# La gestione dei colori in ggplot

Abbiamo visto che in ggplot2 è possibile:

- indicare direttamente i colori (`geom_point(col = "red")` o `geom_barplot(fill = "red")`); oppure
- mappare una variabile (`geom_point(aes(col = x))` o `geom_barplot(aes(fill = x))`);

vedi ggplot

Quando una variabile viene mappata per la selezione dei colori, viene prodotta anche una legenda.

## Le funzioni per le scale

A seconda del tipo di carattere rappresentato, vengono usate scale di colori diverse, ad esempio:

- **carattere continuo** vedi Grafici a dispersione, figura 1
- **carattere discreto** vedi Grafici a barre, figura 3
- **carattere discreto ordinato**: vedi Boxplot, figura 3

In particolare, per i caratteri continui, vengono utilizzate/create palette sequenziali (a gradiente), e divergenti (colori opposti e brillanti agli estremi, che sfumano al centro); per i caratteri discreti vengono invece utilizzate/create palette qualititative, con colori diversi fra di loro, ma di pari intensità/luminosità.

Tali scale possono essere modificate con una serie di funzioni che:

- iniziano con `scale_`;
- sono seguite da `color_` (o `colour_`, è uguale) se l'argomento da utilizzare è `col = `; da `fill_` se l'argomento da utilizzare è `fill = `.

Ad esempio se abbiamo `aes(col = x)`, useremo una funzione che inizia per `scale_color_…()`, mentre per `aes(fill = x)` indicheremo i colori con `scale_fill_…()`:

- `aes(col = x)` + `scale_color_…()`
- `aes(fill = x)` + `scale_fill_…()`

Tab. 1: Funzioni per i colori in ggplot

| **funzione (scale_color_/scale_fill_)** | **carattere** |
|---|---|
| brewer(palette = ) | discreto, scale *brewer* |

| **funzione (scale_color_/scale_fill_)** | **carattere** |
|---|---|
| distiller(palette = ) | continuo, scale *brewer* |
| gradient, gradient2, gradientn | continuo (sequenziale, divergente, toni di grigio) |
| manual | discreto, *per indicare i colori* |
| grey | discreto |
| hue | discreto |
| identity | *per usare una variabile* |

Queste funzioni consentono anche di attribuire un nome arbitrario alla legenda, o di eliminarla:

- `scale_color_gradient2("cavalli", …")` = la legenda avrà come titolo "cavalli" (figura 6)
- `scale_color_gradient2(guide = FALSE)` = la legenda verrà eliminata dal grafico (figura 1)

# ColorBrewer

```
# Pacchetti
library(tidyverse)
library(carData)
```

ColorBrewer è uno strumento online per la selezione di palette di colori, e che offre diverse opzioni di palette di colori accessibili e distinguibili, suddivise fra sequenziali, divergenti e qualitative (<https://colorbrewer2.org>).

Per usare queste gamme di colori, integrate in *ggplot2*, si possono usare:

- `scale_color_brewer()` e `scale_fill_brewer()`per valori discreti
- `scale_color_distiller()` e `scale_fill_distiller()` per valori continui

Queste funzioni richiedono di indicare la palette *brewer* con un numero, ma possono essere usate anche con quelle standard, indicandole per nome. Queste saranno appropriatamente trasformate in rapporto al tipo di carattere rappresentato.

