

# Testing Psicologico

## Lezione 3

Filippo Gambarota

@Università di Padova

21/11/2022

# Probabilità

# Le distribuzioni di probabilità in R

Ci sono tantissime distribuzioni implementate in R, e molte altre caricando specifici pacchetti. `t`, `norm`, `gamma`, `beta`, `poisson`, `exp`, `chisq`, `binom`, ...

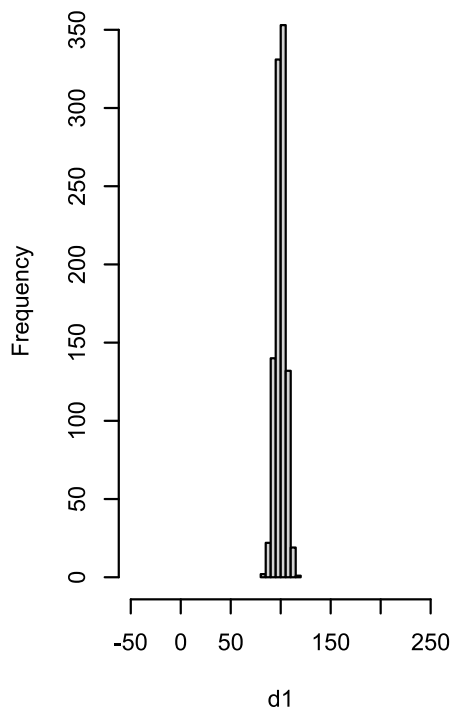
Tutte queste distribuzioni possono essere combinate con 4 funzioni: `r`, `q`, `d` e `p`:

- `rnorm`: genera numeri da una distribuzione normale (molto utile per simulare dati)
- `dnorm`: calcola la densità di probabilità
- `pnorm`: calcola la probabilità cumulata dato un quantile
- `qnorm`: calcola il quantile data la probabilità cumulata

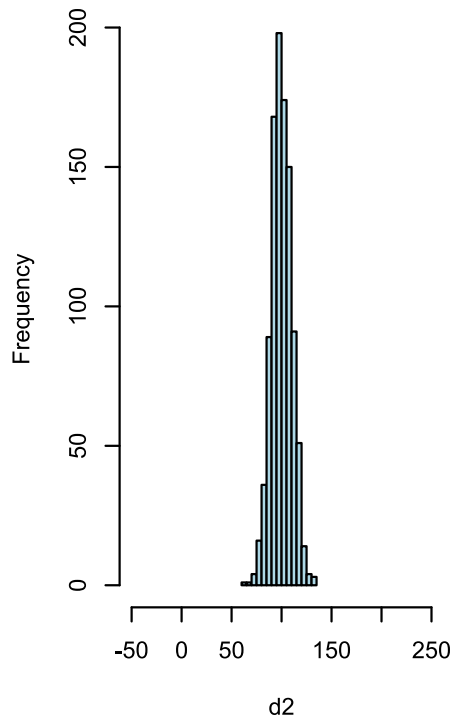
# Distribuzione Normale - `rnorm`

```
par(mfrow = c(1,3))
d1 <- rnorm(n = 1000, mean = 100, sd = 5)
d2 <- rnorm(n = 1000, mean = 100, sd = 10)
d3 <- rnorm(n = 1000, mean = 100, sd = 50)
hist(d1, xlim = c(100-3*50, 100+3*50))
hist(d2, col = "lightblue", xlim = c(100-3*50, 100+3*50))
hist(d3, col = "salmon", xlim = c(100-3*50, 100+3*50))
```

