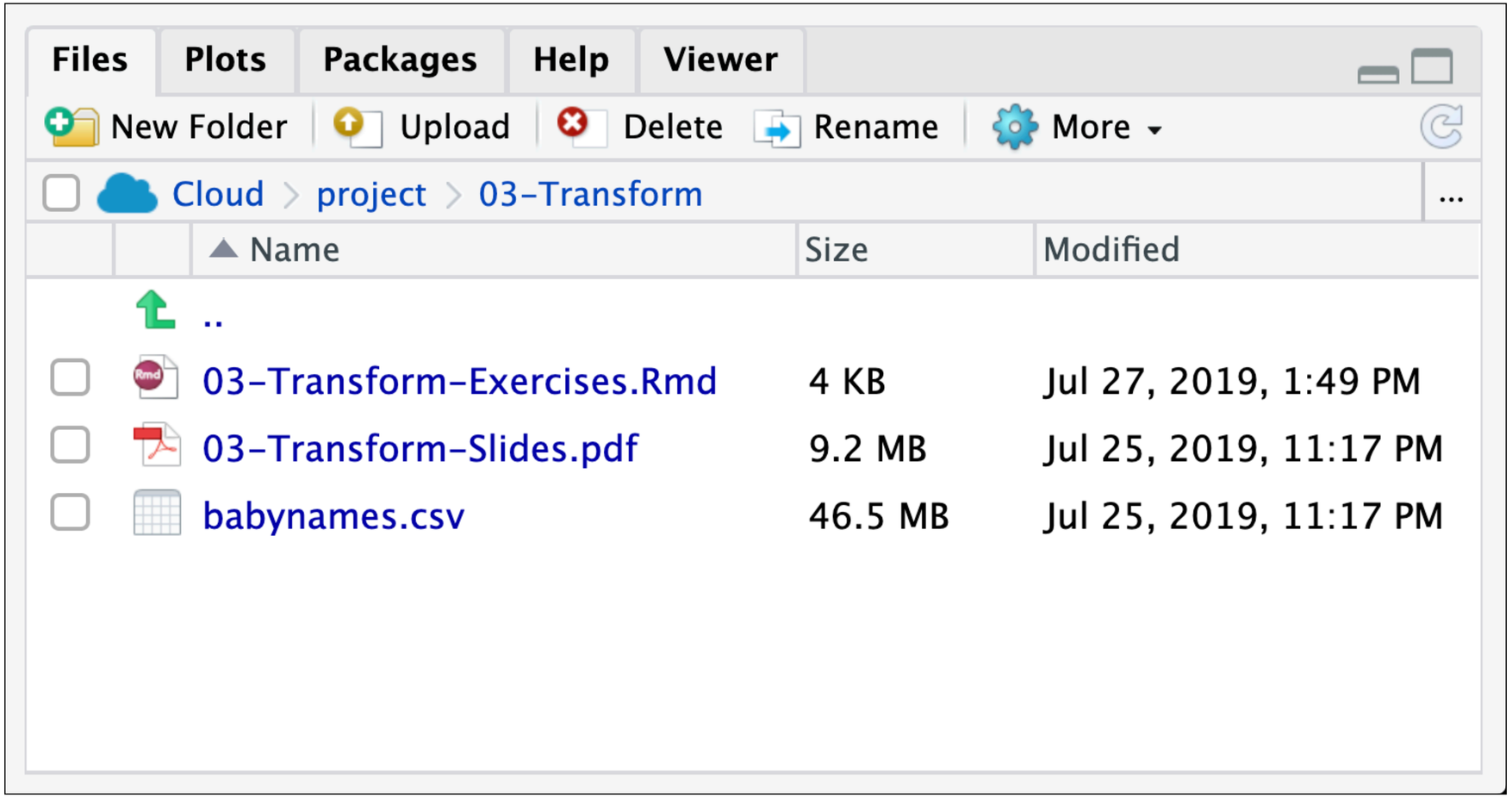
Transform Data with

FUTURE EDUCATION MODENA

babynames.csv

Names and sex of babies born in the US from 1880 to 2017. 1.9M rows.



dplyr

FUTURE EDUCATION MODENA

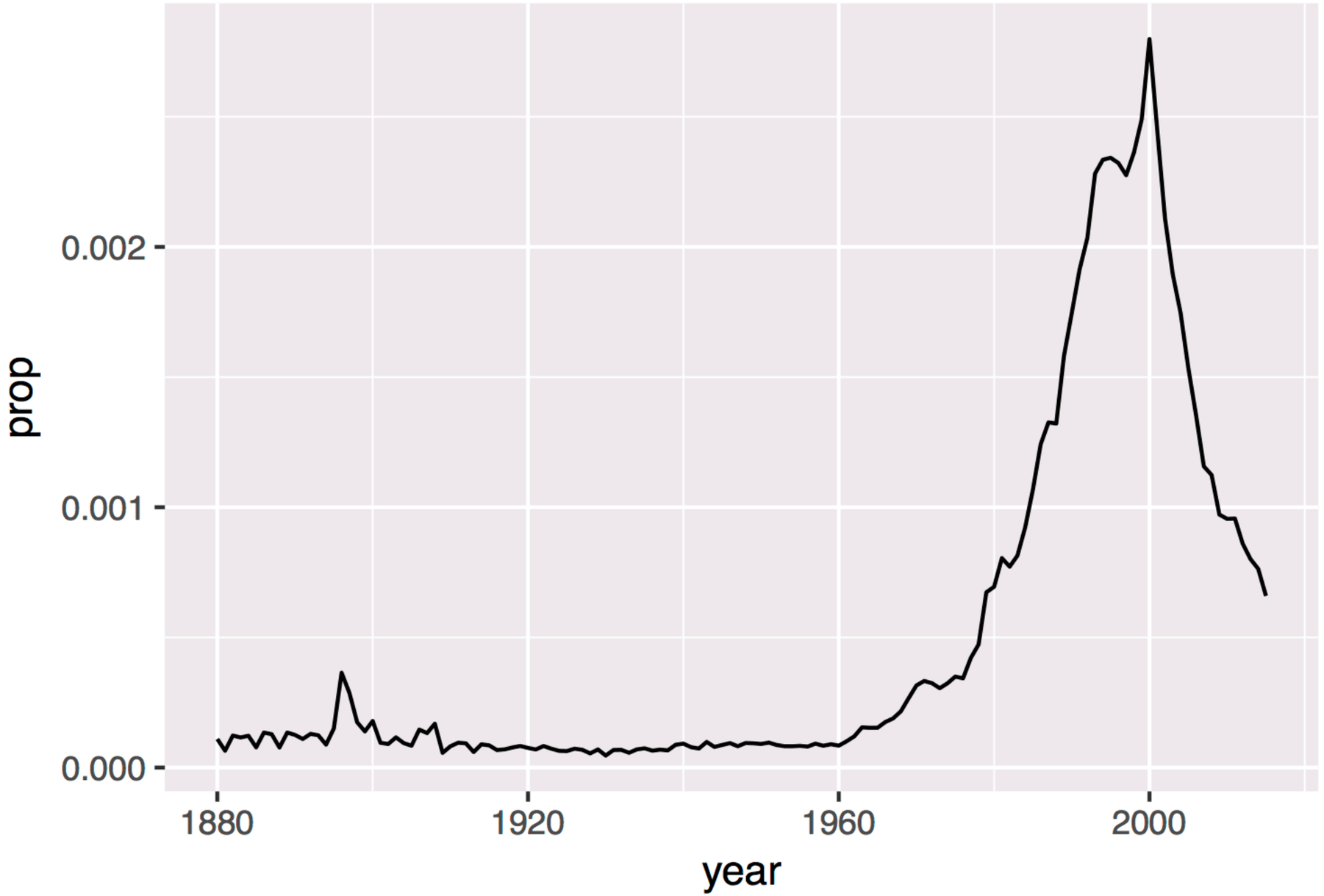
babynames

year <dbl>	sex <chr>	name <chr>	n <dbl>	prop <dbl>
1880	F	Mary	7065	0.07238359
1880	F	Anna	2604	0.02667896
1880	F	Emma	2003	0.02052149
1880	F	Elizabeth	1939	0.01986579
1880	F	Minnie	1746	0.01788843
1880	F	Margaret	1578	0.01616720
1880	F	Ida	1472	0.01508119
1880	F	Alice	1414	0.01448696
1880	F	Bertha	1320	0.01352390
1880	F	Sarah	1288	0.01319605

1–10 of 1,924,665 rows Previous 1 2 3 4 5 6 … 100 Next

FUTURE EDUCATION MODENA

Proportion of boys with the name Garrett



dplyr

Import data

Working directory

R si associa ad una cartella (cioè directory) sul vostro computer. Per vedere quale, eseguite **getwd()** nella console.

- Questa cartella è conosciuta come la vostra “working directory".
- Quando salvate dei file, R li salverà qui
- Quando caricate i file, R li cercherà qui

Crea una cartella per R-studio

1. Crea una cartella dove aggiungere tutti gli script e dataset del corso (tipo desktop>D4SI)
2. Aggiungi il cvs e lo script di oggi
3. Su R studio, nella barra in alto clicca su Session> set working directory > choose working directory > seleziona la cartella che hai creato nel punto 1
4. Importa il CVS su R studio, cliccando in alto a destra su environment> import dataset> from Text (readr)

L'equivalente del codice

Import Text Data

File/URL:

/cloud/project/02-Transform-Data/babynames.csv

Browse...

Data Preview:

year *(double)*	sex *(logical)*	name *(character)*	n *(double)*	prop *(double)*
1880	FALSE	Mary	7065	0.07238359
1880	FALSE	Anna	2604	0.02667896
1880	FALSE	Emma	2003	0.02052149
1880	FALSE	Elizabeth	1939	0.01986579

Previewing first 50 entries.

Import Options:

Name: babynames

Skip: 0

☑ First Row as Names

☑ Trim Spaces

☑ Open Data Viewer

Delimiter: Comma

Quotes: Default

Locale: Configure...

Escape: None

Comment: Default

NA: Default

Code Preview:

```
library(readr)
babynames <- read_csv("02-Transform-Data/babynames.csv")
View(babynames)
```