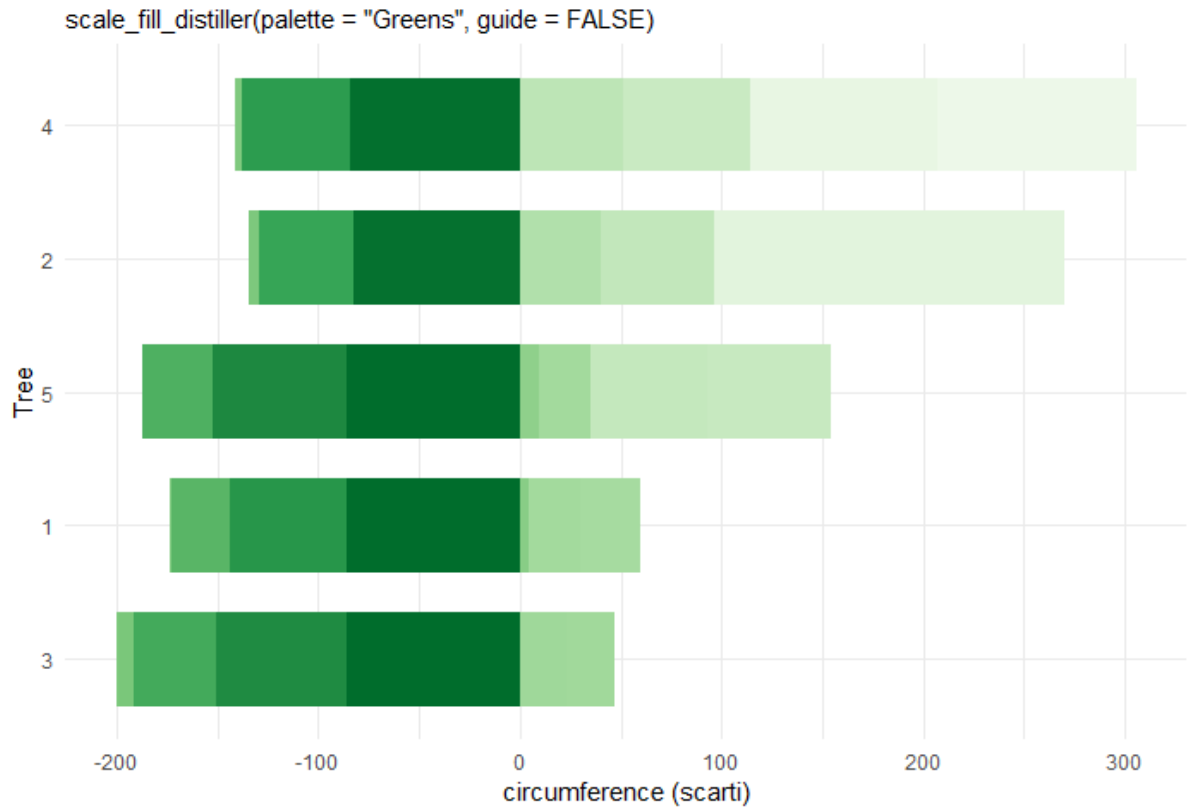
Fra tutte le funzioni indicate in tabella, queste sono piuttosto semplici, molto versatili, oltreché facili da ricordare. In realtà, infatti, anche se sono quattro, si tratta di ricordare *brewer* per i valori discreti e *distiller* per i valori continui.

## Valori continui

`scale_color_distiller()` e `scale_fill_distiller()`:

```
# distiller
```

```
Orange %>%
  mutate("scarti" = circumference - mean(circumference)) %>%
  ggplot(aes(x = Tree, y = scarti)) +
  geom_col(width = 0.7, aes(fill = scarti)) +
  ylab("circumference (scarti)") +
  theme_minimal() +
  coord_flip() +
  scale_fill_distiller(palette = "Greens",
                       guide = FALSE)       # senza legenda
```




Vedi anche grafici a barre, figura 6

## Valori discreti

```
# brewer
SLID %>%
  ggplot(aes(education, wages)) +
  # valore discreto da rappresentare con i colori
  geom_point(aes(col = sex)) +
  scale_color_brewer(palette = "Set1")
```