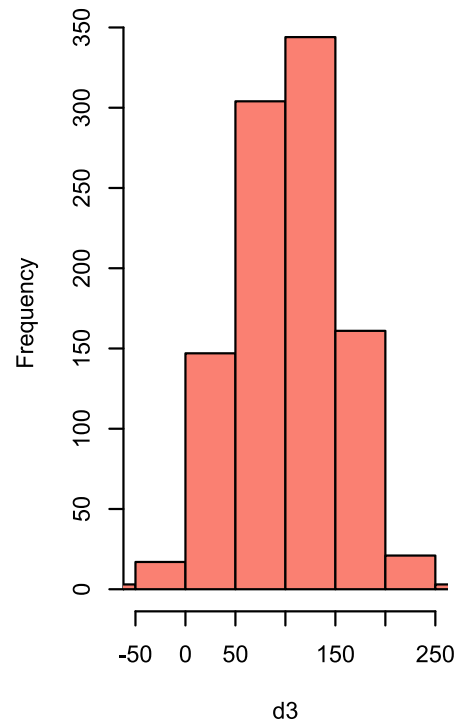
Histogram of d1



Histogram of d2

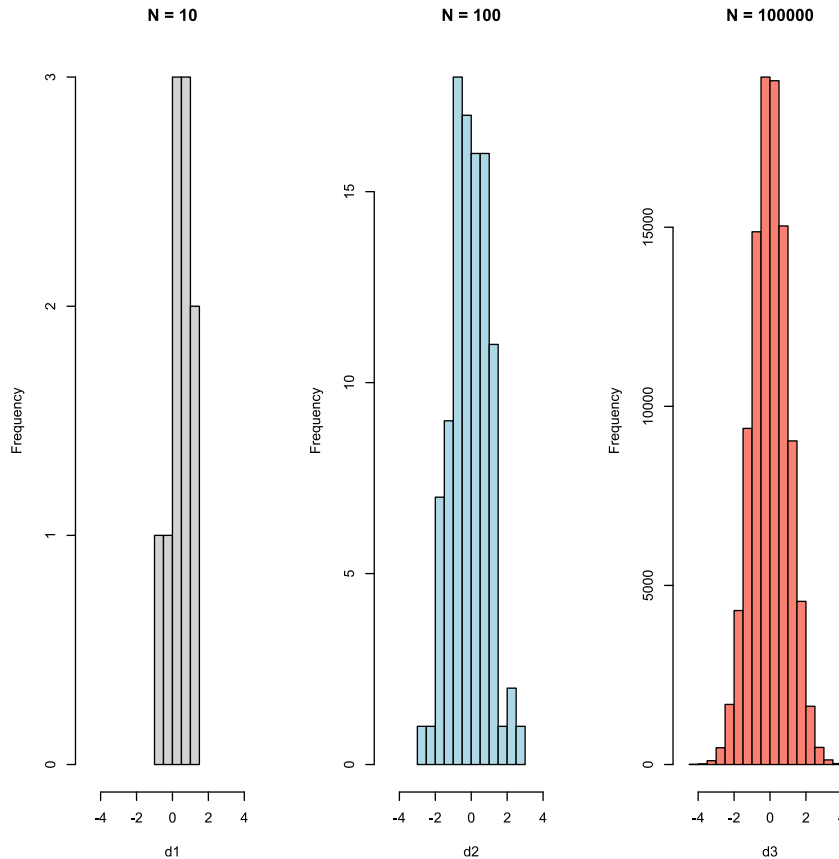


Histogram of d3



# Distribuzione Normale - `rnorm`

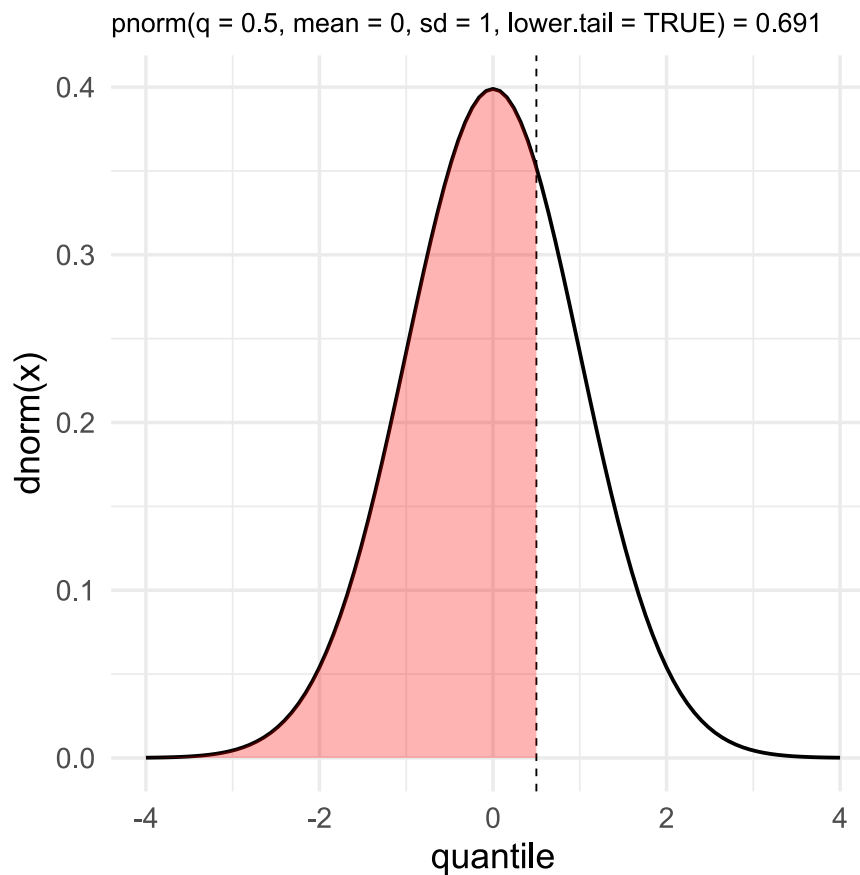
Il parametro critico di `rnorm` è la numerosità. Più osservazioni generiamo più la nostra distribuzione è precisa.



# Distribuzione Normale - `pnorm`

Usiamo una la funzione `ggnorm` per rappresentare `pnorm()`

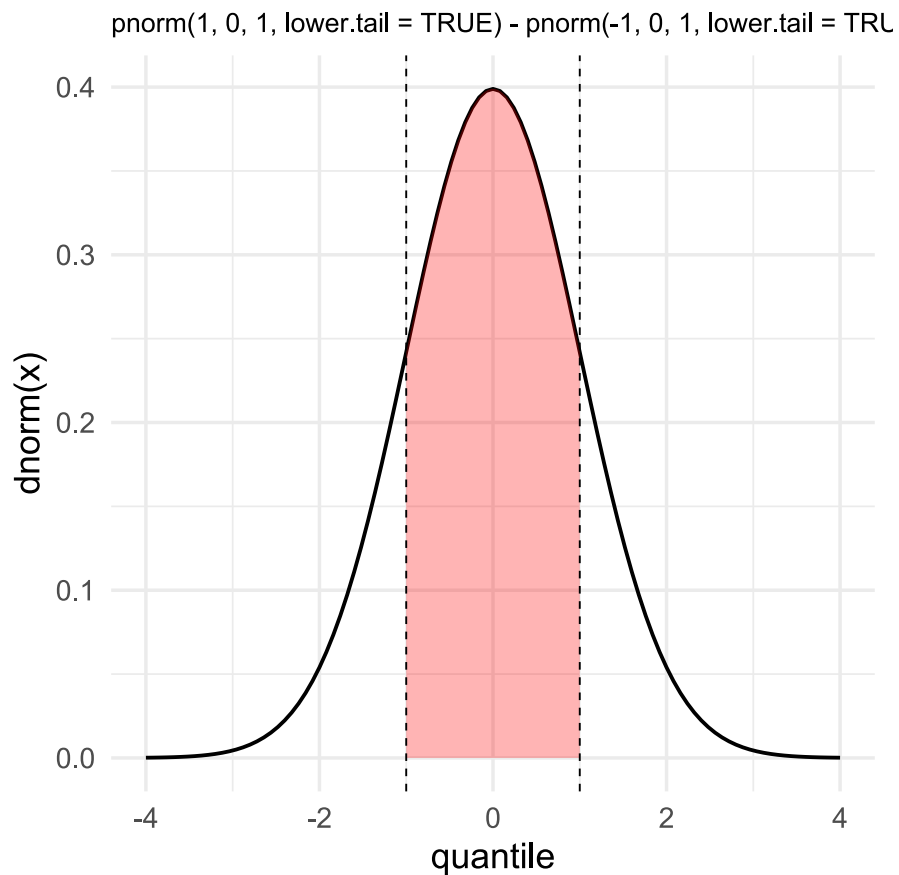
```
ggnorm(q = 0.5, mean = 0, sd = 1, lower.tail = TRUE) # pnorm(0.5, 0, 1, TRUE)
```



