

# Laboratorio di Basi di Dati

R e le basi di dati

Dario Della Monica

12 gennaio 2023

# Outline

Connessione ad una base di dati e esecuzione di query

Utilizzo di R per popolare la base di dati con dati casuali

Analisi statistica dei dati con grafici

# Outline

Connessione ad una base di dati e esecuzione di query

Utilizzo di R per popolare la base di dati con dati casuali

Analisi statistica dei dati con grafici

# Connessione ad una base di dati

- ▶ installare "RPostgreSQL"
  - ▶ `install.packages("RPostgreSQL")`
    - ▶ potrebbe essere necessario installare dei pacchetti per le dipendenze a livello di sistema operativo (es.: `r-cran-dbi`, `r-cran-rpostgresql`)
    - ▶ si può provare la libreria "RPostgres" in alternativa in caso di problemi. Sembra funzionare meglio con alcune versioni recenti di PostgreSQL e le differenze nell'utilizzo sono minime (non testata)
  - ▶ `installed.packages()`: pacchetti installati (matrice)
  - ▶ `remove.packages()` disinstalla un pacchetto
- ▶ caricare la libreria "RPostgreSQL"
  - ▶ `library("RPostgreSQL")`
  - ▶ `.packages()`: vettore dei pacchetti caricati
    - ▶ usare tra parentesi: `(.packages())`  
oppure in un assegnamento: `d <- .packages()`
  - ▶ `detach("package:ggplot2")`: rimuove il pacchetto `ggplot2` dalla lista di pacchetti caricati (*unload* del pacchetto)

## Connessione ad una base di dati (2)

- ▶ caricare il driver PostgreSQL (e memorizzarlo in una variabile)

- ▶ `drv <- dbDriver("PostgreSQL")`

- ▶ creare la connessione (e memorizzarla in una variabile)

- ▶ `con <- dbConnect(drv,  
                  dbname="<nome_database>",  
                  host="<indirizzo_IP_dbserver>",  
                  port=<port_number>, # usually 5432  
                  user="<username>",  
                  password="<password>")`

## Esecuzione di una query

- ▶ con visualizzazione del risultato (dataframe)
  - ▶ `dbGetQuery(con, "<SQLquery>")`
- ▶ con memorizzazione del risultato in un dataframe
  - ▶ `res <- dbGetQuery(con, "<SQLquery>")`
- ▶ per visualizzazioni e manipolazioni successive (il risultato è una variabile di tipo PostgreSQLResult)
  - ▶ `res <- dbSendQuery(con, "<SQLquery>")`
  - ▶ `df <- fetch(res, n = 3)`
  - ▶ `dbHasCompleted(res)` *# FALSE finché la fine della tabella  
# è raggiunta*

# Esecuzione di una query

- ▶ con visualizzazione del risultato (dataframe)
  - ▶ `dbGetQuery(con, "<SQLquery>")`
- ▶ con memorizzazione del risultato in un dataframe
  - ▶ `res <- dbGetQuery(con, "<SQLquery>")`
- ▶ per visualizzazioni e manipolazioni successive  
(il risultato è una variabile di tipo PostgreSQLResult)
  - ▶ `res <- dbSendQuery(con, "<SQLquery>")`
  - ▶ `df <- fetch(res, n = 3)`
  - ▶ `dbHasCompleted(res)` *# FALSE finché la fine della tabella  
# è raggiunta*
  - ▶ `df <- fetch(res, n = 3)`
  - ▶ `dbHasCompleted(res)`
  - ▶ ...
  - ▶ `dbClearResult(res)`

## Esecuzione di una query

- ▶ con visualizzazione del risultato (dataframe)
  - ▶ `dbGetQuery(con, "<SQLquery>")`
- ▶ con memorizzazione del risultato in un dataframe
  - ▶ `res <- dbGetQuery(con, "<SQLquery>")`
- ▶ per visualizzazioni e manipolazioni successive (il risultato è una variabile di tipo PostgreSQLResult)
  - ▶ `res <- dbSendQuery(con, "<SQLquery>")`
  - ▶ `df <- fetch(res, n = 3)`
  - ▶ `dbHasCompleted(res)` *# FALSE finché la fine della tabella  
# è raggiunta*
  - ▶ `df <- fetch(res, n = 3)`
  - ▶ `dbHasCompleted(res)`
  - ▶ ...
  - ▶ `dbClearResult(res)`

```
while (! dbHasCompleted(res))
{
  partial_df <- fetch(res, 2)
  print(partial_df)
}
```