Reading rectangular data using readr

Import

Cancel

FUTURE EDUCATION MODENA

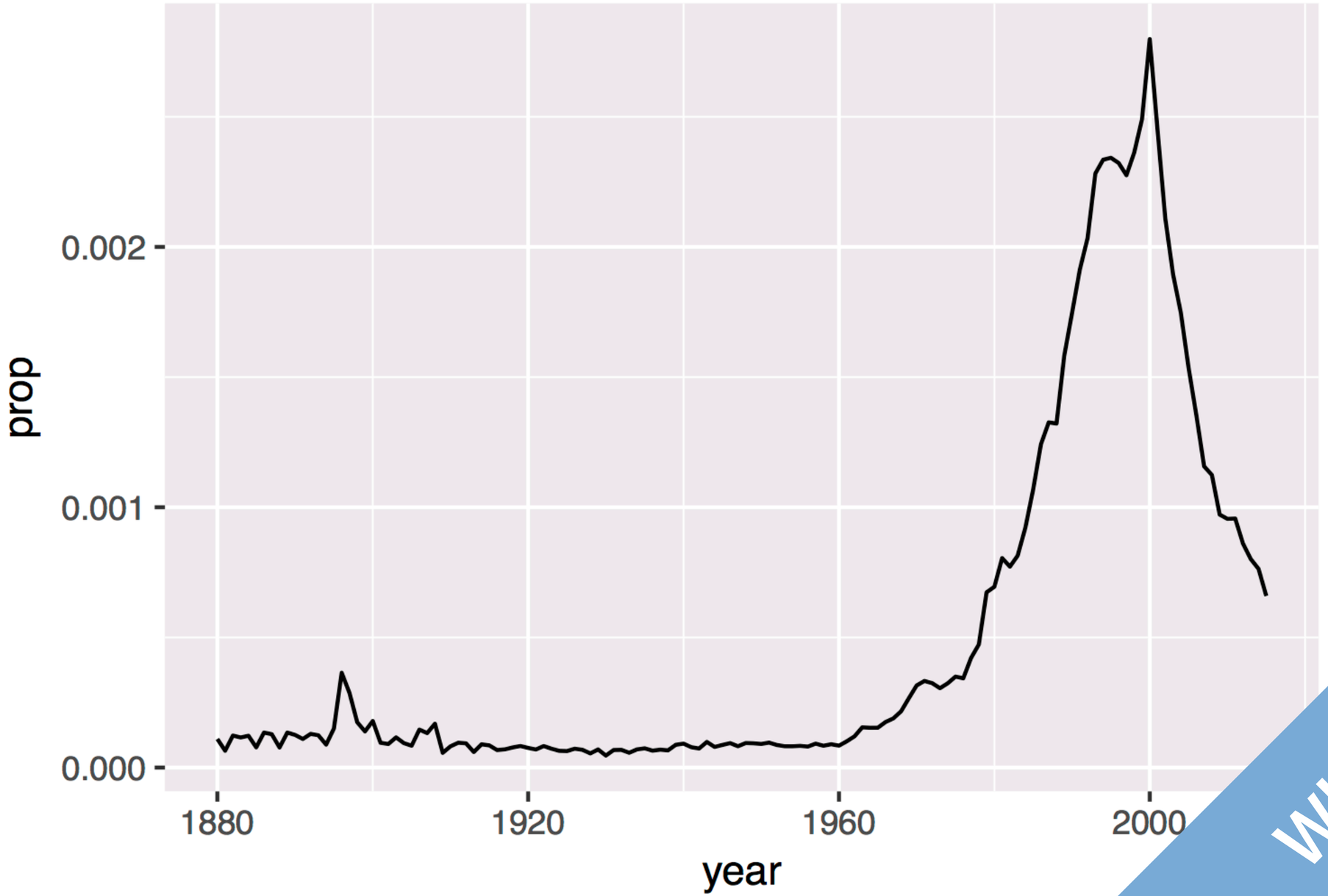
babynames

year <dbl>	sex <chr>	name <chr>	n <dbl>	prop <dbl>
1880	F	Mary	7065	0.07238359
1880	F	Anna	2604	0.02667896
1880	F	Emma	2003	0.02052149
1880	F	Elizabeth	1939	0.01986579
1880	F	Minnie	1746	0.01788843
1880	F	Margaret	1578	0.01616720
1880	F	Ida	1472	0.01508119
1880	F	Alice	1414	0.01448696
1880	F	Bertha	1320	0.01352390
1880	F	Sarah	1288	0.01319605

1–10 of 1,924,665 rows Previous 1 2 3 4 5 6 … 100 Next

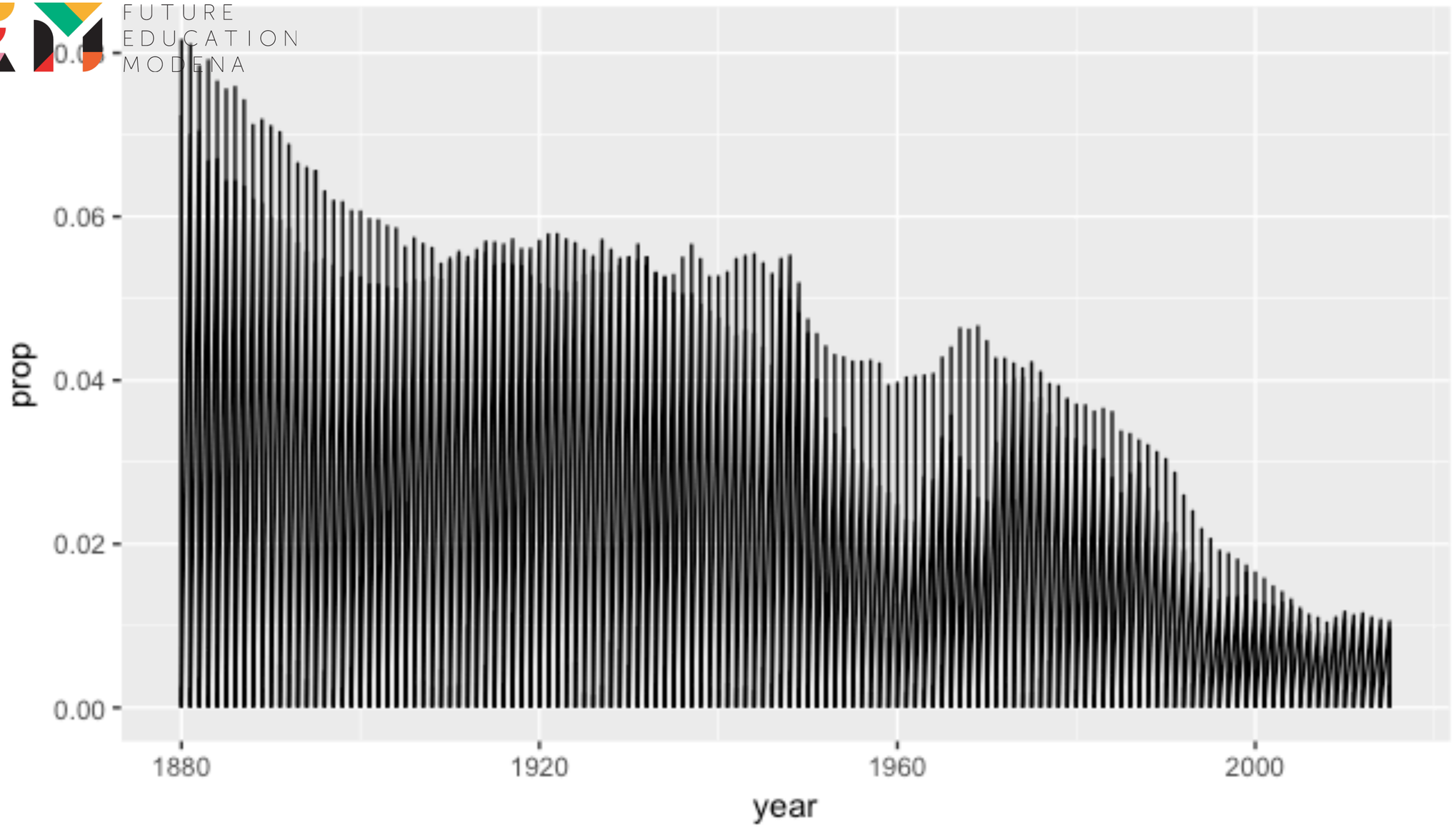
FUTURE EDUCATION MODENA

Proportion of boys with the name Garrett

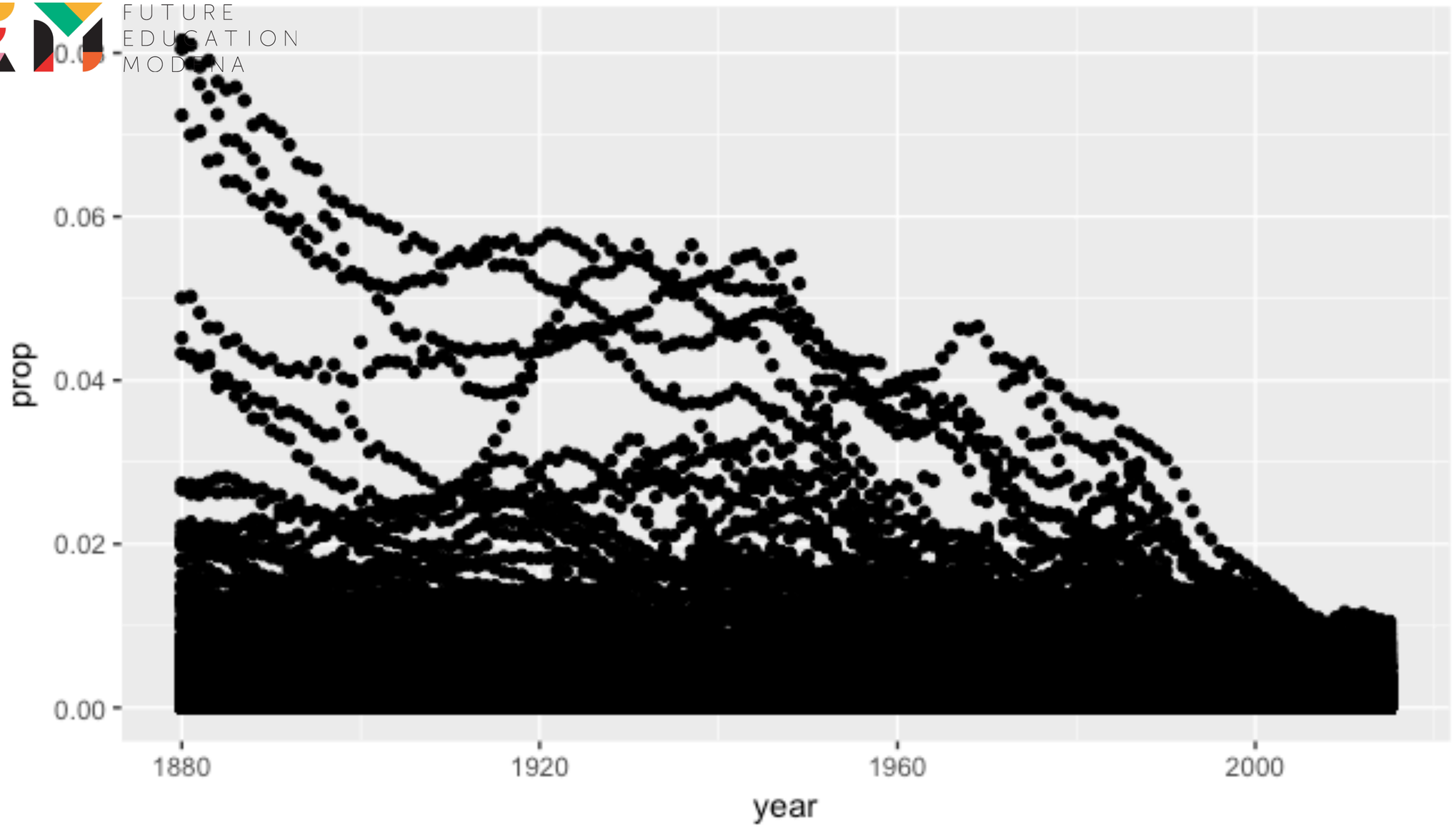


Which geom?