# Distribuzione Normale - `pnorm`

Possiamo anche avere un'area delimitata da 2 quantili

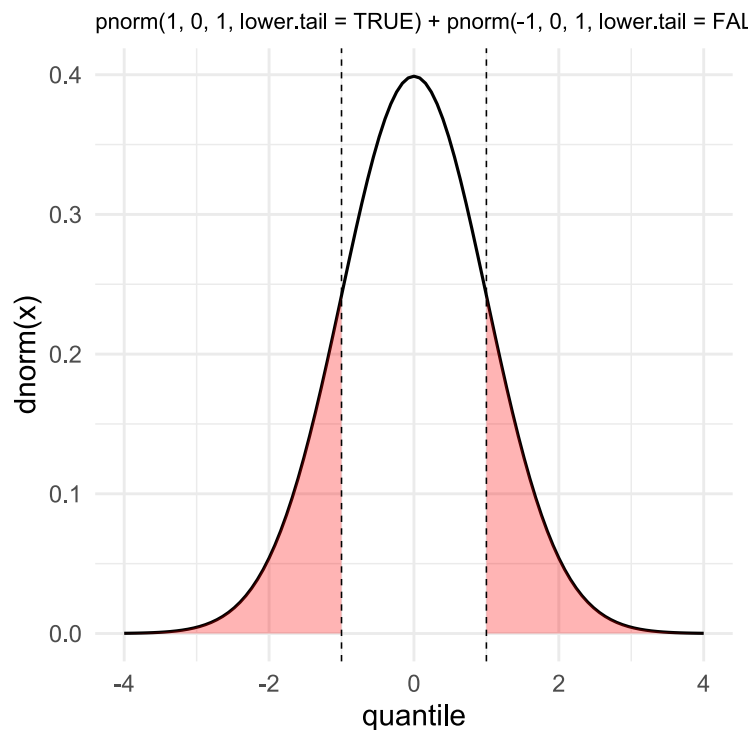
```
ggnorm(q = c(-1, 1), mean = 0, sd = 1) # pnorm(1, 0, 1) - pnorm(-1, 0, 1)
```



# Distribuzione Normale - `pnorm`

Possiamo anche rappresentare il complementare

```
ggnorm(q = c(-1, 1), mean = 0, sd = 1, lower.tail = FALSE) # pnorm(-1, 0, 1, lower.tail = TRUE) + pnorm(1, 0, 1, lower.tail = FALSE)
```



