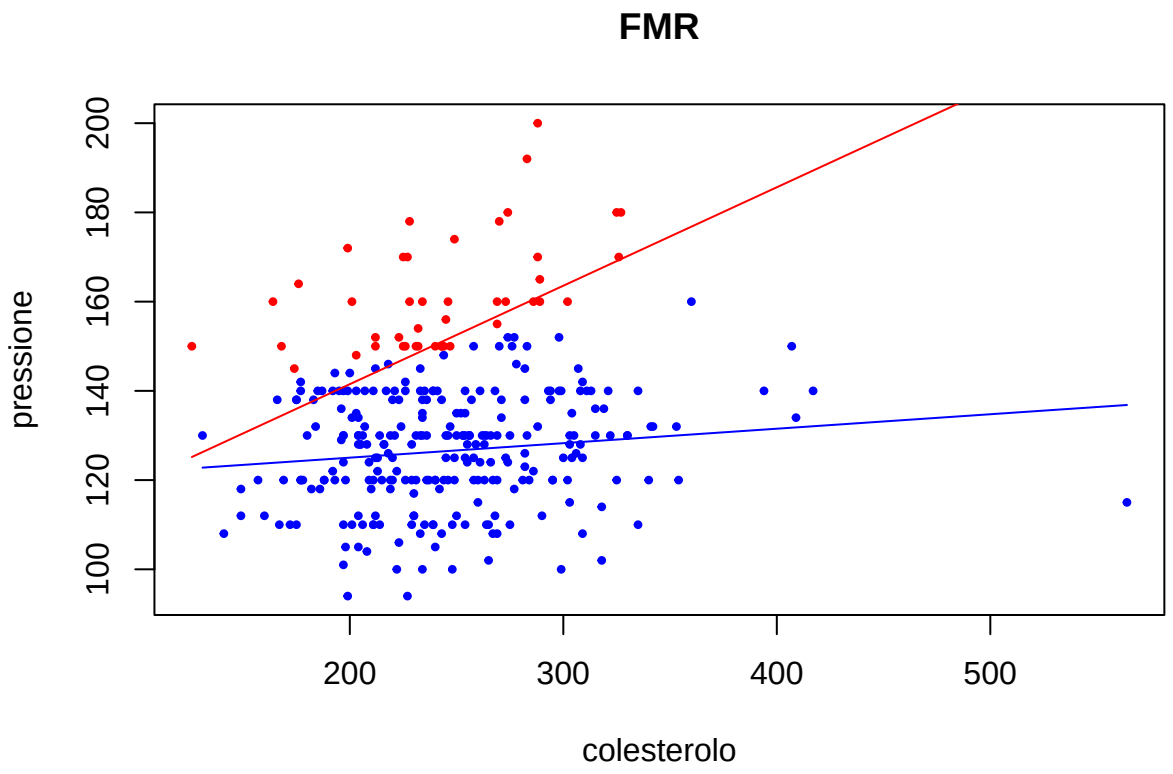
```
post.pres.chol.fmrc<-posterior(sick.fmrc)
head(post.pres.chol.fmrc)
```

```
##           [,1]      [,2]
## 1 0.3920232011 0.6079768
## 2 0.0816935636 0.9183064
## 3 0.1859910753 0.8140089
## 4 0.0532327704 0.9467672
## 5 0.0006778772 0.9993221
## 6 0.3721734334 0.6278266
```

```
pred.pres.chol.fmrc<-clusters(sick.fmrc)
pred.pres.chol.fmrc
```

```
##   [1] 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 1 1 2 2 2 2 1 2 1 1 2 2 2 2 2 2 2 2 2 2
##  [38] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1
## [112] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1
## [149] 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1
## [186] 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [223] 2 1 2 1 2 2 1 2 2 1 1 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 1 1 2 2 2 2 2 1 2 2 2 1
## [260] 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 1 2 1 1 2 2
## [297] 2 1 2 2 2 2 2
```

```
x1 <-chol[pred.pres.chol.fmrc==1]
x2 <-chol[pred.pres.chol.fmrc==2]
FMRC1 <- parameters(sick.fmrc)[1:2,1]
FMRC2 <- parameters(sick.fmrc)[1:2,2]
plot(chol, trestbps, col=c("red","blue")[as.numeric(pred.pres.chol.fmrc)], pch=20, cex=0.7,xlab="coleste
curve(FMRC1[1]+FMRC1[2]*x, c(min(x1), max(x1)),col="RED",add=TRUE)
curve(FMRC2[1]+FMRC2[2]*x, c(min(x2), max(x2)),col="BLUE",add=TRUE)
```