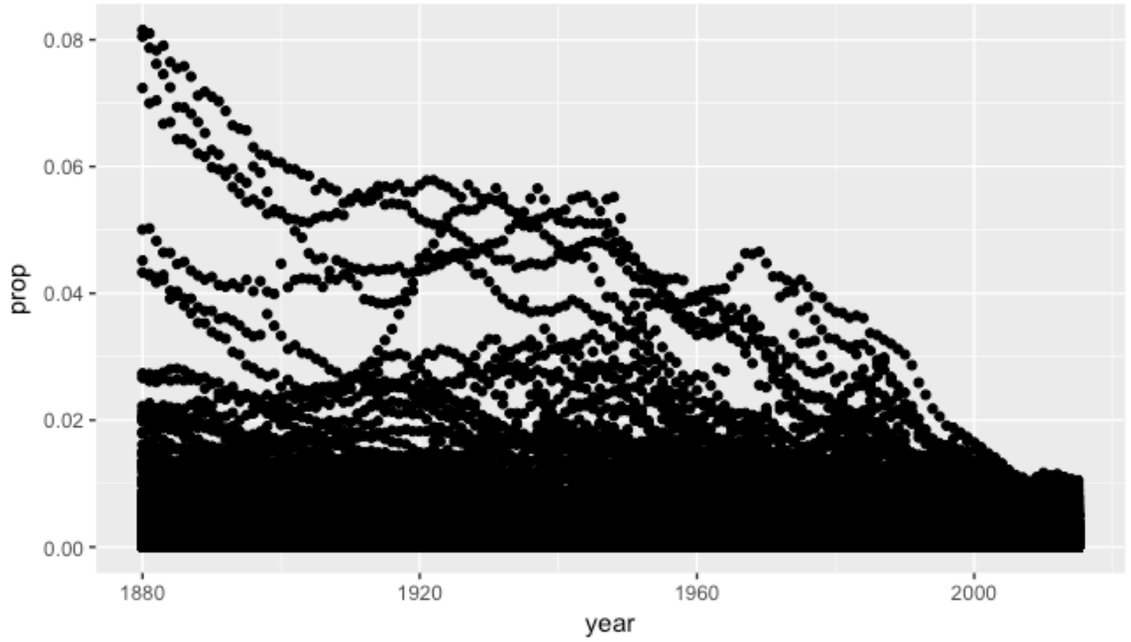
dplyr

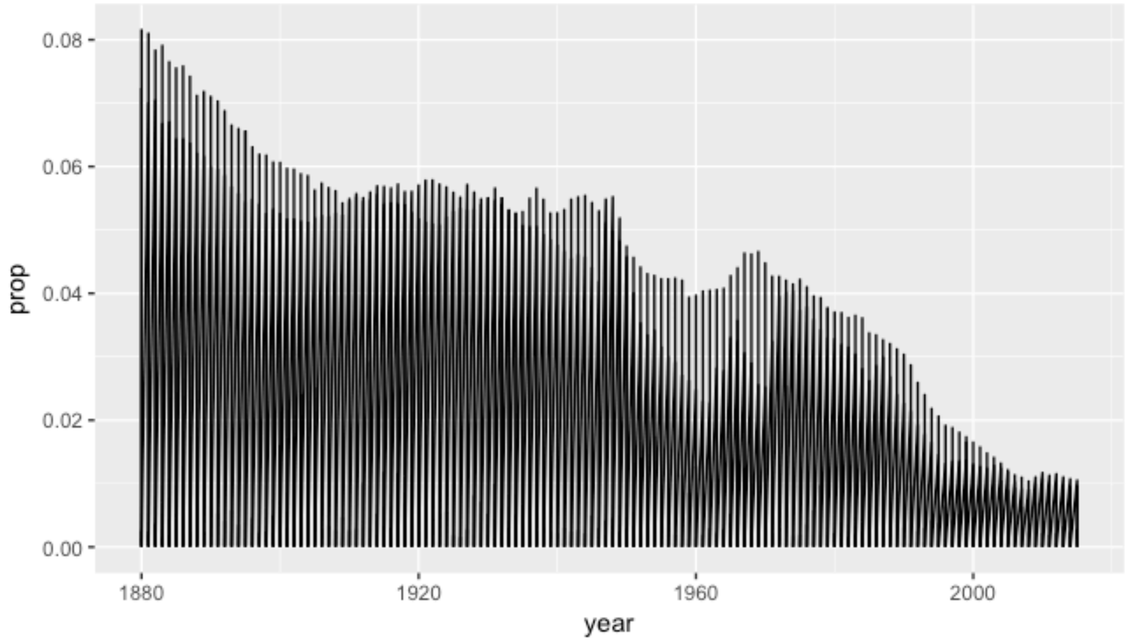


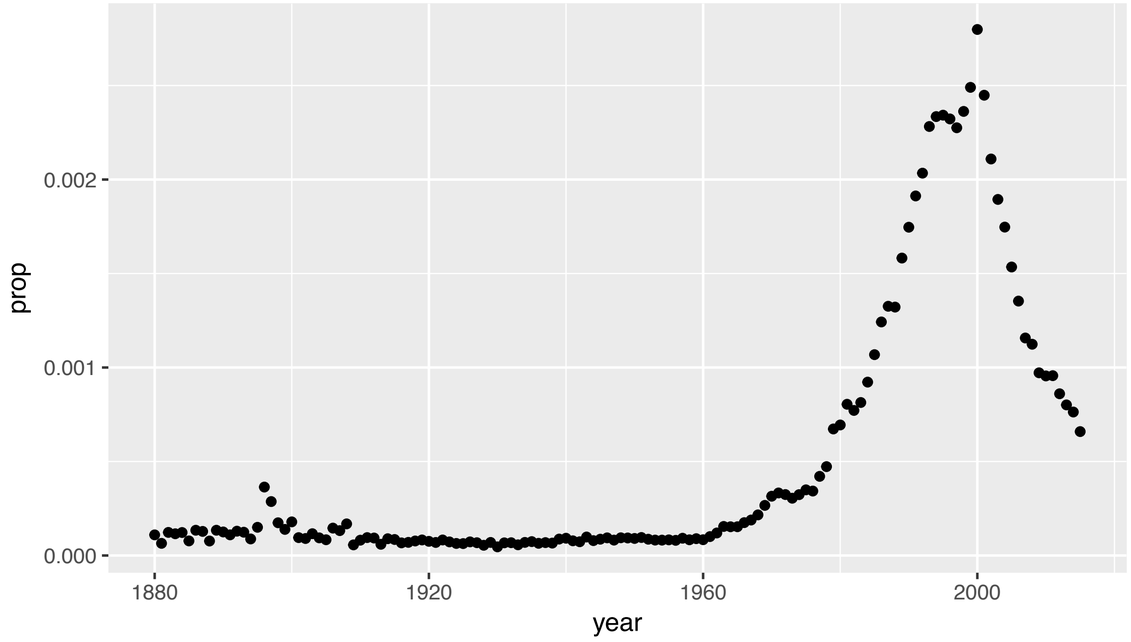

```
ggplot(data = babynames) +
   geom_line(mapping = aes(x = year, y = prop))
```

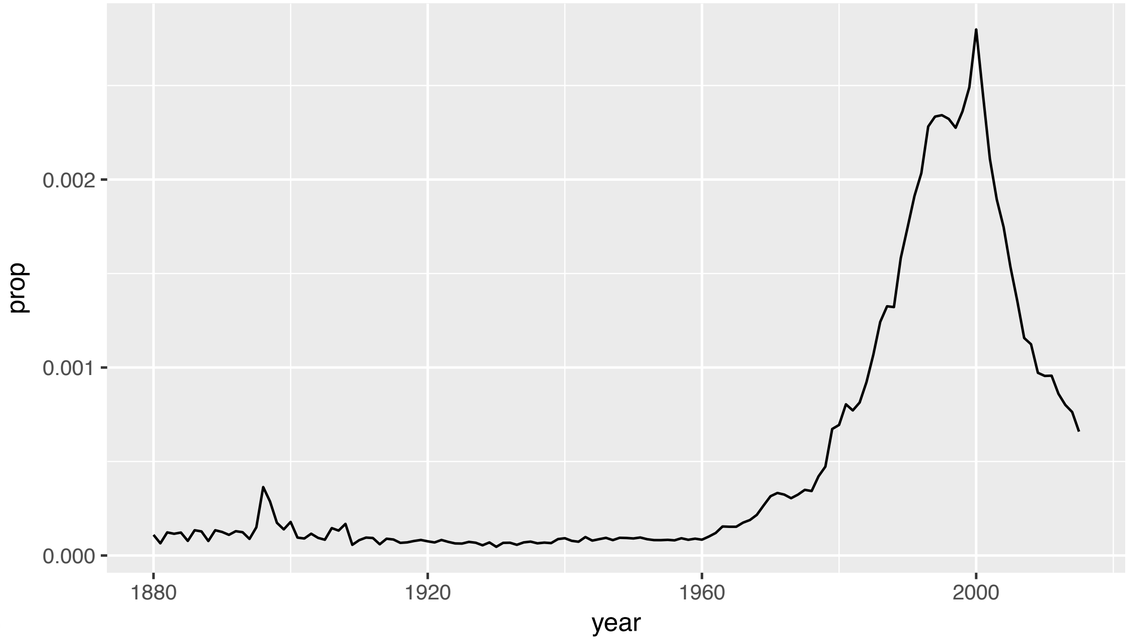



```
ggplot(data = babynames) +
   geom_point(mapping = aes(x = year, y = prop))
```


0.08
0.06
0.04
0.02
0.00
prop
1880
1920
1960
2000
year


0.08
0.06
0.04
0.02
0.00
prop
1880
1920
1960
2000
year


0.002
0.001
0.000
prop
1880
1920
1960
2000
year


0.002
0.001
0.000
prop
1880
1920
1960
2000
year

FUTURE EDUCATION MODENA

Come isolare le osservazioni?

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081
1881	M	William	8524	0,0787
1881	M	James	5442	0,0503
1881	M	Charles	4664	0,0431
1881	M	Garrett	7	0,0001
1881	M	Gideon	7	0,0001

year	sex	name	n	prop
1880	M	Garrett	13	0,0001
1881	M	Garrett	7	0,0001
…	…	Garrett	…	…

dplyr

dplyr

dplyr

Un pacchetto che trasforma i dati.

dplyr implementa una grammatica per trasformare dati tabulari.

FUTURE EDUCATION MODENA

Isolare i dati

select() - extract **variables (colonne)**

filter() - extract **cases (righe)**

arrange() - reorder **cases (righe)**

select()

FUTURE EDUCATION MODENA

select()

Extract columns by name.

```
select(data, …)
```

data frame to transform

name(s) of columns to extract (or a select helper function)

FUTURE EDUCATION MODENA

select()

Extract columns by name.

```
select(babynames, name, prop)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

name	prop
John	0,0815
William	0,0805
James	0,0501
Charles	0,0451
Garrett	0,0001
John	0,081

Your Turn 2

Modificate il codice per selezionare solo la colonna n:

```
select(babynames, name, prop)
```

```
select(babynames, n)
#       n
#    <int>
# 1   7065
# 2   2604
# 3   2003
# 4   1939
# 5   1746
# …     …
```

FUTURE EDUCATION MODENA

select() helpers

: - Seleziona l'intervallo di colonne

```
select(mpg, cty:class)
```

- - Seleziona tutte le colonne tranne

```
select(mpg, -c(cty, hwy))
```

starts_with() - Seleziona le colonne che iniziano con...

```
select(mpg, starts_with("c"))
```

ends_with() - Seleziona le colonne che finiscono con...

```
select(mpg, ends_with("y"))
```

FUTURE EDUCATION MODENA

select() helpers

contains() - Seleziona le colonne i cui nomi contengono…

```
select(mpg, contains("d"))
```

matches() - Seleziona le colonne i cui nomi corrispondono *regular expression*

```
select(mpg, matches("^.{4}$"))
```

one_of() - Seleziona le colonne i cui nomi sono uno di un insieme

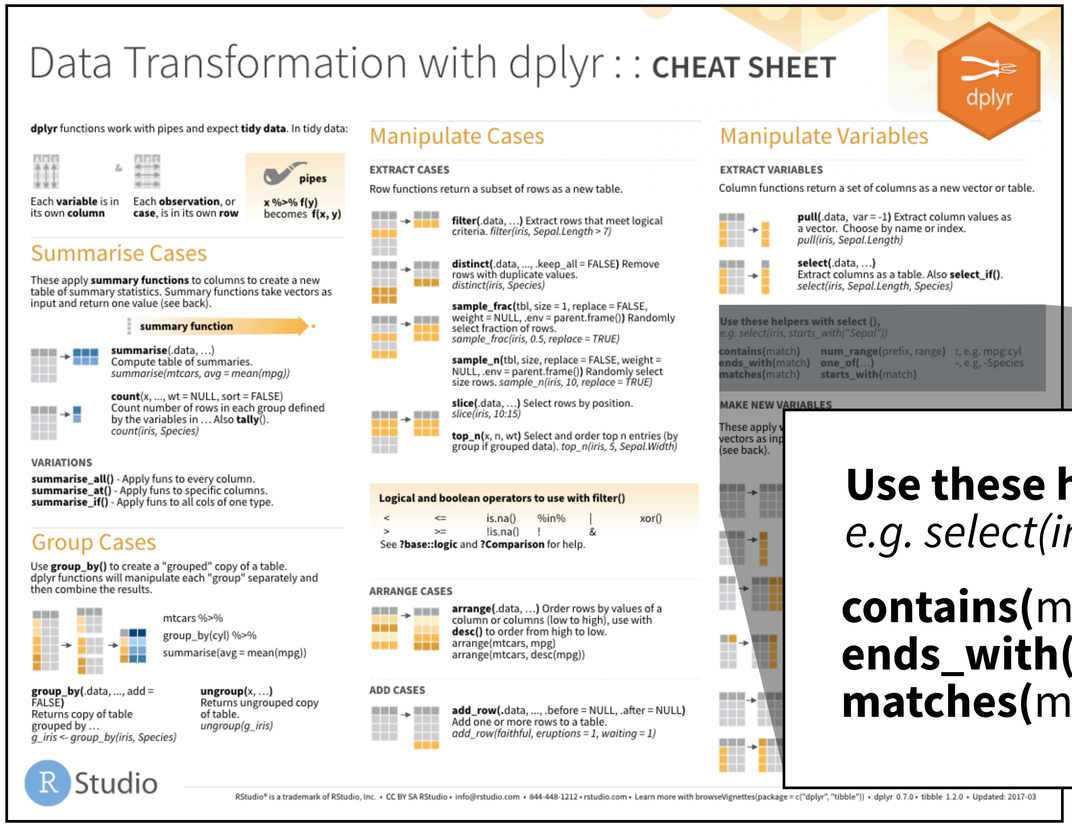
```
select(mpg, one_of(c("fl", "fuel", "Fuel"))
```

num_range() - Seleziona le colonne denominate in stile prefisso e numero

```
select(mpg, num_range("x", 1:5))
```

FUTURE EDUCATION MODENA

select() helpers



Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match**)**	**num_range(**prefix, range**)**	**:**, e.g. mpg:cyl
ends_with(match**)**	**one_of(**…**)**	**-**, e.g, -Species
matches(match**)**	**starts_with(**match**)**	

Quiz

Quale di questi NON è un modo per selezionare le colonne **nome** e **n** insieme?