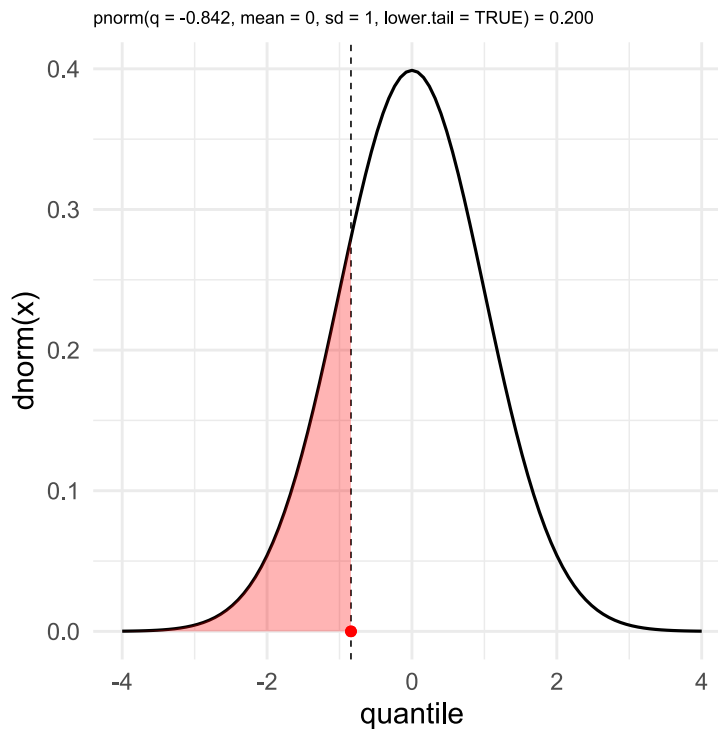


# Distribuzione Normale - `qnorm`

`qnorm` è l'operazione complementare di `pnorm`. Se vogliamo sapere il quantile associato ad una certa probabilità cumulata  $p$  facciamo `qnorm(p, mean, sd)`

```
cumprob <- 0.2 # da -Inf a q
q <- round(qnorm(cumprob, 0, 1), 3)

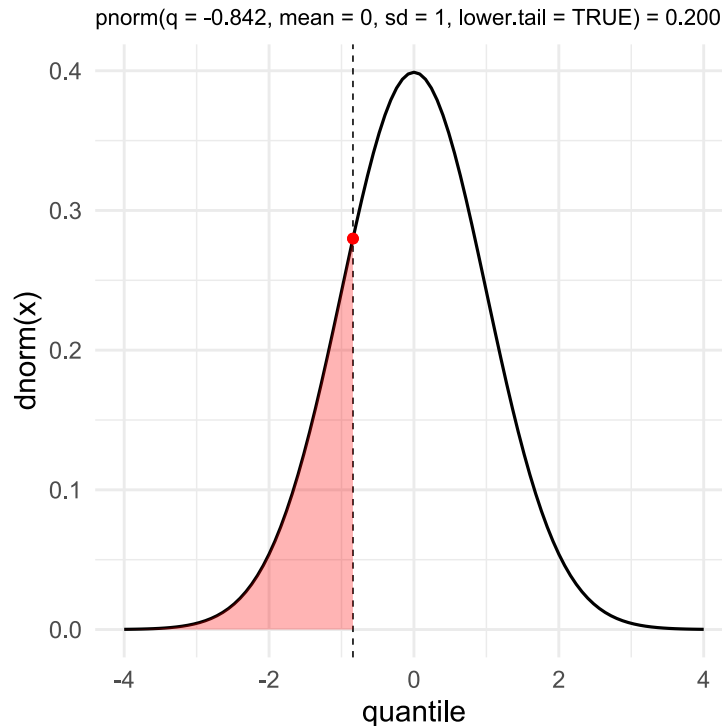
ggnorm(q = q, mean = 0, sd = 1) +
  geom_point(x = q,
            y = 0,
            size = 3,
            color = "red") +
  theme(plot.title = element_text(size = 12))
```



# Distribuzione Normale - `dnorm`

Con `dnorm` calcoliamo la densità di probabilità di un quantile. In termini pratici corrisponde all'asse  $y$  dei nostri grafici:

```
ggnorm(q = q, mean = 0, sd = 1) +  
  geom_point(x = q,  
    y = dnorm(q, 0, 1),  
    size = 3,  
    col = "red")
```



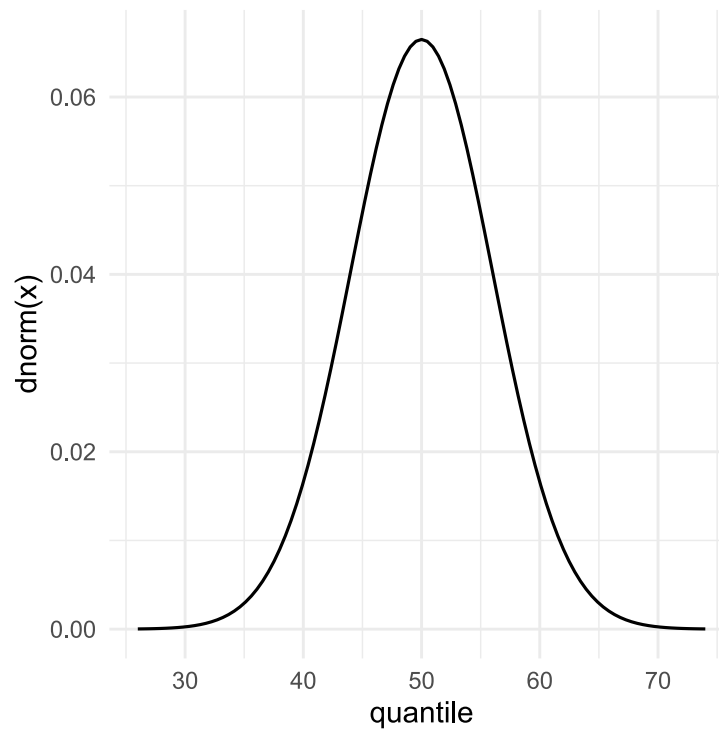
# Esercizi

# Esercizi

Supponiamo che il punteggio  $X$  di un test sull'ansia segua una distribuzione normale con:

- media  $\mu = 50$
- deviazione standard  $\sigma = 6$

```
ggnorm(mean = 50, sd = 6)
```



# Esercizi

Calcolare la probabilità che un soggetto scelto a caso abbia un punteggio:

1. inferiore a 45
2. superiore a 60
3. compreso tra 55 e 65
4. inferiore alla media meno una deviazione standard
5. superiore alla media più una deviazione standard
6. superiore alla media più 4 deviazioni standard

# Soluzioni

```
mu <- 50  
sigma <- 6
```

```
pnorm(45, mu, sigma)
```

```
## [1] 0.2023284
```

```
1 - pnorm(45, mu, sigma) # pnorm(45, mu, sigma, lower.tail = FALSE)
```

```
## [1] 0.7976716
```

```
pnorm(65, mean=mu, sd=sigma) - pnorm(55, mean=mu, sd=sigma)
```

```
## [1] 0.1961187
```

```
pnorm(mu - sigma, mean=mu, sd=sigma)
```

```
## [1] 0.1586553
```

```
1 - pnorm(mu + sigma, mean=mu, sd=sigma)
```

```
## [1] 0.1586553
```

# Esercizi

Calcolare la probabilità che un soggetto scelto a caso abbia un punteggio che non si discosta dalla media più di