## Disconnessione dalla base di dati

▶ `dbDisconnect(con)`

# Outline

Connessione ad una base di dati e esecuzione di query

Utilizzo di R per popolare la base di dati con dati casuali

Analisi statistica dei dati con grafici

# Funzione dbWriteTable

```
dbWriteTable( con,  
              name=c("<schema>", "<tabella>"),  
              value=<data_frame>,  
              append=<T/F>,  
              row.names=<T/F>)
```

## Popolare la tabella studenti

1. scaricare degli elenchi di nomi e cognomi (facile da google o dalla mia pagina web)
2. leggere i 2 file: memorizzare in un vettore una stringa per ogni riga (funzione `readLines`)

## Popolare la tabella studenti

1. scaricare degli elenchi di nomi e cognomi (facile da google o dalla mia pagina web)
2. leggere i 2 file: memorizzare in un vettore una stringa per ogni riga (funzione `readLines`)
  - ▶ `v_nomi<-readLines("./Desktop/nomi.txt")`
  - ▶ `v_cognomi<-readLines("./Desktop/cognomi.txt")`

## Popolare la tabella studenti

1. scaricare degli elenchi di nomi e cognomi (facile da google o dalla mia pagina web)
2. leggere i 2 file: memorizzare in un vettore una stringa per ogni riga (**funzione readLines**)
  - ▶ `v_nomi<-readLines("./Desktop/nomi.txt")`
  - ▶ `v_cognomi<-readLines("./Desktop/cognomi.txt")`
3. creare un dataframe di 10 000 studenti a partire da `v_nomi` e `v_cognomi`

## Popolare la tabella studenti

1. scaricare degli elenchi di nomi e cognomi (facile da google o dalla mia pagina web)
2. leggere i 2 file: memorizzare in un vettore una stringa per ogni riga (funzione `readLines`)

- ▶ `v_nomi<-readLines("./Desktop/nomi.txt")`

- ▶ `v_cognomi<-readLines("./Desktop/cognomi.txt")`

3. creare un dataframe di 10 000 studenti a partire da `v_nomi` e `v_cognomi`

- ▶ 

```
studenti_df <- data.frame(
  matricola = sample(1:1000000, 10000, replace=F),
  nome=sample(v_nomi, 10000, replace = T),
  cognome=sample(v_cognomi, 10000, replace = T),
  sesso=sample(c("m","f"), 10000, replace=T))
```

## Popolare la tabella studenti

1. scaricare degli elenchi di nomi e cognomi (facile da google o dalla mia pagina web)
2. leggere i 2 file: memorizzare in un vettore una stringa per ogni riga (**funzione readLines**)
  - ▶ `v_nomi<-readLines("./Desktop/nomi.txt")`
  - ▶ `v_cognomi<-readLines("./Desktop/cognomi.txt")`
3. creare un dataframe di 10 000 studenti a partire da `v_nomi` e `v_cognomi`
  - ▶ 

```
studenti_df <- data.frame(
  matricola = sample(1:1000000, 10000, replace=F),
  nome=sample(v_nomi, 10000, replace = T),
  cognome=sample(v_cognomi, 10000, replace = T),
  sesso=sample(c("m","f"), 10000, replace=T))
```
4. inserire le righe del dataframe nella tabella del DB con `dbWriteTable`

## Popolare la tabella studenti

1. scaricare degli elenchi di nomi e cognomi (facile da google o dalla mia pagina web)
2. leggere i 2 file: memorizzare in un vettore una stringa per ogni riga (**funzione readLines**)

- ▶ `v_nomi<-readLines("./Desktop/nomi.txt")`

- ▶ `v_cognomi<-readLines("./Desktop/cognomi.txt")`

3. creare un dataframe di 10 000 studenti a partire da `v_nomi` e `v_cognomi`

- ▶ 

```
studenti_df <- data.frame(
  matricola = sample(1:1000000, 10000, replace=F),
  nome=sample(v_nomi, 10000, replace = T),
  cognome=sample(v_cognomi, 10000, replace = T),
  sesso=sample(c("m","f"), 10000, replace=T))
```

4. inserire le righe del dataframe nella tabella del DB con `dbWriteTable`

- ▶ 

```
dbWriteTable(con, name=c("public","studenti"),
  value=studenti_df, append = T, row.names=F)
```

# Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea

## Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea

(a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`

(b) `temp_cdl <- temp_cdl$nome`

(c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`

## Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`
2. creare il vettore delle matricole presenti in tabella `studenti`

## Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`
2. creare il vettore delle matricole presenti in tabella studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `iscritto_a.stud <- temp_matricola$matricola`

## Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`
2. creare il vettore delle matricole presenti in tabella studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `iscritto_a.stud <- temp_matricola$matricola`
3. creare un vettore di 10 000 interi casuali tra 1978 (fondazione di uniuud) e 2021

# Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`
2. creare il vettore delle matricole presenti in tabella studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `iscritto_a.stud <- temp_matricola$matricola`
3. creare un vettore di 10 000 interi casuali tra 1978 (fondazione di uniud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 10000, replace=T)`

# Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`
2. creare il vettore delle matricole presenti in tabella studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `iscritto_a.stud <- temp_matricola$matricola`
3. creare un vettore di 10 000 interi casuali tra 1978 (fondazione di uniud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 10000, replace=T)`
4. creare dataframe a partire da `iscritto_a.cdl`, `iscritto_a.stud` e `iscritto_a.anno`

# Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`
2. creare il vettore delle matricole presenti in tabella studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `iscritto_a.stud <- temp_matricola$matricola`
3. creare un vettore di 10 000 interi casuali tra 1978 (fondazione di uniuud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 10000, replace=T)`
4. creare dataframe a partire da `iscritto_a.cdl`, `iscritto_a.stud` e `iscritto_a.anno`
  - (a) `iscritto_a_df <- data.frame(cdl=iscritto_a.cdl,  
stud=iscritto_a.stud,  
anno=iscritto_a.anno)`

## Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`
2. creare il vettore delle matricole presenti in tabella studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `iscritto_a.stud <- temp_matricola$matricola`
3. creare un vettore di 10 000 interi casuali tra 1978 (fondazione di uniuud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 10000, replace=T)`
4. creare dataframe a partire da `iscritto_a.cdl`, `iscritto_a.stud` e `iscritto_a.anno`
  - (a) `iscritto_a_df <- data.frame(cdl=iscritto_a.cdl,  
stud=iscritto_a.stud,  
anno=iscritto_a.anno)`
5. inserire le righe del dataframe nella tabella del DB con `dbWriteTable`

## Popolare la relazione `iscritto_a`

1. creare un vettore di 10 000 (numero studenti) elementi presi a caso dall'elenco dei corsi di laurea
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 10000, replace=T)`
2. creare il vettore delle matricole presenti in tabella studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `iscritto_a.stud <- temp_matricola$matricola`
3. creare un vettore di 10 000 interi casuali tra 1978 (fondazione di uniud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 10000, replace=T)`
4. creare dataframe a partire da `iscritto_a.cdl`, `iscritto_a.stud` e `iscritto_a.anno`
  - (a) `iscritto_a_df <- data.frame(cdl=iscritto_a.cdl,  
stud=iscritto_a.stud,  
anno=iscritto_a.anno)`
5. inserire le righe del dataframe nella tabella del DB con `dbWriteTable`
  - (a) `dbWriteTable(con, name=c("public","iscritto_a"),  
value=iscritto_a_df, row.names=F, append = T)`

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 1000 (10% di 10000) corsi di laurea a caso

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 1000 (10% di 10000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 1000 (10% di 10000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`
7. creare un vettore contenente il 10% (1000) delle matricole studenti

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 1000 (10% di 10000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`
7. creare un vettore contenente il 10% (1000) delle matricole studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `temp_matricola <- temp_matricola$matricola`
  - (c) `iscritto_a.stud <- sample(temp_matricola, 1000, replace=F)`

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 1000 (10% di 10000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`
7. creare un vettore contenente il 10% (1000) delle matricole studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `temp_matricola <- temp_matricola$matricola`
  - (c) `iscritto_a.stud <- sample(temp_matricola, 1000, replace=F)`
8. creare un vettore di 1000 interi casuali tra 1978 (fondazione di uniuud) e 2021

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 1000 (10% di 10000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`
7. creare un vettore contenente il 10% (1000) delle matricole studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `temp_matricola <- temp_matricola$matricola`
  - (c) `iscritto_a.stud <- sample(temp_matricola, 1000, replace=F)`
8. creare un vettore di 1000 interi casuali tra 1978 (fondazione di uniud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 1000, replace=T)`

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 1000 (10% di 10000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`
7. creare un vettore contenente il 10% (1000) delle matricole studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `temp_matricola <- temp_matricola$matricola`
  - (c) `iscritto_a.stud <- sample(temp_matricola, 1000, replace=F)`
8. creare un vettore di 1000 interi casuali tra 1978 (fondazione di uniud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 1000, replace=T)`
9. creare dataframe a partire da `iscritto_a.cdl`, `iscritto_a.stud` e `iscritto_a.anno`