```
select(babynames, -c(year, sex, prop))
```

```
select(babynames, name:n)
```

```
select(babynames, starts_with("n"))
```

```
select(babynames, ends_with("n"))
```

Quiz

Quale di questi NON è un modo per selezionare le colonne nome e n insieme?

```
select(babynames, -c(year, sex, prop))
```

```
select(babynames, name:n)
```

```
select(babynames, starts_with("n"))
```

```
select(babynames, ends_with("n"))
```

filter()

filter()

Estrarre le righe che soddisfano i criteri logici.

```
filter(.data, … )
```

data frame to transform

uno o più test logici (il filtro restituisce ogni riga per la quale il test è VERO)

sintassi comune

Ogni funzione prende un data frame / tibble come primo argomento e restituisce un data frame / tibble.

```
filter(.data, … )
```

filter()

Estrarre le righe che soddisfano i criteri logici.

```
filter(babynames, name == "Garrett")
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

year	sex	name	n	prop
1880	M	Garrett	13	0,0001
1881	M	Garrett	7	0,0001
…	…	Garrett	…	…

filter()

Estrarre le righe che soddisfano i criteri logici.

```
filter(babynames, name == "Garrett")
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

= imposta
(non restituisce nulla)

== verifica se è uguale
(restituisce VERO o FALSO)

dplyr

Logical tests

`x < y`	Minore di
`x > y`	Maggiore di
`x == y`	uguale a
`x <= y`	Minore o uguale
`x >= y`	Maggiore o uguale
`x != y`	Non uguale
`x %in% y`	Appartiene a

```
x <- 1
x >= 2
# FALSE
```

```r
x <- c(1, 2, 3)
x >= 2
# FALSE TRUE TRUE
```

Pop Quiz

Qual è il risultato?

1 == 1

Pop Quiz

Qual è il risultato?

1 == 1

TRUE

Pop Quiz

Qual è il risultato?

`1 == NA`

Pop Quiz

Qual è il risultato?

1 == NA

NA

Pop Quiz

Qual è il risultato?

NA == NA

Pop Quiz

Qual è il risultato?

NA == NA

NA

Pop Quiz

Qual è il risultato?

is.na(NA)

TRUE

FUTURE EDUCATION MODENA

Logical tests

`x < y`	Minore di
`x > y`	Maggiore di
`x == y`	uguale a
`x <= y`	Minore o uguale
`x >= y`	Maggiore o uguale
`x != y`	Non uguale
`x %in% y`	Appartiene a
`is.na(x)`	È NA
`!is.na(x)`	NON è NA

Your Turn 3

Usa filtro, babynames e gli operatori logici per trovare:

1. Tutte le righe in cui prop è maggiore o uguale a 0,08
2. Tutti i bambini di nome “Sea"

```
filter(babynames, prop >= 0.08)
#    year   sex    name      n       prop
# 1  1880     M    John  9655 0.08154630
# 2  1880     M William  9531 0.08049899
# 3  1881     M    John  8769 0.08098299
```

```
filter(babynames, name == "Sea")
#    year   sex  name      n         prop
# 1  1982     F   Sea     5 2.756771e-06
# 2  1985     M   Sea     6 3.119547e-06
# 3  1986     M   Sea     5 2.603512e-06
# 4  1998     F   Sea     5 2.580377e-06
```

FUTURE EDUCATION MODENA

Due errori comuni

1. Usare `=` al posto di `==`

```
filter(babynames, name = "Sea")
filter(babynames, name == "Sea")
```

2. Dimenticarsi le virgolette “”

```
filter(babynames, name == Sea)
filter(babynames, name == "Sea")
```

filter()

Estrarre le righe che soddisfano *ogni* criterio logico.

```
filter(babynames, name == "Garrett", year == 1880)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

year	sex	name	n	prop
1880	M	Garrett	13	0,0001

filter()

Estrarre le righe che soddisfano *ogni* criterio logico.

```
filter(babynames, name == "Garrett" & year == 1880)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

→

year	sex	name	n	prop
1880	M	Garrett	13	0,0001

FUTURE EDUCATION MODENA

Operatori Booleani

?base::Logic

`a & b`	and
`a | b`	or
`xor(a,b)`	exactly or
`!a`	not
`( )`	To group tests . & evaluates before |

$$x >= 2 \ \& \ x < 3$$

TRUE & TRUE

TRUE

Your Turn 4

Usa gli operatori booleani per modificare il codice qui sotto per restituire solo le righe che contengono:

- Ragazzi di nome Sue

- Nomi che sono stati usati esattamente da 5 o 6 bambini nel 1880

- Nomi che sono uno di Acura, Lexus, o Yugo

```
filter(babynames, name == "Sea" | name == "Anemone")
```

```
filter(babynames, name == "Sue", sex == "M")
#    year   sex  name      n          prop
# 1  1917 M      Sue         7    0.0000073
# 2  1927 M      Sue         5    0.0000043
# …     … …         …         …             …
```

```
filter(babynames, (n == 5 | n == 6) & year == 1880)
#    year   sex     name      n          prop
# 1  1880      F    Abby      6 6.147289e-05
# 2  1880      F  Aileen      6 6.147289e-05
# …     …      …         …      …             …
```

Parentheses matter

```
filter(babynames, name == "Acura" | name =="Lexus" |name =="Yugo")
#    year   sex  name      n          prop
# 1  1990      F Lexus    36 1.752932e-05
# 2  1990      M Lexus    12 5.579156e-06
# …     …      …      …      …             …
```

Altri due errori comuni

3. Collassare più test in uno solo

```
filter(babynames, 10 < n < 20)
filter(babynames, 10 < n, n < 20)
```

4. Mettere insieme molti test (quando si potrebbe usare %in%)

```
filter(babynames, n == 5 | n == 6 | n == 7 | n == 8)
filter(babynames, n %in% c(5, 6, 7, 8))
```

```
filter(babynames, name == "Sue", sex == "M")
#    year   sex  name      n          prop
# 1  1917 M      Sue       7    0.0000073
# 2  1927 M      Sue       5    0.0000043
# …     … …        …        …           …
```

```
filter(babynames, (n == 5 | n == 6) & year == 1880)
#    year   sex    name      n          prop
# 1  1880     F    Abby      6 6.147289e-05
# 2  1880     F  Aileen      6 6.147289e-05
# …     …     …       …      …           …
```

```
filter(babynames, name == "Acura" | name =="Lexus" |name =="Yugo")
#    year   sex  name      n          prop
# 1  1990     F Lexus    36 1.752932e-05
# 2  1990     M Lexus    12 5.579156e-06
# …     …     …     …      …           …
```

arrange()

FUTURE EDUCATION MODENA

arrange()

Riordina i valori dal più piccolo al più grande

```
arrange(.data, …)
```

data frame to transform

one or more columns to order by (additional columns will be used as tie breakers)

FUTURE EDUCATION MODENA

arrange()

Riordina i valori dal più piccolo al più grande

```
arrange(babynames, n)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

year	sex	name	n	prop
1880	M	Garrett	13	0,0001
1880	M	Charles	5348	0,0451
1880	M	James	5927	0,0501
1881	M	John	8769	0,081
1880	M	William	9532	0,0805
1880	M	John	9655	0,0815

dplyr

desc()

Riordina i valori dal più GRANDE al più PICCOLO

```
arrange(babynames, desc(n))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1881	M	John	8769	0,081
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001

Help me

Qual è il più piccolo valore di **n**?

Qual è il più grande?

```
arrange(babynames, n, prop)
#      year   sex         name     n          prop
#   1  2007     M        Aaban     5 2.259872e-06
#   2  2007     M      Aareon      5 2.259872e-06
#   3  2007     M        Aaris     5 2.259872e-06
#   4  2007     M          Abd     5 2.259872e-06
#   5  2007     M  Abdulazeez     5 2.259872e-06
#   6  2007     M   Abdulhadi     5 2.259872e-06
#   7  2007     M  Abdulhamid     5 2.259872e-06
#   8  2007     M  Abdulkadir     5 2.259872e-06
#   9  2007     M Abdulraheem     5 2.259872e-06
# 10  2007     M  Abdulrahim     5 2.259872e-06
# … with 1,924,655 more rows
```

```
arrange(babynames, desc(n))
#      year   sex     name      n        prop
#  1   1947     F    Linda 99680 0.05483609
#  2   1948     F    Linda 96211 0.05521159
#  3   1947     M    James 94763 0.05102057
#  4   1957     M Michael 92726 0.04238659
#  5   1947     M   Robert 91646 0.04934237
#  6   1949     F    Linda 91010 0.05184281
#  7   1956     M Michael 90623 0.04225479
#  8   1958     M Michael 90517 0.04203881
#  9   1948     M    James 88588 0.04969679
# 10   1954     M Michael 88493 0.04279403
# … with 1,924,655 more rows
```

```
arrange(babynames, desc(prop))
#      year   sex     name      n       prop
#  1   1880     M     John   9655 0.08154630
#  2   1881     M     John   8769 0.08098299
#  3   1880     M  William   9531 0.08049899
#  4   1883     M     John   8894 0.07907324
#  5   1881     M  William   8524 0.07872038
#  6   1882     M     John   9557 0.07831617
#  7   1884     M     John   9388 0.07648751
#  8   1882     M  William   9298 0.07619375
#  9   1886     M     John   9026 0.07582198
# 10   1885     M     John   8756 0.07551791
# ... with 1,858,679 more rows
```

```
arrange(babynames, desc(n))
#      year   sex     name      n       prop
#  1   1947     F    Linda  99680 0.05483609
#  2   1948     F    Linda  96211 0.05521159
#  3   1947     M    James  94763 0.05102057
#  4   1957     M  Michael  92726 0.04238659
#  5   1947     M   Robert  91646 0.04934237
#  6   1949     F    Linda  91010 0.05184281
#  7   1956     M  Michael  90623 0.04225479
#  8   1958     M  Michael  90517 0.04203881
#  9   1948     M    James  88588 0.04969679
# 10   1954     M  Michael  88493 0.04279403
# ... with 1,858,679 more rows
```

Your Turn 5

Qual è stato il nome più popolare in un singolo anno? Usare desc() e arrange() per trovare:

- il più alto n
- il più alto prop

%>%

Steps

```
boys_2015 <- filter(babynames, year == 2015, sex == "M")
boys_2015 <- select(boys_2015, name, n)
boys_2015 <- arrange(boys_2015, desc(n))
boys_2015
```

1. Filtrare i nomi dei bambini solo ai ragazzi (M) nati nel 2015
2. Seleziona le colonne **nome** e **n** dal risultato
3. Disponi queste colonne in modo che i nomi più popolari appaiano in cima.

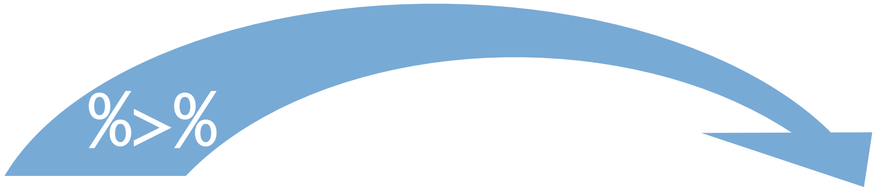
Steps

```
boys_2015 <- filter(babynames, year == 2015, sex == "M")
boys_2015 <- select(boys_2015, name, n)
boys_2015 <- arrange(boys_2015, desc(n))
boys_2015
```

Steps

```
arrange(select(filter(babynames, year == 2015,
  sex == "M"), name, n), desc(n))
```

The pipe operator %>%