1. una deviazione standard
2. 2 deviazioni standard
3. 3 deviazioni standard

Calcolare

1. il 95-esimo percentile della distribuzione dei punteggi
2. il rango percentile associato a un soggetto che ha ottenuto un punteggio di 43

# Soluzioni

```
a <- mu - sigma  
b <- mu + sigma  
pnorm(b, mean=mu, sd=sigma) - pnorm(a, mean=mu, sd=sigma)
```

```
## [1] 0.6826895
```

```
a <- mu - 2*sigma  
b <- mu + 2*sigma  
pnorm(b, mean=mu, sd=sigma) - pnorm(a, mean=mu, sd=sigma)
```

```
## [1] 0.9544997
```

```
a <- mu - 3*sigma  
b <- mu + 3*sigma  
pnorm(b, mean=mu, sd=sigma) - pnorm(a, mean=mu, sd=sigma)
```

```
## [1] 0.9973002
```



# Soluzioni

```
x <- qnorm(0.95, mean=mu, sd=sigma)
x
```

```
## [1] 59.86912
```

```
pnorm(x, mean=mu, sd=sigma)
```

```
## [1] 0.95
```

```
p <- pnorm(43, mean=mu, sd=sigma)
round(p * 100,1)
```

```
## [1] 12.2
```

```
qnorm(p, mean=mu, sd=sigma)
```

```
## [1] 43
```

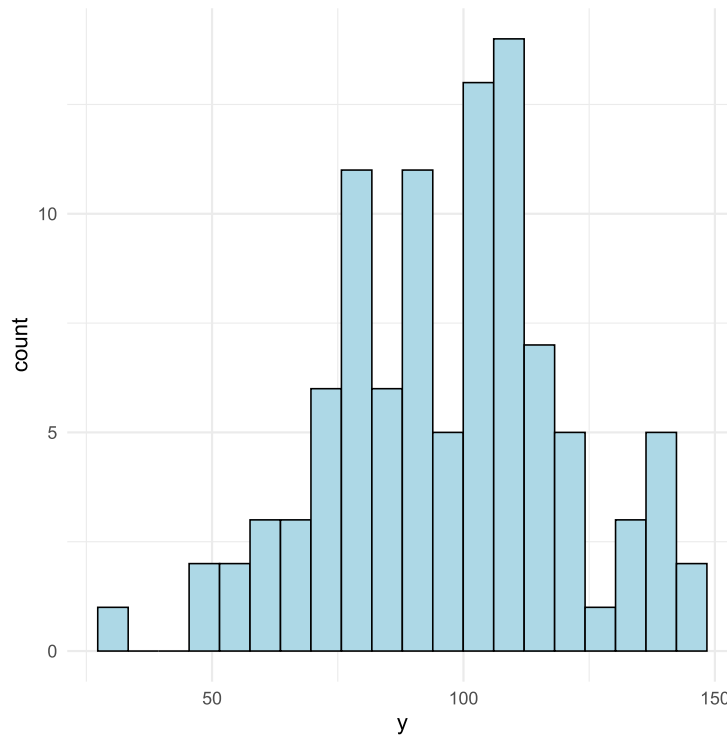
Punti Z

# Punti Z

I punti Z si calcolano  $z_i = \frac{x_i - \mu}{\sigma}$ . Vediamo passo per passo: generiamo dei dati da una distribuzione normale con media  $\mu = 100$  e deviazione standard  $\sigma = 20$ :

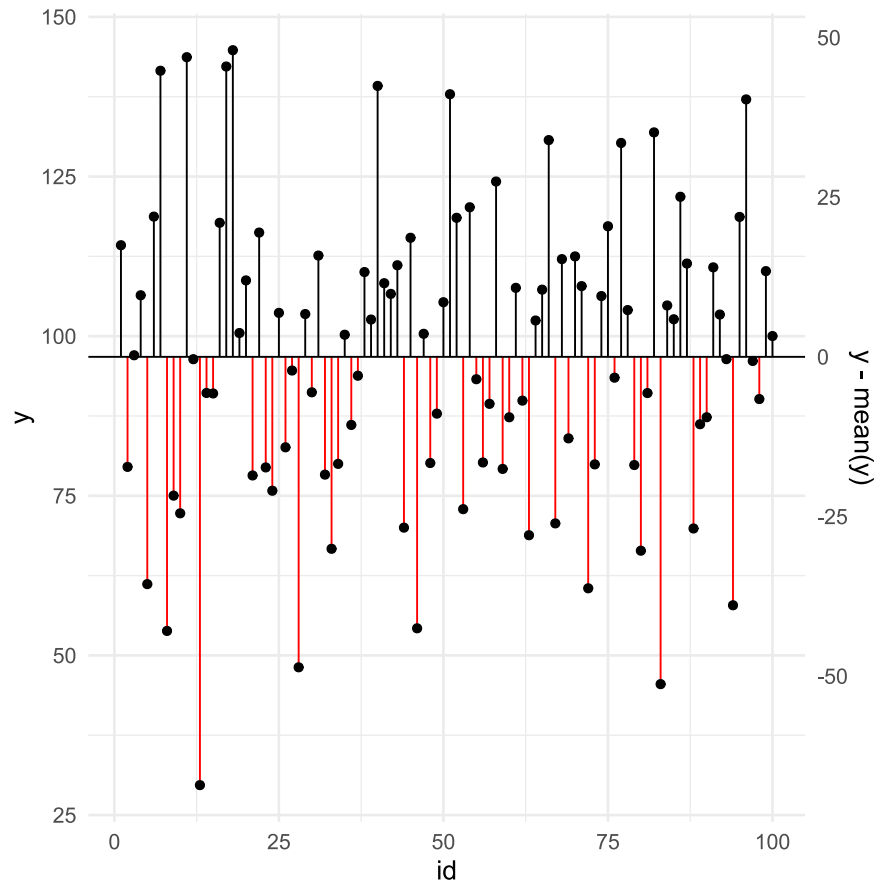
```
dat <- data.frame(y = rnorm(100, 100, 20))
```

```
ggplot(dat, aes(x = y)) +  
  geom_histogram(bins = 20, color = "black", fill = "lightblue") +  
  theme_minimal(base_size = 15)
```