LINEAR DISCRIMINANT ANALYSIS

A questo punto si svolge un'analisi sulla linear discriminant analysis, cercando di separare e classificare al meglio i diversi soggetti malati di cuore e quelli non malati.
All'occorrenza cerchiamo di classificare i malati e non malati secondo la pressione arteriosa a riposo, il colesterolo sierico nel sangue, la frequenza cardiaca massima raggiunta, la depressione ST indotta dall'esercizio rispetto al riposo e l'età.

```
lda.fit=lda(target~chol+age+thalach+trestbps+oldpeak)
lda.fit
```

```
## Call:
## lda(target ~ chol + age + thalach + trestbps + oldpeak)
##
## Prior probabilities of groups:
##         0         1
## 0.4554455 0.5445545
##
## Group means:
##       chol      age  thalach trestbps   oldpeak
## 0 251.0870 56.60145 139.1014 134.3986 1.5855072
## 1 242.2303 52.49697 158.4667 129.3030 0.5830303
##
## Coefficients of linear discriminants:
##                    LD1
## chol     -0.002413477
## age      -0.002060067
## thalach   0.030054553
## trestbps -0.007848707
## oldpeak  -0.590010556
```

Si possono osservare le medie nei due gruppi. Vediamo che l'età media nel gruppo dei malati è di poco più bassa rispetto ai non malati e ovviamente la frequenza massima raggiunta è più alta per chi ha una malattia cardiaca rispetto a chi non la presenta. Si fissa il tutto nella matrice di confusione e si commentano i gruppi:

```
confMat.LDA<-addmargins(table(pred.class,target))
confMat.LDA
```

```
##           target
## pred.class   0   1 Sum
##        0    90  32 122
##        1    48 133 181
##        Sum 138 165 303
```

Si può constatare che l'errore di misclassificazione è del 26%. I falsi positivi sono 48 ovvero circa il 27 % dei positivi, mentre i falsi negativi sono 32 su 122, ovvero il 26%. L'errore di primo tipo è del 34,78%, mentre quello di secondo tipo è del 19,39%. La sensitività è dell'80,61%, mentre la specificità è del 65,22%. Ad ogni modo, una sensitività dell'80% può considerarsi "soddisfacente". E' vero anche che trattando di malattia cardiaca, forse ci si dovrebbe aspettare una sensitività molto più alta, quasi prossima al 99%.

A questo punto si calcola l'accuratezza del modello:

```
delta.LDA=(confMat.LDA[1,2]+confMat.LDA[2,1])/N*100
delta.LDA
```

```
## [1] 26.40264
```