```
babynames    %>%    filter(_____, n == 99680)
```

Passa il risultato a sinistra nel primo argomento della funzione a destra. Così, per esempio, questi fanno la stessa cosa. Provate.

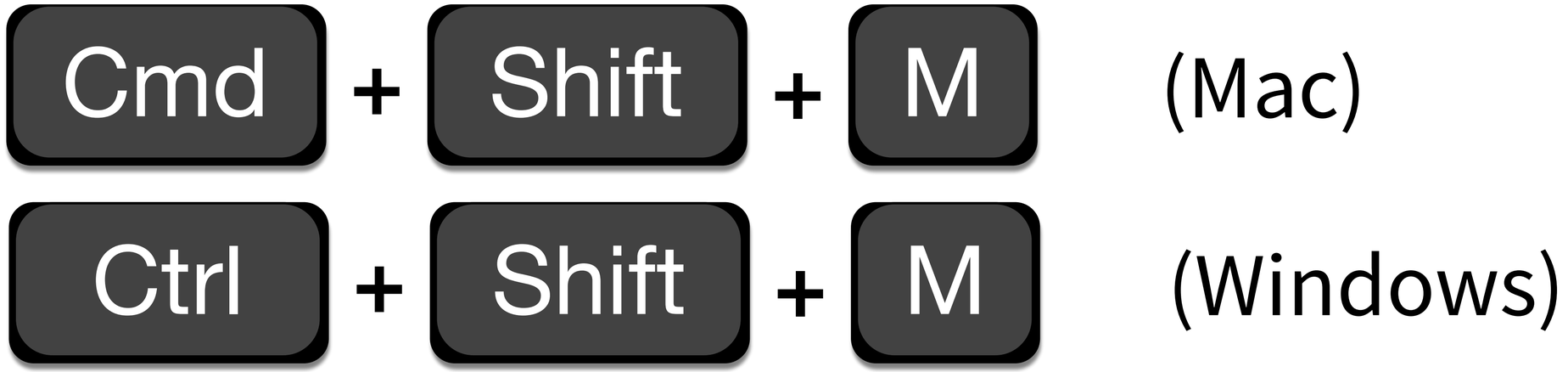
```
filter(babynames, n == 99680)
babynames %>% filter(n == 99680)
```

Pipes

```
babynames
boys_2015 <- filter(babynames, year == 2015, sex == "M")
boys_2015 <- select(boys_2015, name, n)
boys_2015 <- arrange(boys_2015, desc(n))
boys_2015
```

```
babynames %>%
  filter(year == 2015, sex == "M") %>%
  select(name, n) %>%
  arrange(desc(n))
```

Scorciatoia per digitare %>%



Your Turn 5

Usate %>% per scrivere una sequenza di funzioni che:

1. Filtra i nomi dei bambini per le ragazze (F) nate nel 2017, poi...
2. Seleziona il **nome** e le colonne **n**, poi...
3. Organizza i risultati in modo che i **nomi** più popolari siano vicini alla cima.

```
babynames %>%
  filter(year == 2017, sex == "F") %>%
  select(name, n) %>%
  arrange(desc(n))
#    name          n
#   1 Emma       19738
#   2 Olivia     18632
#   3 Ava        15902
#   4 Isabella   15100
#   5 Sophia     14831
#   6 Mia        13437
#   7 Charlotte  12893
#   8 Amelia     11800
#   9 Evelyn     10675
## … with 20,170 more rows
```

Payoff!

Your Turn 6 - Exam

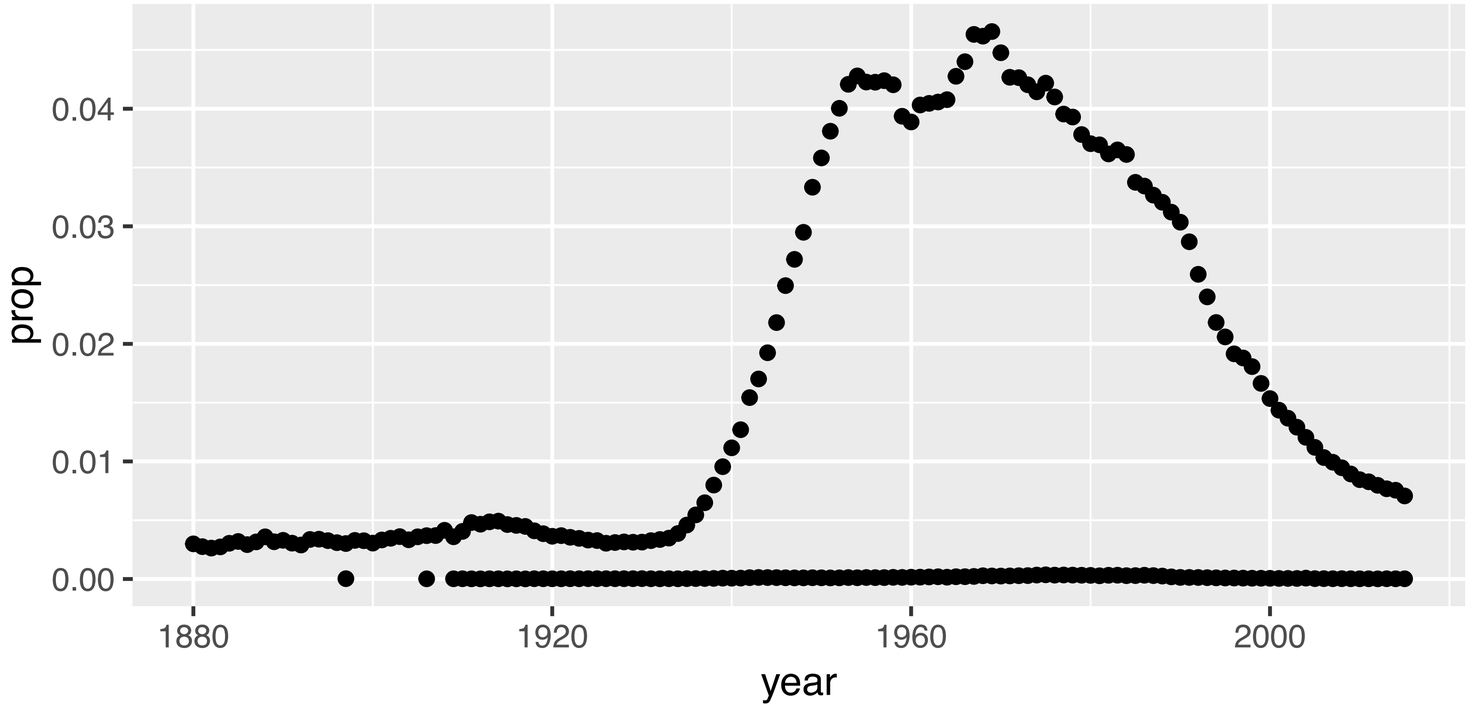
1. Ritagliare babynames alle sole righe che contengono il nome “**Beyonce**” e il sesso F
2. Seleziona il risultato alle sole colonne che appariranno nel tuo grafico (non strettamente necessario, ma pratica utile)
3. Traccia i risultati come un grafico a linee con **l'anno** sull'asse x e **prop** sull'asse y