# Punti Z

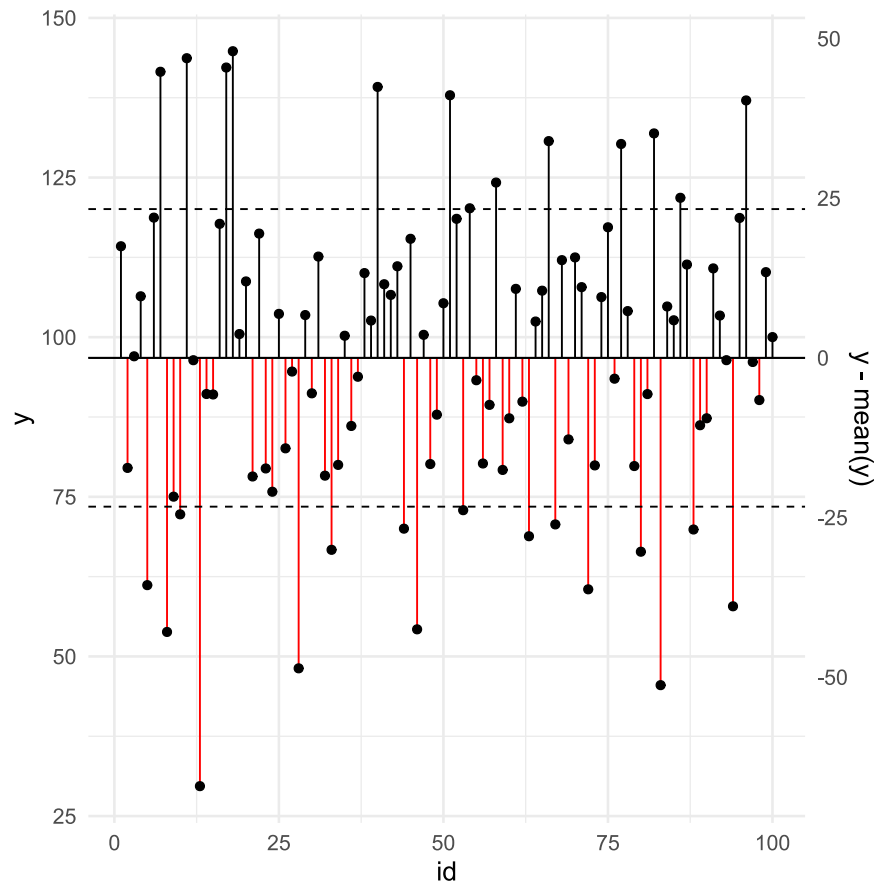
Prima sottraiamo la media da ogni valore  $x_{cen} = x_i - \mu$ . Se un valore è vicino alla media avrà un punteggio vicino a 0:



# Punti Z

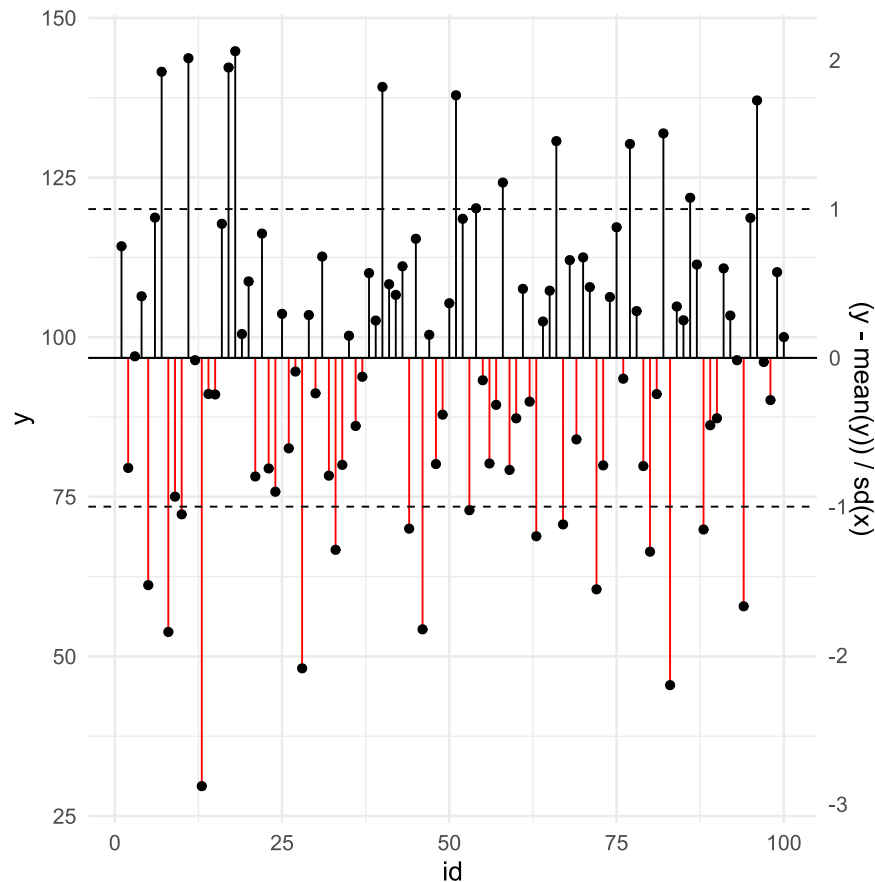
Poi calcoliamo e rappresentiamo la deviazione standard che è

$$\sigma = \sqrt{\frac{\sum_{i=1}^N (x_i - \mu_x)^2}{N}} \text{ (la radice quadrata della media degli scarti al quadrato):}$$