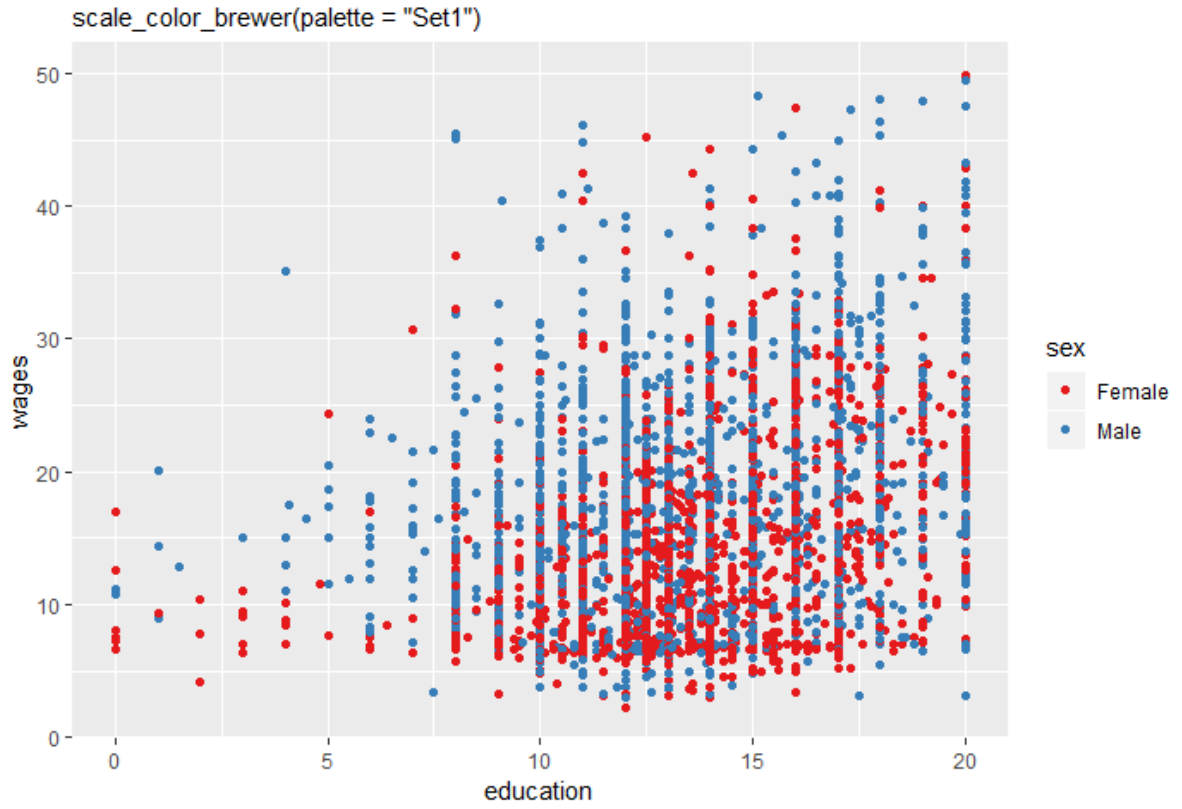
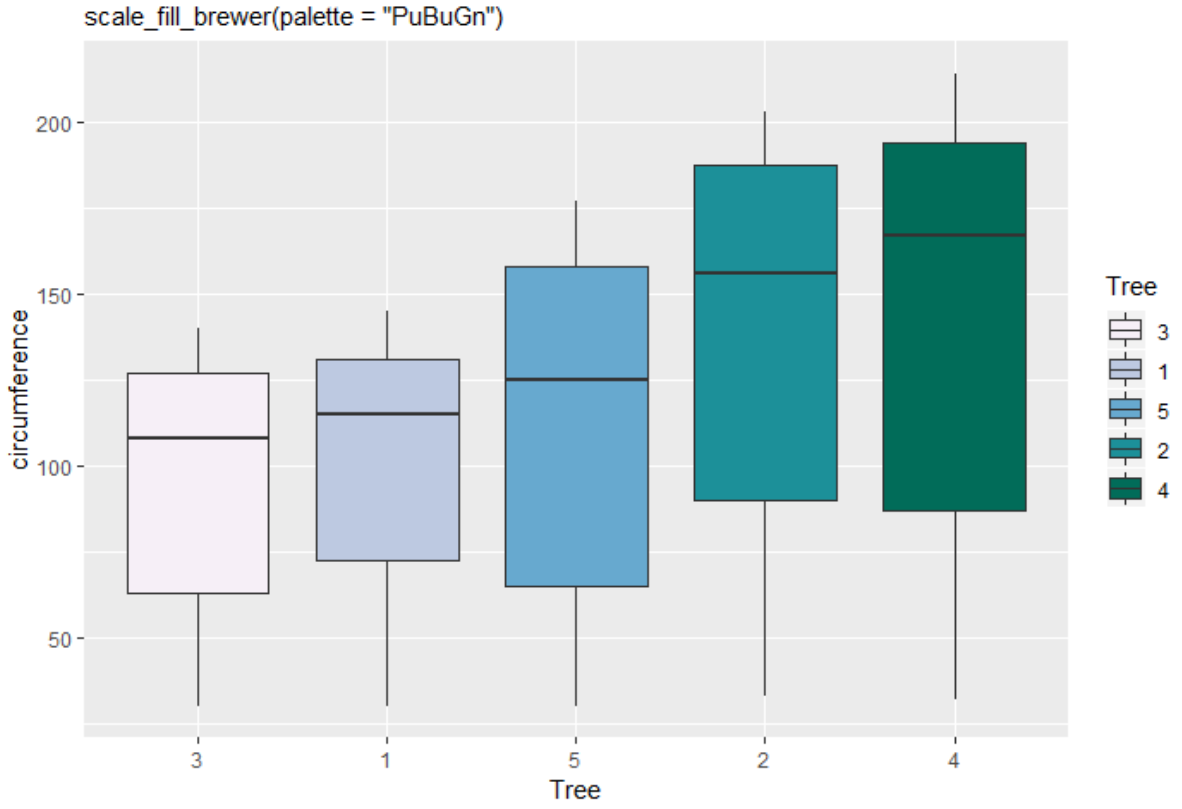