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 10% (1000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`
7. creare un vettore contenente il 10% (1000) delle matricole studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `temp_matricola <- temp_matricola$matricola`
  - (c) `iscritto_a.stud <- sample(temp_matricola, 1000, replace=F)`
8. creare un vettore di 1000 interi casuali tra 1978 (fondazione di uniud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 1000, replace=T)`
9. creare dataframe a partire da `iscritto_a.cdl`, `iscritto_a.stud` e `iscritto_a.anno`
  - (a) `iscritto_a_df <- data.frame(cdl=iscritto_a.cdl,  
stud=iscritto_a.stud,  
anno=iscritto_a.anno)`

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 10% (1000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`
7. creare un vettore contenente il 10% (1000) delle matricole studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `temp_matricola <- temp_matricola$matricola`
  - (c) `iscritto_a.stud <- sample(temp_matricola, 1000, replace=F)`
8. creare un vettore di 1000 interi casuali tra 1978 (fondazione di uniud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 1000, replace=T)`
9. creare dataframe a partire da `iscritto_a.cdl`, `iscritto_a.stud` e `iscritto_a.anno`
  - (a) `iscritto_a_df <- data.frame(cdl=iscritto_a.cdl,  
stud=iscritto_a.stud,  
anno=iscritto_a.anno)`
10. inserire le righe del dataframe nella tabella del DB con `dbWriteTable`

## Popolare la relazione `iscritto_a` (2)

Assumiamo che il 10% degli iscritti si è iscritto a 2 corsi di laurea

6. creare un vettore contenente il 10% (1000) corsi di laurea a caso
  - (a) `temp_cdl <- dbGetQuery(con, "select nome from corsi_di_laurea")`
  - (b) `temp_cdl <- temp_cdl$nome`
  - (c) `iscritto_a.cdl <- sample(temp_cdl, 1000, replace=T)`
7. creare un vettore contenente il 10% (1000) delle matricole studenti
  - (a) `temp_matricola <- dbGetQuery(con, "select matricola from studenti")`
  - (b) `temp_matricola <- temp_matricola$matricola`
  - (c) `iscritto_a.stud <- sample(temp_matricola, 1000, replace=F)`
8. creare un vettore di 1000 interi casuali tra 1978 (fondazione di uniud) e 2021
  - (a) `iscritto_a.anno <- sample(1978:2021, 1000, replace=T)`
9. creare dataframe a partire da `iscritto_a.cdl`, `iscritto_a.stud` e `iscritto_a.anno`
  - (a) `iscritto_a_df <- data.frame(cdl=iscritto_a.cdl,  
stud=iscritto_a.stud,  
anno=iscritto_a.anno)`
10. inserire le righe del dataframe nella tabella del DB con `dbWriteTable`
  - (a) `dbWriteTable(con, name=c("public","iscritto_a"),  
value=iscritto_a_df, append=T, row.names=F)`

# Outline

Connessione ad una base di dati e esecuzione di query

Utilizzo di R per popolare la base di dati con dati casuali

Analisi statistica dei dati con grafici

Visualizzare numero iscritti per anno

# Visualizzare numero iscritti per anno

1. ottenere i dati dalla base di dati

# Visualizzare numero iscritti per anno

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
                    "SELECT anno, count(*)  
                      FROM iscritto_a  
                      GROUP BY anno  
                      ORDER BY anno")
```

# Visualizzare numero iscritti per anno

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
                    "SELECT anno, count(*)  
                      FROM iscritto_a  
                      GROUP BY anno  
                      ORDER BY anno")
```

## 2. visualizzare il grafico con plot

# Visualizzare numero iscritti per anno

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
                    "SELECT anno, count(*)  
                      FROM iscritto_a  
                      GROUP BY anno  
                      ORDER BY anno")
```

## 2. visualizzare il grafico con plot

```
▶ plot(df$anno, df$count, "o")
```

Visualizzare numero iscritti per anno e corso di laurea

# Visualizzare numero iscritti per anno e corso di laurea

1. ottenere i dati dalla base di dati

# Visualizzare numero iscritti per anno e corso di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT anno, cdl, count(*)  
  FROM iscritto_a  
  GROUP BY anno, cdl  
  ORDER BY anno, cdl")
```

# Visualizzare numero iscritti per anno e corso di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT anno, cd1, count(*)  
  FROM iscritto_a  
  GROUP BY anno, cd1  
  ORDER BY anno, cd1")
```

## 2. manipolare i dati per ottenere strutture adatte all'utilizzo con [matplotlib](#): un vettore con gli anni e una matrice con una colonna per ogni corso di laurea ed una riga per ogni anno

# Visualizzare numero iscritti per anno e corso di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT anno, cdl, count(*)  
  FROM iscritto_a  
  GROUP BY anno, cdl  
  ORDER BY anno, cdl")
```