# Punti Z

Infine dividiamo ogni scarto dalla media per la deviazione standard  
 $z_i = \frac{x_i - \mu}{\sigma}$ . In questo modo uno scarto che è vicino alla deviazione standard  
prenderà il valore  $\sim 1$  e così rapportato a tutte le distanze:



# Punti Z

Trasformare dei dati grezzi in punti z è un modo molto utile per interpretarli:

```
mu <- 10
sigma <- 4
n <- 10

datz <- data.frame(
  id = 1:n,
  y = rnorm(n, mu, sigma)
)

datz
```

```
##      id      y
## 1     1 13.913118
## 2     2 10.250330
## 3     3 10.872929
## 4     4 14.193387
## 5     5  5.217891
## 6     6  2.959409
## 7     7 12.005541
## 8     8  6.508148
## 9     9  7.308680
## 10    10 10.859862
```

# Punti Z

Calcoliamo i punti z rispetto ai valori normativi con  $z_i = \frac{x_i - \mu}{\sigma}$ :

```
datz$yz <- (datz$y - mu)/sigma  
datz$yz
```

```
## [1] 0.97827951 0.06258247 0.21823213 1.04834672 -1.19552734 -1.76014764  
## [7] 0.50138527 -0.87296289 -0.67283009 0.21496545
```

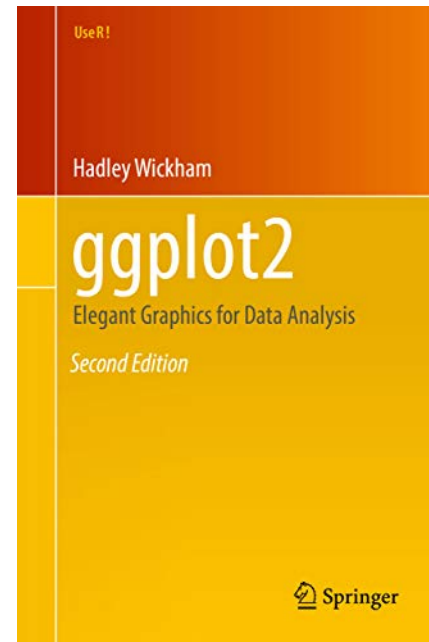
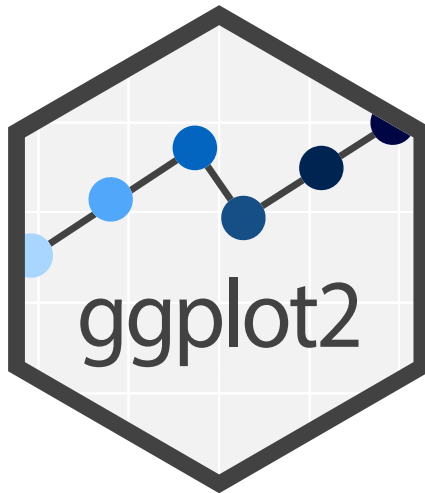
Come li interpretiamo?



ggplot2

# ggplot2

`ggplot2` è un pacchetto per creare grafici alternativo ai grafici di base estremamente potente (e divertente 😊)

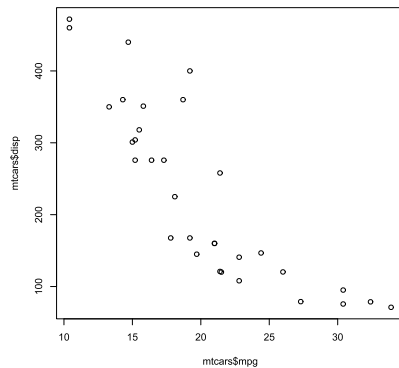


# ggplot2

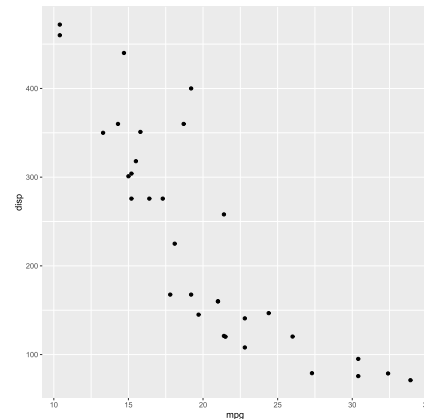
```
# install.packages("ggplot2")  
library(ggplot2)
```

La logica è leggermente diversa rispetto ai grafici di base. Con `ggplot` si compongono i grafici a *strati* concatenando con `+`

```
plot(mtcars$mpg, mtcars$disp)
```



```
ggplot(mtcars, aes(x = mpg, y = disp)) +  
  geom_point()
```