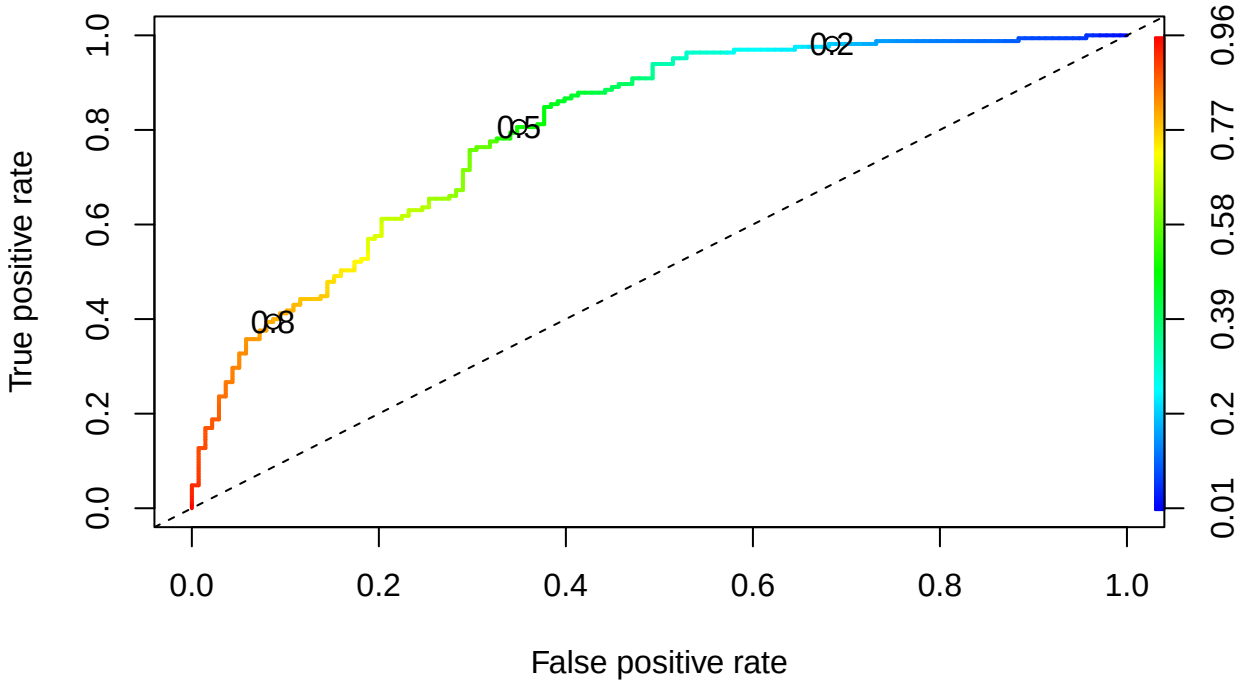
```
accuracy.LDA=100-delta.LDA
accuracy.LDA
```

```
## [1] 73.59736
```

Si trova che l'accuratezza è intorno al 73,6%. Dato il caso studio, un'accuratezza che si attesta intorno al 74% non la si può considerare ottimale. Questo considerando anche un errore di primo e secondo tipo elevati.

CURVA ROC LDA

```
pred.t.LDA=prediction(lda.predict$posterior[,2], target)
perf.t.LDA=performance(pred.t.LDA, measure = "tpr", x.measure = "fpr")
plot(perf.t.LDA,colorize=TRUE,lwd=2, print.cutoffs.at=c(0.2,0.5,0.8))
abline(a=0,b=1, lty=2)
```



Si calcola l'area al di sotto della curva ROC:

```
performance(pred.t.LDA, measure = "auc", x.measure = "fpr")@y.values
```

```
## [[1]]
## [1] 0.8007905
```

In accordo con quanto ottenuto finora, l'area al di sotto della curva roc è di circa l'80%.

LR

Volendo fare un confronto con la logistic regression:

```
glm.fit2=glm(target ~ chol+age+thalach+trestbps+oldpeak, family=binomial, data=data.new)
summary(glm.fit2, digits=3)
```

```
##
## Call:
## glm(formula = target ~ chol + age + thalach + trestbps + oldpeak,
##     family = binomial, data = data.new)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1760  -0.8127   0.4934   0.8608   2.4040
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.964169   1.754389  -1.120    0.263
## chol        -0.003086   0.002704  -1.141    0.254
## age         -0.001224   0.017409  -0.070    0.944
## thalach      0.034568   0.007288   4.743  2.1e-06 ***
## trestbps    -0.011122   0.008406  -1.323    0.186
## oldpeak     -0.713136   0.141772  -5.030  4.9e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 322.07  on 297  degrees of freedom
## AIC: 334.07
##
## Number of Fisher Scoring iterations: 4
```

Si può constatare che molte variabili perdono di significatività.

```
glm.predict2=predict.glm(glm.fit2,data.new)
glm.probs2=predict.glm(glm.fit2,type="response")
glm.pred2=rep
glm.predict2=predict.glm(glm.fit2,data)
```

```
glm.probs2=predict.glm(glm.fit2,type="response")
glm.pred2=rep("No",N)
glm.pred2[glm.probs2>.5]="Yes"
```

```
confMat.LR<-addmargins(table(glm.pred2,target))
confMat.LR
```

```
##          target
## glm.pred2   0   1 Sum
##       No   91  35 126
##       Yes  47 130 177
##       Sum 138 165 303
```

Si può notare che aumenta di 3 unità l'errore di secondo tipo, mentre diminuisce di una unità l'errore di primo tipo. I risultati trovati rispecchiano quanto ottenuto anche con lda.

