Laboratorio 3

Distribuzioni e studi di simulazione

3.1 Distribuzione normale

R consente di calcolare automaticamente densità, funzione di ripartizione e quantili per molte distribuzioni di interesse. È inoltre possibile generare realizzazioni pseudo-casuali dalle stesse distribuzioni Ad esempio, consideriamo la distribuzione normale standardizzata. Esistono 4 funzioni ad essa relative:

- `dnorm(x)` calcola il valore della densità in <x>;
- `pnorm(q)` calcola il valore della ripartizione in <q>;
- `qnorm(p)` calcola il quantile di livello <p>;
- `rnorm(n)` genera un campione da una normale standard di dimensione <n>.

Ad esempio, per ottenere il grafico della funzione di ripartizione di una normale standard:

```
> curve(pnorm(x),from=-5,to=5,n=100)
```

Ovviamente, possono essere trattate anche distribuzioni normali non standardizzate: è sufficiente aggiungere nell'ordine la media e la deviazione standard nelle chiamate sopra viste.

```
> args(pnorm)
function (q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
> curve(pnorm(x,mean=2,sd=0.7),add=T,col=2)
```

Alcune delle distribuzioni disponibili sono:

R	Distribuzione	Parametri	Defaults
chisq	chi-quadrato	df	-
exp	exponenziale	rate	1
f	F	df1, df2	-, -
gamma	Gamma	shape, scale	-, 1
lnorm	log-normale	meanlog, sdlog	0, 1
norm	normale	mean, sd	0, 1
t	t di Student	df	-
unif	uniforme	min, max	0, 1

3.2 Adattamento ad una distribuzione normale

Consideriamo un campione di numerosità 10 generato da una distribuzione normale standard:

```
> x <- rnorm(10)
```

Supponiamo di non sapere che il campione proviene da una popolazione normale. Proviamo a studiare graficamente la distribuzione dei dati per capire da che distribuzione deriva.

```
> hist(x)
> hist(x, nclass=8)
> hist(x, nclass=15)
> boxplot(x)
```

Aumentiamo la numerosità:

```
> xx <- rnorm(100)
> hist(xx)
> boxplot(xx)
```

Che strumenti grafici abbiamo a disposizione per vedere se proviene o meno da una distribuzione normale?

```
> qqnorm(xx)
> qqline(xx)
```

Esercizio: provare a verificare la normalità di x. Che cosa si puo' dire?

Per rendersi conto dell'aspetto del `qqnorm` quando i dati non sono normali, generiamo dei dati da una distribuzione "simile" alla normale e da una completamente diversa e poi confrontiamoli:

```
> y <- rt(100,2)
> hist(y)
> qqnorm(y)
> qqline(y)

> z <- rexp(100)
> hist(z)
> qqnorm(z)
> qqline(z)

> par(mfrow=c(1,2))
> qqnorm(xx)
> qqline(xx)
> qqnorm(y)
> qqline(y)
> qqnorm(xx)
> qqline(xx)
> qqnorm(z)
> qqline(z)
> par(mfrow=c(1,1))
```

Per verificare l'adattamento dei dati ad una distribuzione diversa dalla normale, il comando `qqnorm` non può essere usato. Un grafico analogo può essere ottenuto con il comando (es. nel caso della distribuzione esponenziale):

```
> qqplot(qexp(ppoints(z)), sort(z))
> abline(0,1)
```

3.3 Simulazione della distribuzione degli stimatori nel modello di regressione lineare semplice normale

Per studiare la distribuzione di $(\hat{\beta}_1, \hat{\beta}_2)$ in un modello di regressione lineare semplice normale, costruiamo artificialmente delle realizzazioni da un modello di questo tipo.

```
> x <- 1:30
> error <- rnorm(30, mean=0, sd=4)
> y <- 5 + 3*x + error
```

Proviamo a costruire il diagramma di dispersione:

```
> plot(x, y)
```

Ovviamente, la linearità della relazione è evidente!

La stima dei parametri della regressione $\mathtt{y} = \beta_1 + \beta_2 \mathtt{x}$ sono:

```
> beta2 <- cov(x,y)/var(x)
> beta2
[1] 3.071235
> beta1 <- mean(y) - beta2*mean(x)
> beta1
[1] 5.148649
```

Supponiamo ora di aumentare la varianza dell'errore:

```
> error <- rnorm(30, mean=0, sd=10)
> y <- 5 + 3*x + error
> plot(x, y)
```

I punti sono meno allineati di prima.

Esercizio: ripetere l'analisi aumentando ancora la varianza del termine d'errore. Valutare gli effetti dell'aumentata variabilità. Provare a vedere cosa succede se si diminuisce la varianza.
Supponiamo ora di generare un campione di 30 elementi, con $\beta_1 = 5$, $\beta_2 = 3$ e $\sigma^2 = 4^2 = 16$.

```
> x <- 1:30
> error <- rnorm(30, mean=0, sd=4)
> y <- 5 + 3*x + error
```

La stima di β_2 è pari a

```
> beta2 <- cov(y,x)/var(x)
```

Se generiamo un campione diverso otteniamo un valore diverso della stima. Se generiamo 1000 campioni (con gli stessi valori dei parametri) e registriamo le corrispondenti stime di β_2, possiamo poi valutare (empiricamente) la distribuzione dello stimatore $\hat{\beta}_2$.