```
babynames %>%
  filter(name == "Beyonce", sex == “F") %>%
  select(year, prop) %>%
  ggplot() +
    geom_line(mapping = aes(year, prop))
```

Plotting groups

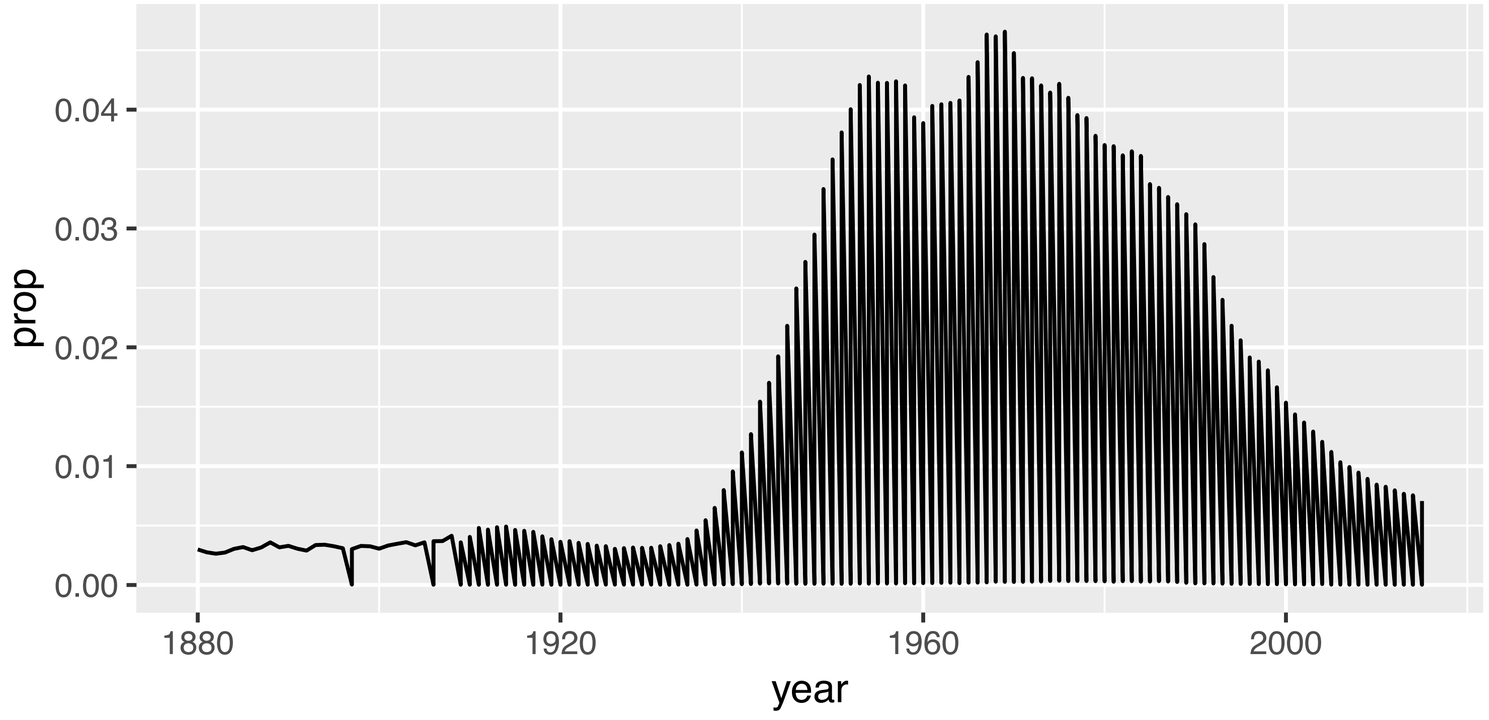
```
babynames %>%
   filter(name == "Michael") %>%
   ggplot() +
      geom_point(mapping = aes(year, prop))
```

non filtrato per SEX

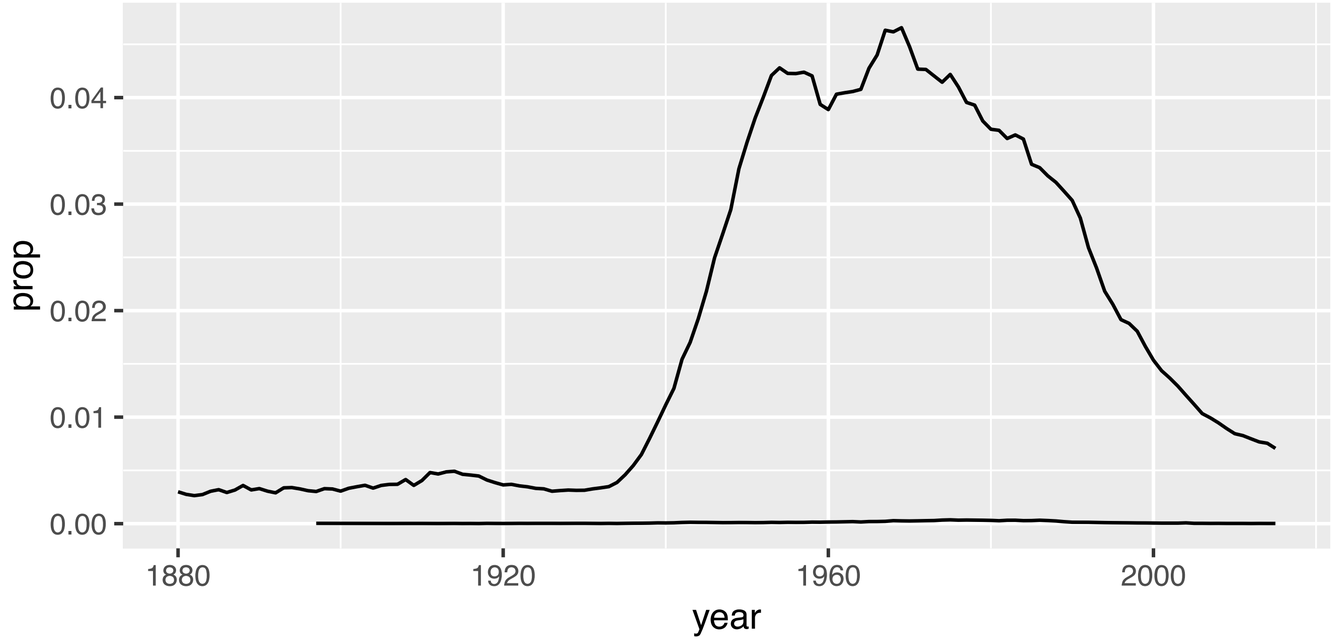


dplyr

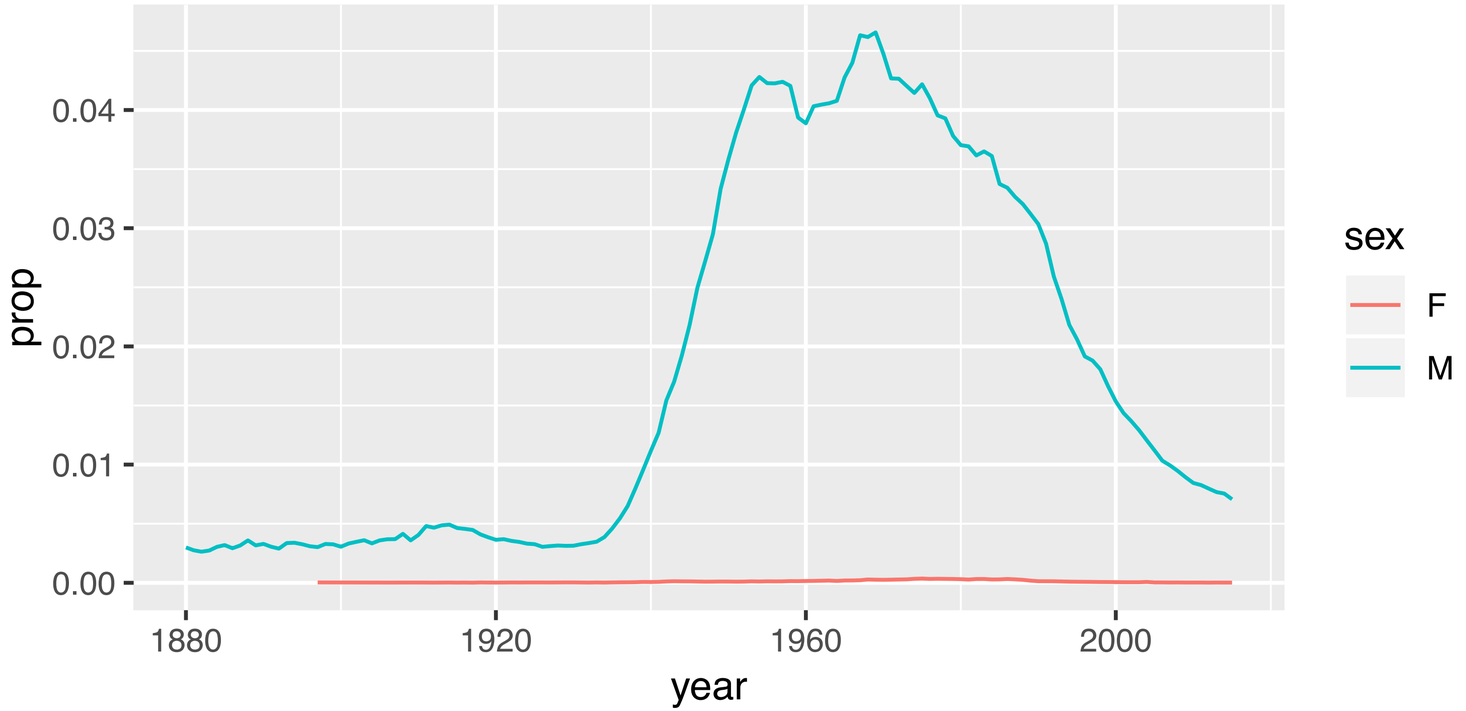
```
babynames %>%
  filter(name == "Michael") %>%
  ggplot() +
    geom_line(mapping = aes(year, prop))
```



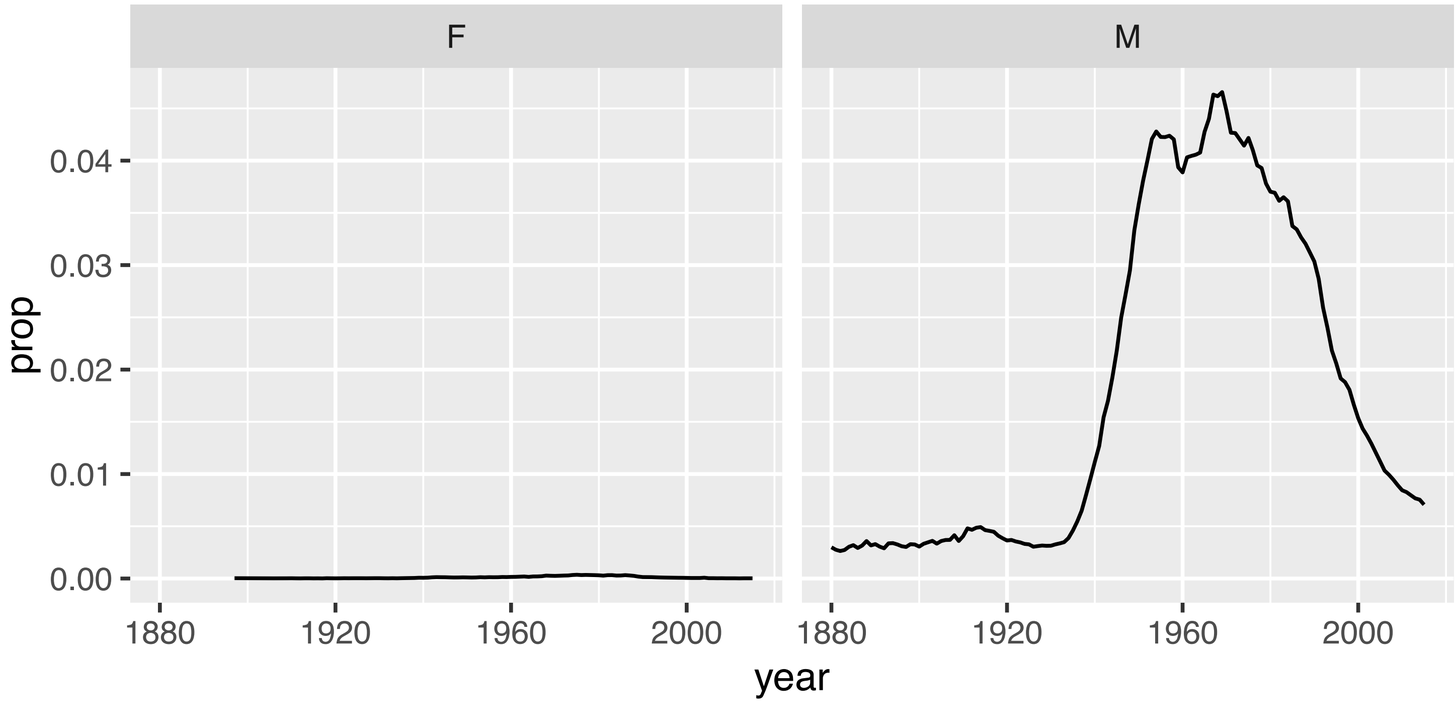
```
babynames %>%
  filter(name == "Michael") %>%
  ggplot() +
    geom_line(mapping = aes(year, prop, group = sex))
```



```
babynames %>%
   filter(name == "Michael") %>%
   ggplot() +
      geom_line(mapping = aes(year, prop, color = sex))
```



```
babynames %>% filter(name == "Michael") %>%
  ggplot() +
    geom_line(mapping = aes(year, prop)) +
    facet_wrap(~ sex)
```



Quali sono i nomi più popolari?

Quiz

Abbiamo abbastanza informazioni per:

1. Calcolare il numero totale di bambini con ogni nome?

FUTURE EDUCATION MODENA

Deriving information

summarise() - summarise **variables**

group_by() - group **cases**

mutate() - create new **variables**

summarise()

summarise()

Calcola la tabella dei riassunti.

```
babynames %>% summarise(total = sum(n), max = max(n))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

total	max
348120517	99686

Your Turn 7

Completate il codice per estrarre le righe dove **nome == "Khaleesi"**. Poi usate **summarise**() e **sum**() e **min**() per trovare:

1. Il numero totale di bambini chiamati Khaleesi
2. Il primo anno in cui Khaleesi è apparsa nei dati

```
babynames %>%
   filter(name == "Khaleesi") %>%
   summarise(total = sum(n), first = min(year))
#   total first
# 1  1964  2011
```

FUTURE EDUCATION MODENA

Summary functions

Prende un vettore come input.
Restituisce un singolo valore come output.

Summary Functions

TO USE WITH SUMMARISE ()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function

COUNTS

dplyr::**n()** - number of values/rows
dplyr::**n_distinct()** - # of uniques
sum(!is.na()) - # of non-NA's

LOCATION

mean() - mean, also **mean(!is.na())**
median() - median

LOGICALS

mean() - Proportion of TRUE's
sum() - # of TRUE's

POSITION/ORDER

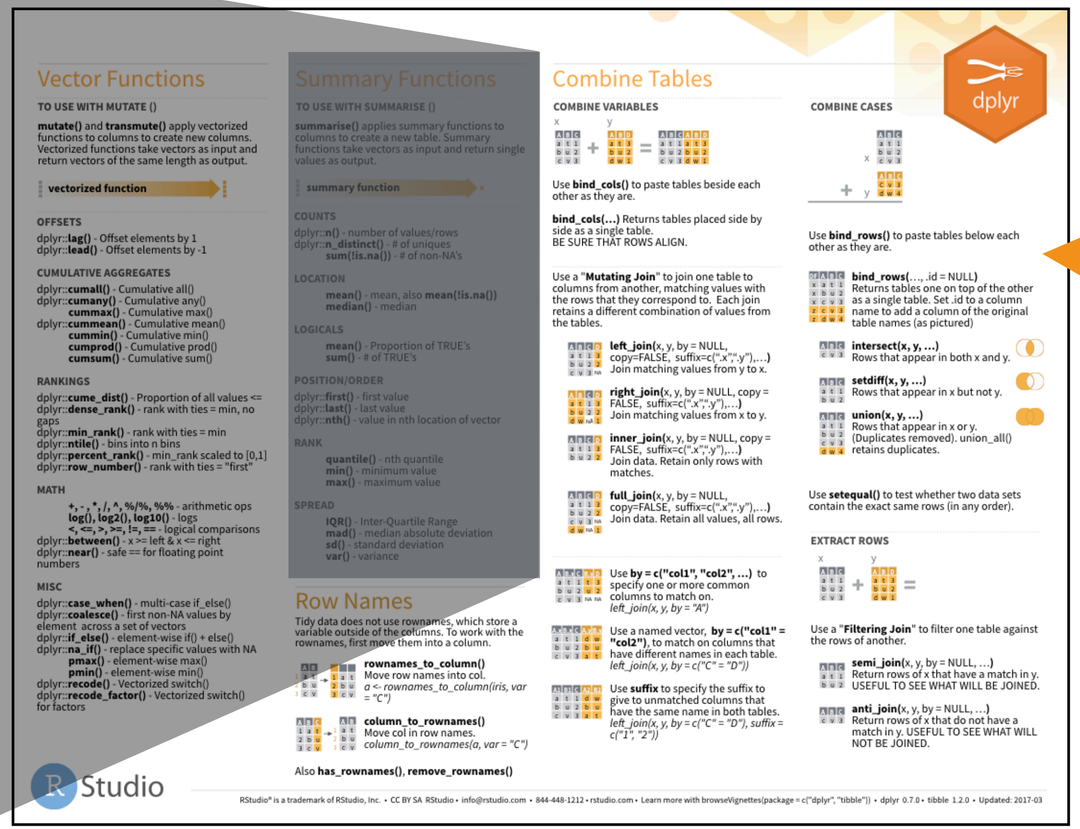
dplyr::**first()** - first value
dplyr::**last()** - last value
dplyr::**nth()** - value in nth location of vector

RANK

quantile() - nth quantile
min() - minimum value
max() - maximum value

SPREAD

IQR() - Inter-Quartile Range
mad() - median absolute deviation
sd() - standard deviation
var() - variance



on back

n()

Il numero di righe in un set di dati/gruppo

```
babynames %>% summarise(n = n())
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

→

n
1924665

n_distinct()

Il numero di valori distinti in una variabile

```
babynames %>% summarise(n = n(), nname = n_distinct(name))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

n	nname
1924665	97310

Come dovremmo definire la popolarità?