Anche qui si calcola l'accuratezza del modello:

```
delta.LR=(confMat.LR[1,2]+confMat.LR[2,1])/N*100
delta.LR
```

```
## [1] 27.06271
```

```
accuracy.LR=100-delta.LR
accuracy.LR
```

```
## [1] 72.93729
```

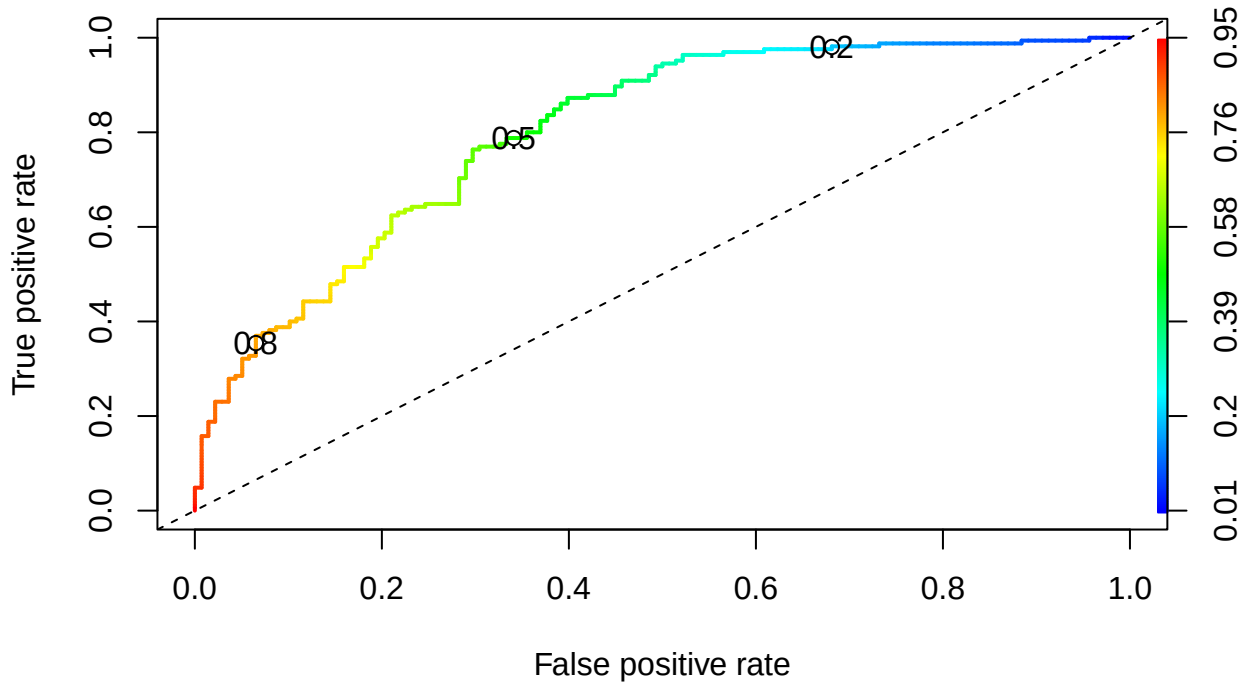
È più bassa rispetto che al modello con LDA. Soprattutto perché aumenta di poco l'errore di secondo tipo e diminuisce di altrettanto poco l'errore di primo tipo, è più accurato il modello stimato con LDA.

La curva ROC per il modello LR

```
contrasts(target)
```

```
##   1
## 0 0
## 1 1
```

```
pred.t.LR=prediction(glm.probs2, target)
perf.t.LR=performance(pred.t.LR, measure = "tpr", x.measure = "fpr")
plot(perf.t.LR,colorize=TRUE,lwd=2)
plot(perf.t.LR,colorize=TRUE,lwd=2, print.cutoffs.at=c(0.2,0.5,0.8))
abline(a=0,b=1, lty=2)
```




```
performance(pred.t.LR, measure = "auc", x.measure = "fpr")@y.values
```

```
## [[1]]
## [1] 0.8017128
```

l'area al di sotto della curva ROC, anche in questo caso, è dell'80%.