Rifacciamo per 1000 volte (non a mano!) il procedimento sopra elencato.

```
> beta2.sim <- vector("numeric", length=1000)

> for (i in 1:1000)
+ {
+       error <- rnorm(30, mean=0, sd=4)
+       y <- 5 + 3*x + error
+       beta2.sim[i] <- cov(y,x)/var(x)
+ }
```

Abbiamo prima definito un vettore `beta2.sim` in cui registreremo i valori delle stime. Utilizziamo poi il ciclo `for (i in 1:1000)` { *operazioni* }. L'indice `i` assumerà valori da 1 a 1000 e per ognuno di questi valori eseguirà le *operazioni* all'interno delle parentesi { }.

A questo punto il vettore `beta2.sim` conterrà le 1000 stime del parametro β_2, relative ai 1000 diversi campioni. Quindi abbiamo 1000 realizzazioni indipendenti della variabile casuale $\hat{\beta}_2$ (lo stimatore di β_2). Possiamo valutare graficamente la distribuzione di $\hat{\beta}_2$:

```
> hist(beta2.sim)
> boxplot(beta2.sim)
```

Dalla teoria sappiamo che la distribuzione di $\hat{\beta}_2$ è normale, con media β_2 e varianza $\sigma^2/(\sum_i (x_i - \bar{x})^2)$. In questo caso, β_2 è uguale a 3 e la varianza è pari a:

```
> var.beta2 <- 4^2/sum((x-mean(x))^2)
> var.beta2
[1] 0.007119021
```

Possiamo confrontare questi valori con i valori empirici delle simulazione:

```
> mean(beta2.sim)
> var(beta2.sim)
```

Possiamo confrontare la distribuzione empirica di $\hat{\beta}_2$ con quella teorica anche graficamente.

```
> hist(beta2.sim, prob=T)
> lines(density(beta2.sim))
> lines(seq(2.7, 3.3, 0.01),
+ dnorm(seq(2.7, 3.3, 0.01), 3, sqrt(var.beta2)), lwd=2)
```

Esercizio: Costruire la distribuzione simulata per lo stimatore del coefficiente β_1. Possiamo ricorrere ad uno studio di simulazione (è questo il nome ufficiale di quanto fatto prima!) anche per valutare il livello di copertura *reale* di un intervallo di confidenza di livello *nominale* 0.95.

Concentriamoci nuovamente sul coefficiente di regressione β_2.

```
> x <- 1:30
> error <- rnorm(30, mean=0, sd=4)
> y <- 5 + 3*x + error
```

L'intervallo di confidenza di livello 0.95 è dato da

$$\hat{\beta}_2 \pm \sqrt{\hat{\mathrm{Var}}(\hat{\beta}_2)} t_{n-2}(0.025),$$

dove $\hat{\mathrm{Var}}(\hat{\beta}_2)$ è la stima della varianza di $\hat{\beta}_2$ e $t_{n-2}(0.025)$ è il quantile di livello 0.025 di una t di Student con $n-2$ gradi di libertà.

```
> n <- 30
> beta2 <- cov(x,y)/var(x)
> beta1 <- mean(y) - beta2*mean(x)
> residui <- y - beta1 - beta2*x
> s2 <- sum(residui^2)/(n-2)
> var.beta2 <- s2/sum((x-mean(x))^2)
> beta2.lower <- beta2 + qt(0.025, n-2) * sqrt(var.beta2)
> beta2.upper <- beta2 - qt(0.025, n-2) * sqrt(var.beta2)
> beta2.lower
[1] 2.750327
> beta2.upper
[1] 3.170599
```

La teoria ci dice che replicando l'esperimento un numero elevato di volte, circa il 95% degli intervalli di confidenza osservati contiene il vero valore del parametro, che in questo caso è $\beta_2 = 3$. Proviamo a verificare questo punto con uno studio di simulazione.

Salviamo in un file di testo (usando un editor ASCII, ad esempio Notepad o Wordpad di Windows) le istruzioni seguenti:

```
beta2.ic <- matrix(NA, ncol=2, nrow=1000)

for (i in 1:1000)
{
   error <- rnorm(30, mean=0, sd=4)
   y <- 5 + 3*x + error
   beta2 <- cov(x,y)/var(x)
   beta1 <- mean(y) - beta2*mean(x)
   residui <- y - beta1 - beta2*x
   s2 <- sum(residui^2)/(n-2)
   var.beta2 <- s2/sum((x-mean(x))^2)
   beta2.lower <- beta2 + qt(0.025, n-2) * sqrt(var.beta2)
   beta2.upper <- beta2 - qt(0.025, n-2) * sqrt(var.beta2)
   beta2.ic[i,1] <- beta2.lower
   beta2.ic[i,2] <- beta2.upper
}
```

Generiamo, come prima, 1000 campioni diversi e su questi calcoliamo i 1000 intervalli di confidenza. Si noti che gli estremi inferiore e superiore sono memorizzati nella matrice `beta2.ic`.

Salviamo il file ad esempio con il nome `simul.R`.

Le istruzioni possono essere passate a `R` con un semplice copia-incolla oppure con l'istruzione:

```
> source('H:/.../simul.R')
```

Calcoliamo la percentuale degli intervalli che contengono il vero valore di β_2, ovvero il livello di copertura *reale*:

```
sum( (beta2.ic[,1] <= 3) & (beta2.ic[,2] >= 3) )/1000
[1] 0.949
```

Il valore è prossimo al livello di copertura *nominale* fissato!

Esercizio: Valutare il livello di copertura reale dell'intervallo di confidenza di livello nominale 0.9 per il coefficiente β_1.