## 2. manipolare i dati per ottenere strutture adatte all'utilizzo con `matplot`: un vettore con gli anni e una matrice con una colonna per ogni corso di laurea ed una riga per ogni anno

```
▶ colonna_anno <- unique(df$anno)  
▶ matr <- matrix(df$count, nrow = length(unique(df$cdl)))  
▶ rownames(matr) <- unique(df$cdl)  
▶ matr <- t(matr)
```

# Visualizzare numero iscritti per anno e corso di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT anno, cdl, count(*)  
  FROM iscritto_a  
  GROUP BY anno, cdl  
  ORDER BY anno, cdl")
```

## 2. manipolare i dati per ottenere strutture adatte all'utilizzo con `matplot`: un vettore con gli anni e una matrice con una colonna per ogni corso di laurea ed una riga per ogni anno

```
▶ colonna_anno <- unique(df$anno)  
▶ matr <- matrix(df$count, nrow = length(unique(df$cdl)))  
▶ rownames(matr) <- unique(df$cdl)  
▶ matr <- t(matr)
```

## 3. visualizzare il grafico con `matplot`

# Visualizzare numero iscritti per anno e corso di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT anno, cdl, count(*)  
  FROM iscritto_a  
  GROUP BY anno, cdl  
  ORDER BY anno, cdl")
```

## 2. manipolare i dati per ottenere strutture adatte all'utilizzo con `matplot`: un vettore con gli anni e una matrice con una colonna per ogni corso di laurea ed una riga per ogni anno

```
▶ colonna_anno <- unique(df$anno)  
▶ matr <- matrix(df$count, nrow = length(unique(df$cdl)))  
▶ rownames(matr) <- unique(df$cdl)  
▶ matr <- t(matr)
```

## 3. visualizzare il grafico con `matplot`

```
▶ matplot(colonna_anno, matr, type="l")
```

Visualizzare grafico a barre degli studenti divisi per sesso e corsi di laurea

# Visualizzare grafico a barre degli studenti divisi per sesso e corsi di laurea

1. ottenere i dati dalla base di dati

# Visualizzare grafico a barre degli studenti divisi per sesso e corsi di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT cdl, sesso, count(*)  
  FROM studenti JOIN iscritto_a ON matricola = stud  
  GROUP BY cdl, sesso  
  ORDER BY cdl, sesso")
```

# Visualizzare grafico a barre degli studenti divisi per sesso e corsi di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT cdl, sesso, count(*)  
  FROM studenti JOIN iscritto_a ON matricola = stud  
  GROUP BY cdl, sesso  
  ORDER BY cdl, sesso")
```

## 2. ri-arrangiare i dati in un formato *barplot-friendly*

# Visualizzare grafico a barre degli studenti divisi per sesso e corsi di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT cdl, sesso, count(*)  
  FROM studenti JOIN iscritto_a ON matricola = stud  
  GROUP BY cdl, sesso  
  ORDER BY cdl, sesso")
```

## 2. ri-arrangiare i dati in un formato *barplot-friendly*

```
▶ matr <- matrix(df$count, nrow = 2)  
▶ rownames(matr) <- c("f","m")  
▶ colnames(matr) <- unique(df$cdl)
```

# Visualizzare grafico a barre degli studenti divisi per sesso e corsi di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT cdl, sesso, count(*)  
  FROM studenti JOIN iscritto_a ON matricola = stud  
  GROUP BY cdl, sesso  
  ORDER BY cdl, sesso")
```

## 2. ri-arrangiare i dati in un formato *barplot-friendly*

```
▶ matr <- matrix(df$count, nrow = 2)  
▶ rownames(matr) <- c("f","m")  
▶ colnames(matr) <- unique(df$cdl)
```

## 3. visualizzare il grafico con barplot

# Visualizzare grafico a barre degli studenti divisi per sesso e corsi di laurea

## 1. ottenere i dati dalla base di dati

```
▶ df <- dbGetQuery(con,  
  "SELECT cdl, sesso, count(*)  
  FROM studenti JOIN iscritto_a ON matricola = stud  
  GROUP BY cdl, sesso  
  ORDER BY cdl, sesso")
```

## 2. ri-arrangiare i dati in un formato *barplot-friendly*

```
▶ matr <- matrix(df$count, nrow = 2)  
▶ rownames(matr) <- c("f","m")  
▶ colnames(matr) <- unique(df$cdl)
```

## 3. visualizzare il grafico con barplot

```
▶ barplot(matr)  
▶ barplot(matr, beside=T)
```