Si prova ad abbassare la soglia di cut-off e si analizzano i risultati.

```
glm.fit3=glm(target ~ chol+age+thalach+trestbps+oldpeak, family=binomial, data=data.new)
summary(glm.fit3, digits=3)
```

```
##
## Call:
## glm(formula = target ~ chol + age + thalach + trestbps + oldpeak,
##     family = binomial, data = data.new)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1760  -0.8127   0.4934   0.8608   2.4040
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.964169   1.754389  -1.120    0.263
## chol        -0.003086   0.002704  -1.141    0.254
## age         -0.001224   0.017409  -0.070    0.944
## thalach      0.034568   0.007288   4.743  2.1e-06 ***
## trestbps    -0.011122   0.008406  -1.323    0.186
## oldpeak     -0.713136   0.141772  -5.030  4.9e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 417.64  on 302  degrees of freedom
## Residual deviance: 322.07  on 297  degrees of freedom
## AIC: 334.07
##
## Number of Fisher Scoring iterations: 4
```

```
glm.predict3=predict.glm(glm.fit3,data.new)
glm.probs3=predict.glm(glm.fit3,type="response")
glm.pred3=rep("No",N)
glm.pred3[glm.probs3>.3]="Yes"
```

```
confMat.LR2<-addmargins(table(glm.pred3,target))
confMat.LR2
```

```
##          target
## glm.pred3   0   1 Sum
##       No   66   6  72
##       Yes  72 159 231
##       Sum 138 165 303
```

Si può vedere come, abbassando la soglia, diminuisca la percentuale di falsi negativi, ma aumenta al 52,17% quella di falsi positivi. In questo caso, bisognerebbe confrontarsi con gli esperti in materia per stabilire le priorità sugli errori. Ovvero, prediligere un minore errore di secondo tipo, quindi meno falsi negativi, oppure un minore errore di primo tipo, quindi meno falsi positivi.

```r
delta.LR2=(confMat.LR2[1,2]+confMat.LR2[2,1])/N*100
delta.LR2
```

```
## [1] 25.74257
```

```r
accuracy.LR2=100-delta.LR2 # accuracy
accuracy.LR2
```

```
## [1] 74.25743
```

L'accuratezza risulta essere del 74,26 %, maggiore rispetto a quella derivante dalla LDA e dal LR con una soglia al 50%. In ambito medico, sottoporre a cura un paziente che in realtà non è malato, potrebbe essere molto pericoloso. Dunque, si potrebbe prediligere un errore di primo tipo più basso, anche a fronte di un'accuratezza più elevata.

QDA

Infine, si stima il modello tramite la QDA, anche se le osservazioni totali risultano essere 303.

```
qda.fit=qda(target ~ chol+age+thalach+trestbps+oldpeak,data=data.new)
qda.fit
```

```
## Call:
## qda(target ~ chol + age + thalach + trestbps + oldpeak, data = data.new)
##
## Prior probabilities of groups:
##         0         1
## 0.4554455 0.5445545
##
## Group means:
##       chol      age  thalach trestbps   oldpeak
## 0 251.0870 56.60145 139.1014 134.3986 1.5855072
## 1 242.2303 52.49697 158.4667 129.3030 0.5830303
```

```
qda.class=predict(qda.fit,target)$class
table(qda.class,target)
```

```
##          target
## qda.class   0   1
##         0  89  25
##         1  49 140
```

In questo caso, la percentuale di falsi postivi rimane pressoché uguale rispetto al modello stimato con GLM e con la LDA, diminuisce di 10 unità il numero di falsi negativi. In particolare, la sensitività del test aumenta all'84,85%. Si analizza anche l'accuratezza

```
accuracy.QDA<-sum(diag(table(qda.class,target)))/N*100
accuracy.QDA
```

```
## [1] 75.57756
```

Si trova dunque una percentuale di accuratezza del 75,58% maggiore rispetto agli altri due modelli. Anche in questo caso l'accuratezza non può essere considerata soddisfacente per via del problema affrontato.