Un nome è popolare se:

1. SUM - un gran numero di bambini ha il nome quando si somma attraverso gli anni
2. RANK - si classifica costantemente tra i primi nomi di anno in anno.

```
babynames %>%
  filter(name == "Khaleesi" & sex == "F")
```

```
##     year sex   name         n       prop
##  1  2011 F     Khaleesi    28 0.0000145
##  2  2012 F     Khaleesi   146 0.0000754
##  3  2013 F     Khaleesi   243 0.000126
##  4  2014 F     Khaleesi   369 0.000189
##  5  2015 F     Khaleesi   341 0.000175
##  6  2016 F     Khaleesi   371 0.000192
##  7  2017 F     Khaleesi   466 0.000249
```

```
babynames %>%
  filter(name == "Khaleesi" & sex == "F") %>%
  summarise(total = sum(n))
##    total
## 1   1964
```

Possiamo farlo con ogni nome?

Grouping cases

```
pollution <- tribble(
        ~city,   ~size, ~amount,
  "New York", "large",       23,
  "New York", "small",       14,
    "London", "large",       22,
    "London", "small",       16,
   "Beijing", "large",      121,
   "Beijing", "small",       56
)
```

pollution

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	amount (μg/m^3)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6

```
pollution %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

FUTURE EDUCATION MODENA

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6

FUTURE
EDUCATION
MODENA

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
42	252	6

FUTURE EDUCATION MODENA

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

mean	sum	n
18,5	37	2
19,0	38	2
88,5	177	2

group_by() + summarise()

FUTURE EDUCATION MODENA

group_by()

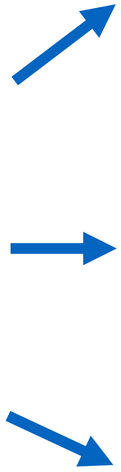
Raggruppa i casi per valori comuni di una o più colonne.

```
pollution %>%
  group_by(city)
```

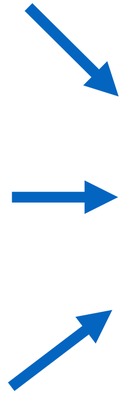
```
# A tibble: 6 x 3
# Groups:   city [3]
  city     size  amount
  <chr>    <chr>  <dbl>
1 New York large     23
2 New York small     14
```

group_by()

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

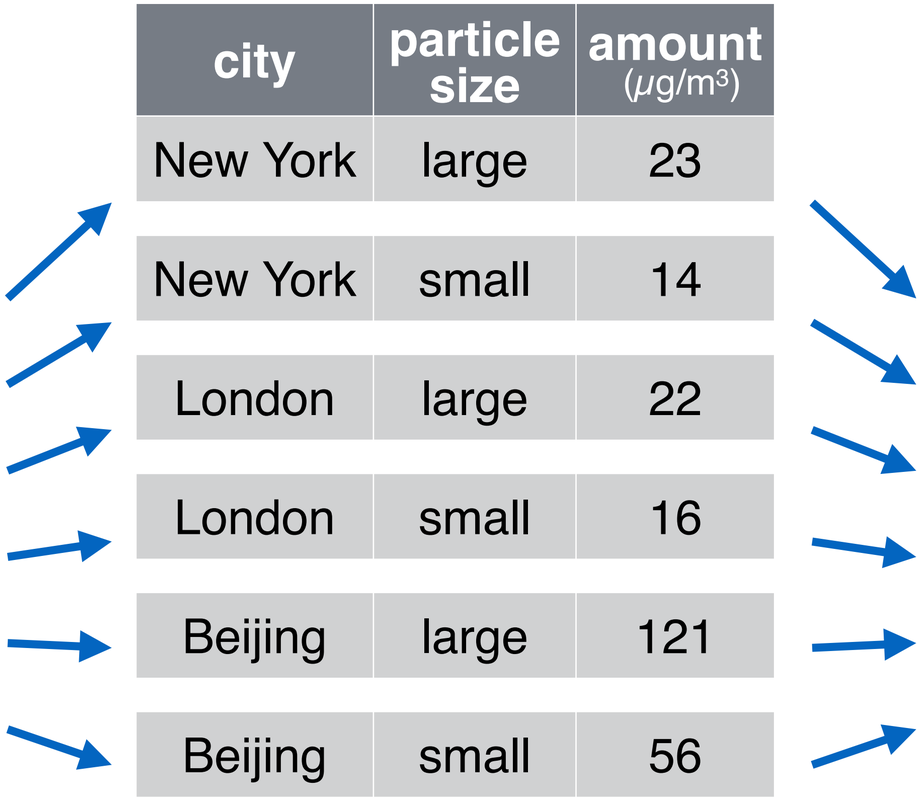


city	mean	sum	n
New York	18,5	37	2
London	19,0	38	2
Beijing	88,5	177	2

```
pollution %>%
  group_by(city) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

group_by()

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56



city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

city	particle size	mean	sum	n
New York	large	23	23	1
New York	small	14	14	1
London	large	22	22	1
London	small	16	16	1
Beijing	large	121	121	1
Beijing	small	56	56	1

```
pollution %>%
  group_by(city, size) %>%
  summarise(mean = mean(amount), sum = sum(amount), n = n())
```

group_by()

Groups cases by common values.

```
babynames %>%
  group_by(sex) %>%
  summarise(total = sum(n))
```

sex	total
F	172371079
M	175749438

ungroup()

Rimuove i criteri di raggruppamento da un frame di dati.

```
babynames %>%
  group_by(sex) %>%
  ungroup() %>%
  summarise(total = sum(n))
```

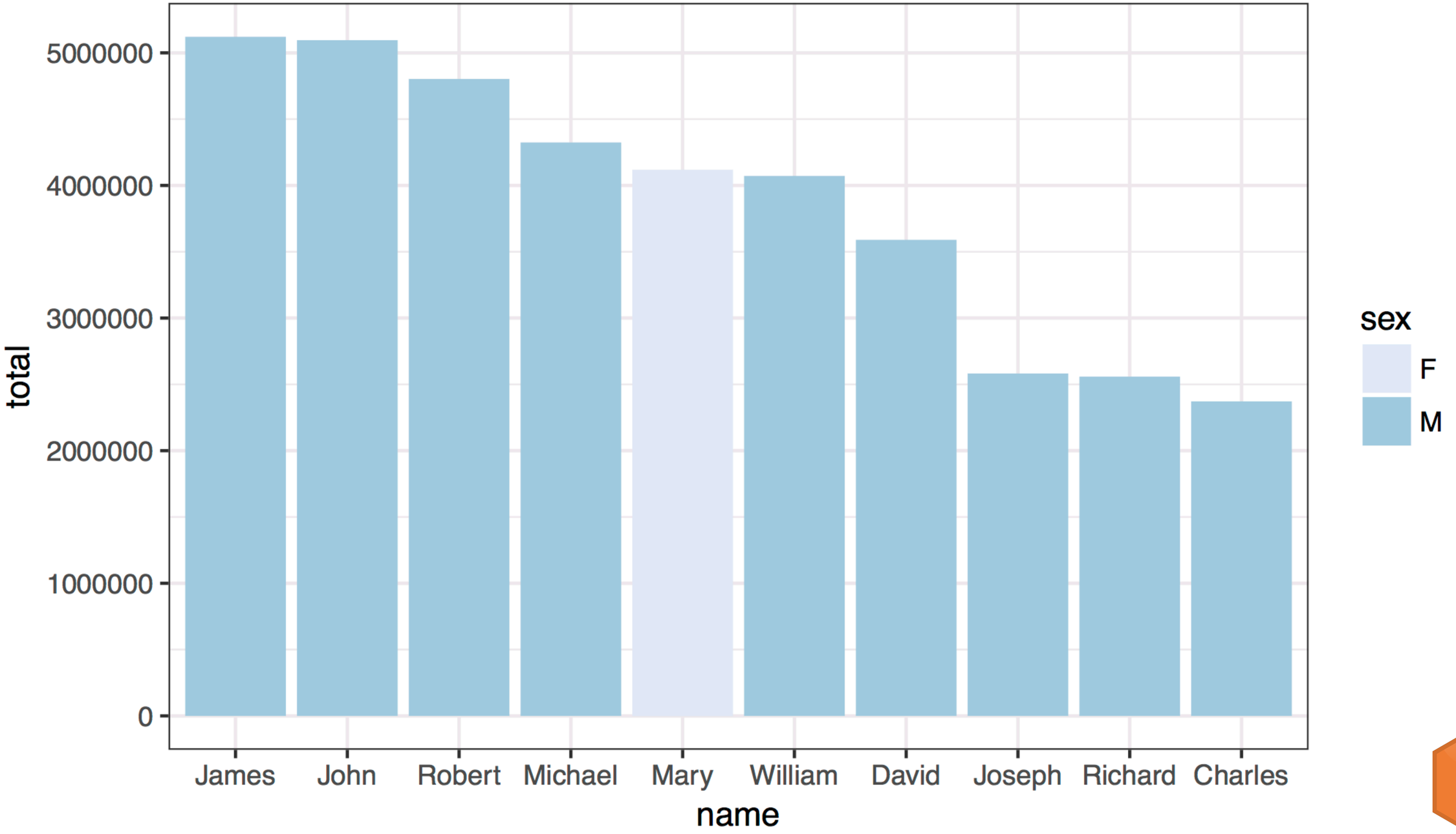
total
348120517

Your Turn 8

Completate il codice con group_by(), summarise() e arrange() per visualizzare le dieci combinazioni di nome e sesso più popolari. Calcolare la popolarità come il numero totale di bambini con un dato nome e sesso.

```
babynames %>%
  group_by(name, sex) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total))
#       name   sex   total
#  1   James     M 5120990
#  2    John     M 5095674
#  3  Robert     M 4803068
#  4 Michael     M 4323928
#  5    Mary     F 4118058
#  6 William     M 4071645
#  7   David     M 3589754
#  8  Joseph     M 2581785
#  9 Richard     M 2558165
# 10 Charles     M 2371621
# … with 107,963 more rows
```

FUTURE
EDUCATION
MODENA



dplyr

```
babynames %>%
  group_by(name, sex) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total)) %>%
  ungroup() %>%
  slice(1:10) %>%
  ggplot() +
    geom_col(mapping = aes(x = fct_reorder(name,
      desc(total)), y = total, fill = sex)) +
    theme_bw() +
    scale_fill_brewer() +
    labs(x = "name")
```

```
babynames %>%
  filter(name == "Khaleesi" & sex == "F") %>%
  summarise(total = sum(n))
##    total
## 1   1964
```

Can we do this for each name?

```
babynames %>%
  filter(name == "Khaleesi" & sex == "F") %>%
  summarise(total = sum(n))
##    total
## 1   1964

babynames %>%
  group_by(name, sex) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total))
```

```
babynames %>%
   filter(name == "Khaleesi" & sex == "F") %>%
   summarise(total = sum(n))
```

Raggruppa per le variabili che hai usato per ottenere il tuo test case

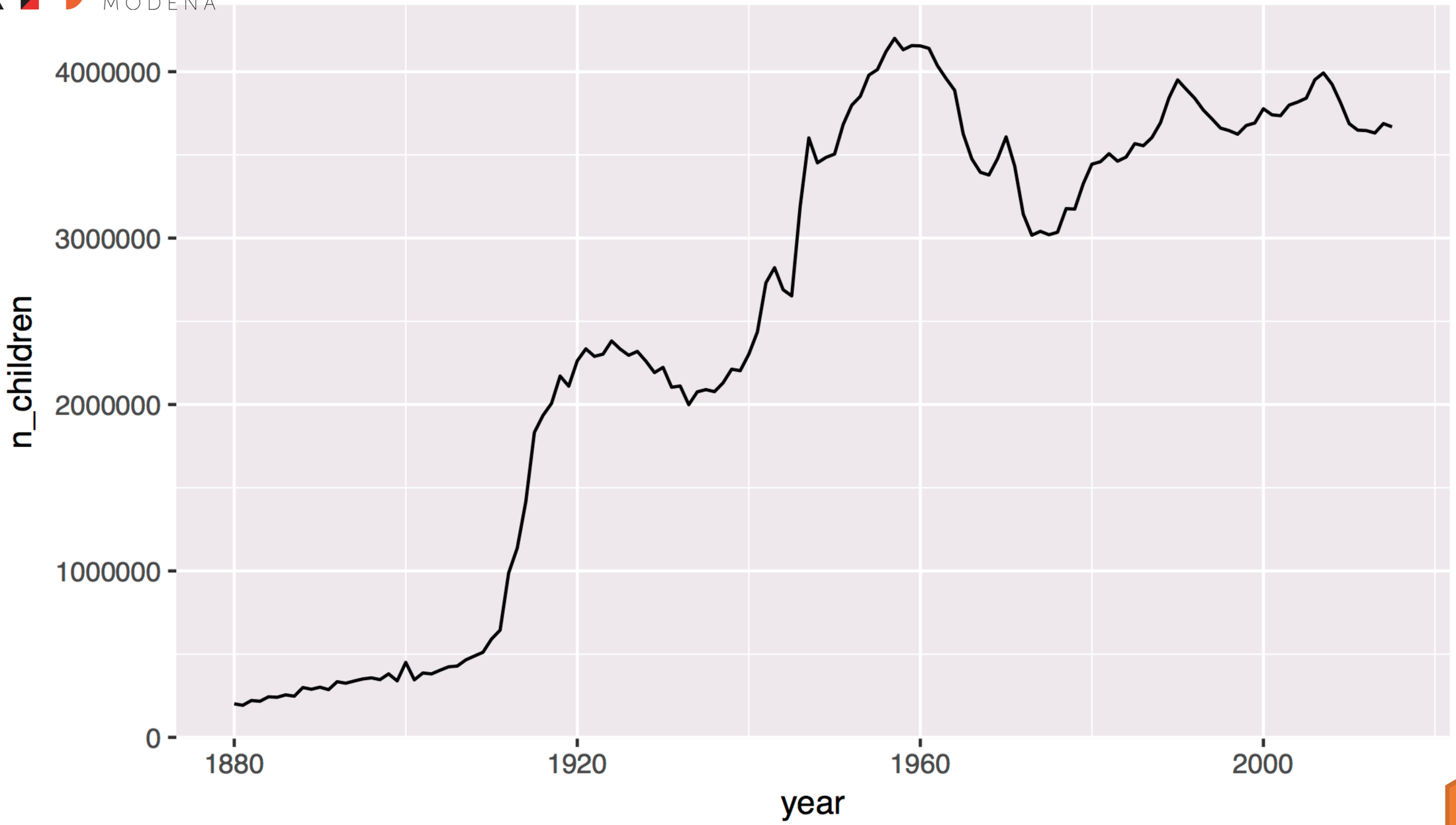
```
babynames %>%
   group_by(name, sex) %>%
   summarise(total = sum(n))
```