# ggplot2

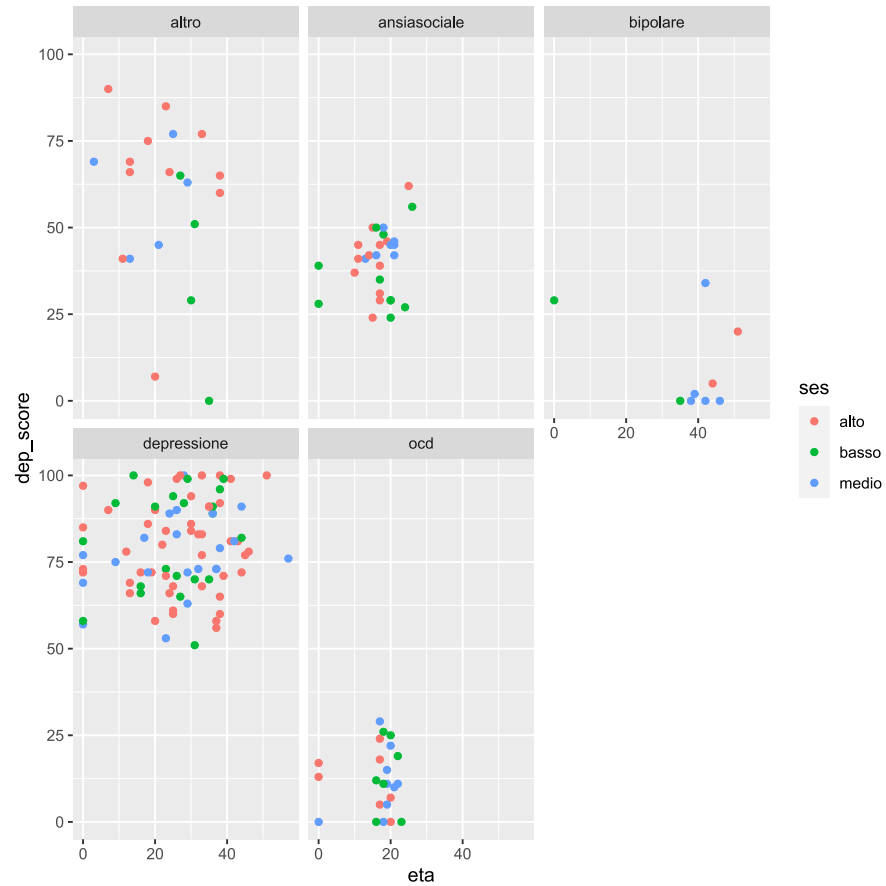
Ma la potenza di `ggplot` si vede nella semplicità nel creare grafici complessi. Prendiamo i dati della lezione precedente:

```
dat <- read.csv("../data/psych.csv", header = TRUE, sep = ";") # importare  
dat <- dat[complete.cases(dat), ]
```

Facciamo uno `scatterplot` della relazione tra `eta` e `depressione` per ogni `diagnosi` colorando in base al `ses`:

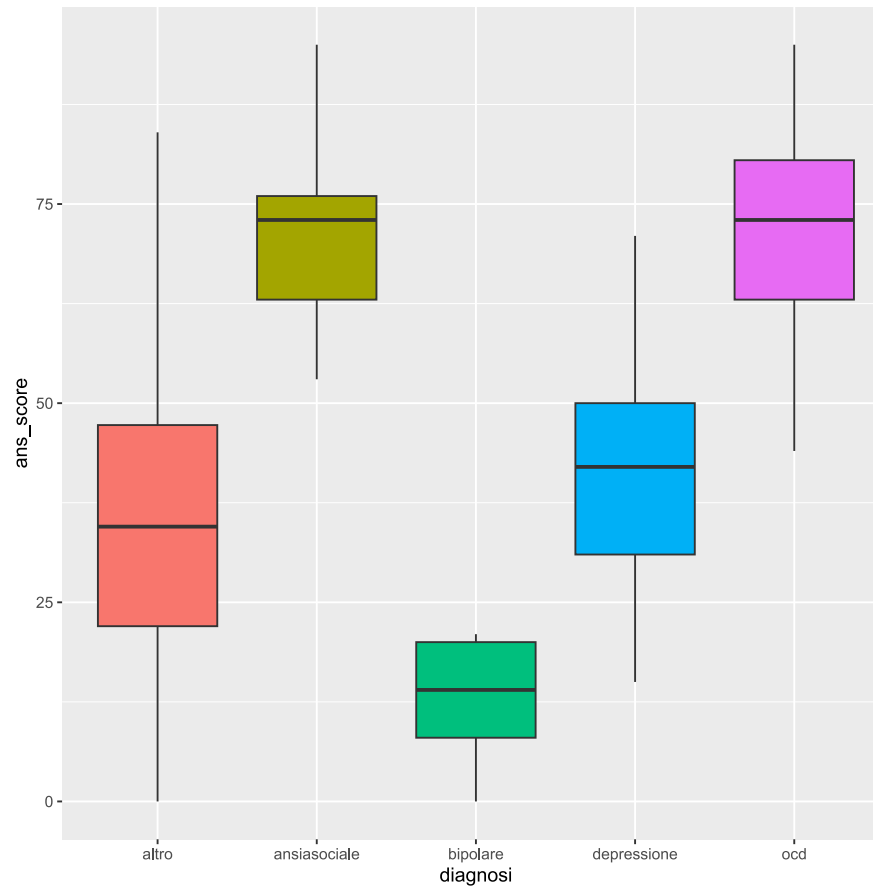
```
ggplot(dat, aes(x = eta, y = dep_score, color = ses)) +  
  geom_point() +  
  facet_wrap(~diagnosi)
```

# ggplot2



# ggplot2

```
ggplot(dat, aes(x = diagnosi, y = ans_score, fill = diagnosi)) +  
  geom_boxplot(show.legend = FALSE)
```



... ma molto altro! 😄

- <https://r-graph-gallery.com/ggplot2-package.html>
- <https://ggplot2-book.org/>
- <http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html>