Your Turn 9

Usa group_by() per calcolare il numero totale di bambini nati per ogni anno.

Traccia i risultati come un grafico a linee: totale vs. anno.

```
babynames %>%
  group_by(year) %>%
  summarise(n_children = sum(n)) %>%
  ggplot() +
    geom_line(mapping = aes(x = year, y = n_children))
```


4000000
3000000
2000000
1000000
0
n_children
1880
1920
1960
2000
year

Qual è stato il nome più classificato per ogni anno?

mutate()

FUTURE EDUCATION MODENA

mutate()

Crea nuove colonne

```
babynames %>%
  mutate(percent = round(prop*100, 2))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

year	sex	name	n	prop	percent
1880	M	John	9655	0,0815	8,15
1880	M	William	9532	0,0805	8,05
1880	M	James	5927	0,0501	5,01
1880	M	Charles	5348	0,0451	4,51
1880	M	Garrett	13	0,0001	0,01
1881	M	John	8769	0,081	8,1

mutate()

Crea nuove colonne

```
babynames %>%
  mutate(percent = round(prop*100, 2), nper = round(percent))
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

year	sex	name	n	prop	percent	nper
1880	M	John	9655	0,0815	8,15	8
1880	M	William	9532	0,0805	8,05	8
1880	M	James	5927	0,0501	5,01	5
1880	M	Charles	5348	0,0451	4,51	5
1880	M	Garrett	13	0,0001	0,01	0
1881	M	John	8769	0,081	8,1	8

```
babynames %>%
  mutate(rank =                ?                )
```

FUTURE EDUCATION MODENA

Vectorized functions

Take a vector as input.

Return a vector of the same length as output.

Vector Functions

TO USE WITH MUTATE ()

mutate() and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function

OFFSETS

dplyr::**lag()** - Offset elements by 1
dplyr::**lead()** - Offset elements by -1

CUMULATIVE AGGREGATES

dplyr::**cumall()** - Cumulative all()
dplyr::**cumany()** - Cumulative any()
cummax() - Cumulative max()
dplyr::**cummean()** - Cumulative mean()
cummin() - Cumulative min()
cumprod() - Cumulative prod()
cumsum() - Cumulative sum()

RANKINGS

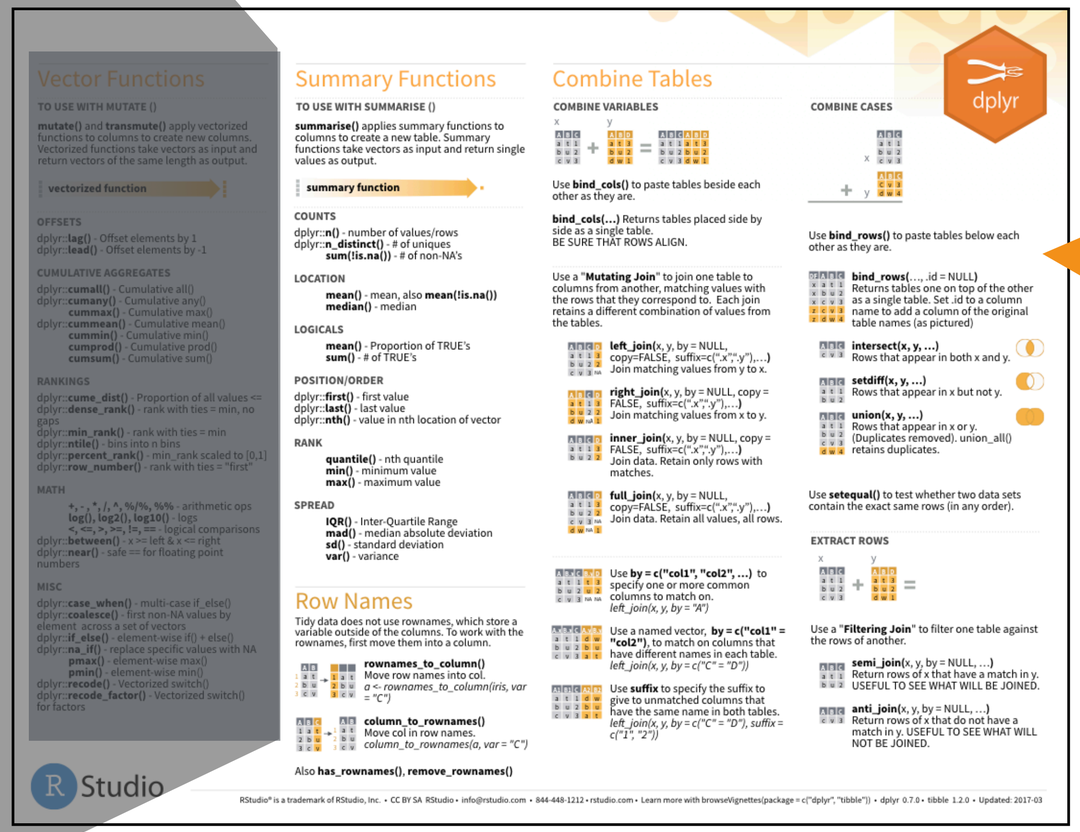
dplyr::**cume_dist()** - Proportion of all values <=
dplyr::**dense_rank()** - rank with ties = min, no gaps
dplyr::**min_rank()** - rank with ties = min
dplyr::**ntile()** - bins into n bins
dplyr::**percent_rank()** - min_rank scaled to [0,1]
dplyr::**row_number()** - rank with ties = "first"

MATH

+, - , *, /, ^, %/%, %% - arithmetic ops
log(), log2(), log10() - logs
<, <=, >, >=, !=, == - logical comparisons
dplyr::**between()** - x >= left & x <= right
dplyr::**near()** - safe == for floating point numbers

MISC

dplyr::**case_when()** - multi-case if_else()
dplyr::**coalesce()** - first non-NA values by element across a set of vectors
dplyr::**if_else()** - element-wise if() + else()
dplyr::**na_if()** - replace specific values with NA
pmax() - element-wise max()
pmin() - element-wise min()
dplyr::**recode()** - Vectorized switch()
dplyr::**recode_factor()** - Vectorized switch() for factors



min_rank()

A go-to ranking function (ties share the lowest rank)

```
min_rank(c(50, 100, 100, 1000))
# [1] 1 2 2 4
```

```
min_rank(desc(c(50, 100, 100, 1000)))
# [1] 4 2 2 1
```

Your Turn 10

Usare mutate() e min_rank() per classificare ogni riga in babynames dal più grande n al più basso n.

```
babynames %>%
  mutate(rank = min_rank(desc(prop)))


## A tibble: 1,924,665 x 6
#    year sex   name           n    prop  rank
# 1  1880 F     Mary        7065 0.0724    14
# 2  1880 F     Anna        2604 0.0267   709
# 3  1880 F     Emma        2003 0.0205  1131
# 4  1880 F     Elizabeth   1939 0.0199  1192
# 5  1880 F     Minnie      1746 0.0179  1427
# 6  1880 F     Margaret    1578 0.0162  1683
## … with 1,924,659 more rows
```

Your Turn 11

Raggruppa i nomi di bambini per anno e poi classifica nuovamente i dati. Filtrate i risultati alle sole righe in cui rank == 1.

```
babynames %>%
  group_by(year) %>%
  mutate(rank = min_rank(desc(prop))) %>%
  filter(rank == 1)


# A tibble: 138 x 6
# Groups:   year [138]
    year sex   name        n   prop  rank
 1  1880 M     John     9655 0.0815     1
 2  1881 M     John     8769 0.0810     1
 3  1882 M     John     9557 0.0783     1
# … with 135 more rows
```

FUTURE EDUCATION MODENA

Recap: Single table verbs

Extract variables with **select()**

Extract cases with **filter()**

Arrange cases, with **arrange()**.

Make tables of summaries with **summarise()**.

Make new variables, with **mutate()**.

FUTURE EDUCATION MODENA

select()

Extract columns by name.

```
select(babynames, n)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

n
9655
9532
5927
5348
13
8769

Still a dataframe

FUTURE EDUCATION MODENA

$

Extract column contents as a vector.

```
babynames$n
```

babynames

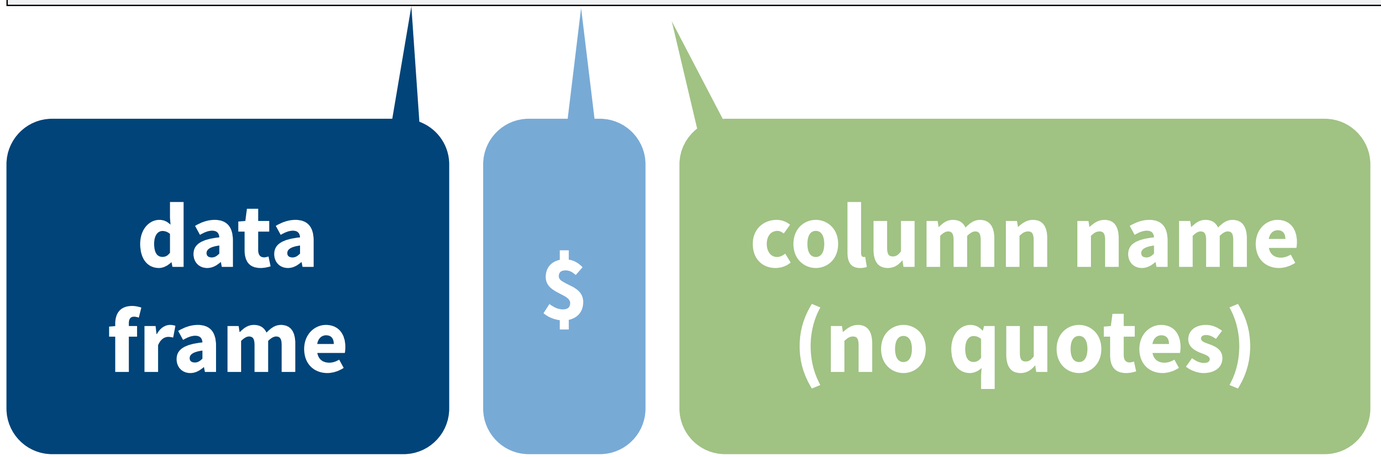
year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

→ 9655 9532 5927 5348 …

$

Extract column contents as a vector.

```
babynames$n
```



FUTURE EDUCATION MODENA

pull()

Pipe friendly version of $

```
babynames %>% pull(n)
```

babynames

year	sex	name	n	prop
1880	M	John	9655	0,0815
1880	M	William	9532	0,0805
1880	M	James	5927	0,0501
1880	M	Charles	5348	0,0451
1880	M	Garrett	13	0,0001
1881	M	John	8769	0,081

→ 9655 9532 5927 5348 …

Transform Data with