Fig. 2: brewer(palette = “Set1”)

```
Orange %>%
  ggplot(aes(Tree, circumference)) +
  geom_boxplot(aes(fill = Tree)) +
  scale_fill_brewer(palette = "PuBuGn")
```



# Toni di grigio

Particolarmente utile è rappresentare i grafici in bianco e nero: spesso tesi, report e pubblicazioni scientifiche non prevedono l'uso del colore.

## Valori continui

A tale scopo, per i caratteri continui, possiamo usare:

- `scale_color_distiller(palette = “Greys”)`, o `scale_fill_distiller(palette = “Greys”)`

e magari scegliere il tema in bianco e nero con `theme_bw()`

```
cars %>%
  ggplot(aes(speed, dist, col = speed)) +
  geom_point() +
  scale_color_distiller(palette = "Greys") +
  theme_bw()
```

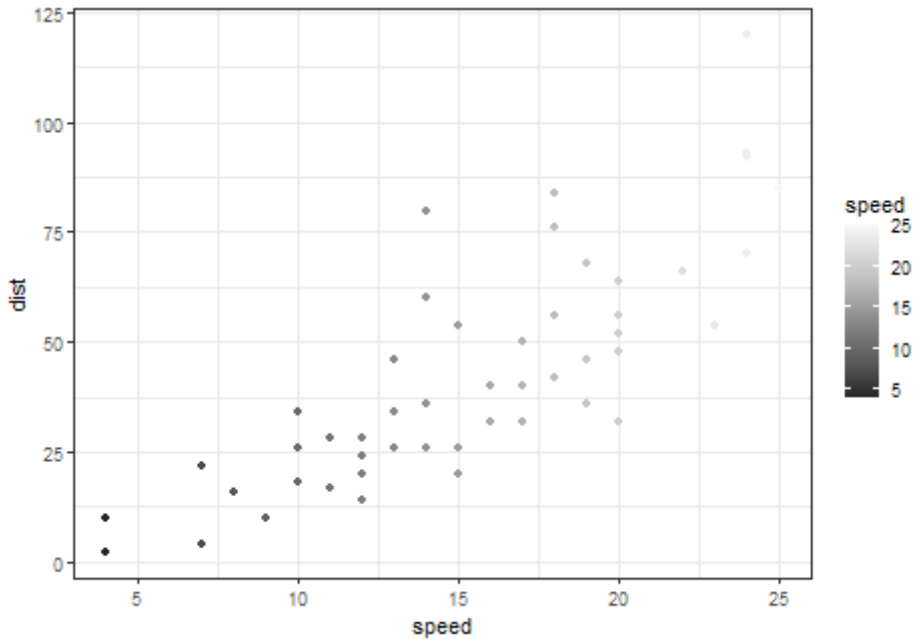



Fig. 3: Toni di grigio, carattere continuo

```
mtcars %>%
  ggplot(aes(wt, mpg)) +
  geom_point(aes(col = cyl), size = 2) +
  scale_color_distiller(palette = "Greys",
                        # cambiare la direzione dei colori
                        direction = 1) +
  labs(x = "Peso", y = "Miglia/Gallone", col = "Cilindrata",
```

```
       caption = "Dataset: mtcars") +
  theme_bw()
```

## Valori discreti

Per i caratteri discreti è possibile usare:

- `scale_color_brewer(palette = "Greys")`, o `scale_fill_brewer(palette = "Greys")`
- `scale_color_grey()` / `scale_fill_grey()`

```
Orange %>%
  ggplot(aes(x = Tree, y = circumference,
             fill = Tree)) +
  geom_boxplot(alpha = 0.8) +
  labs(fill = NULL) +
  scale_fill_grey() +
  theme_bw()
```

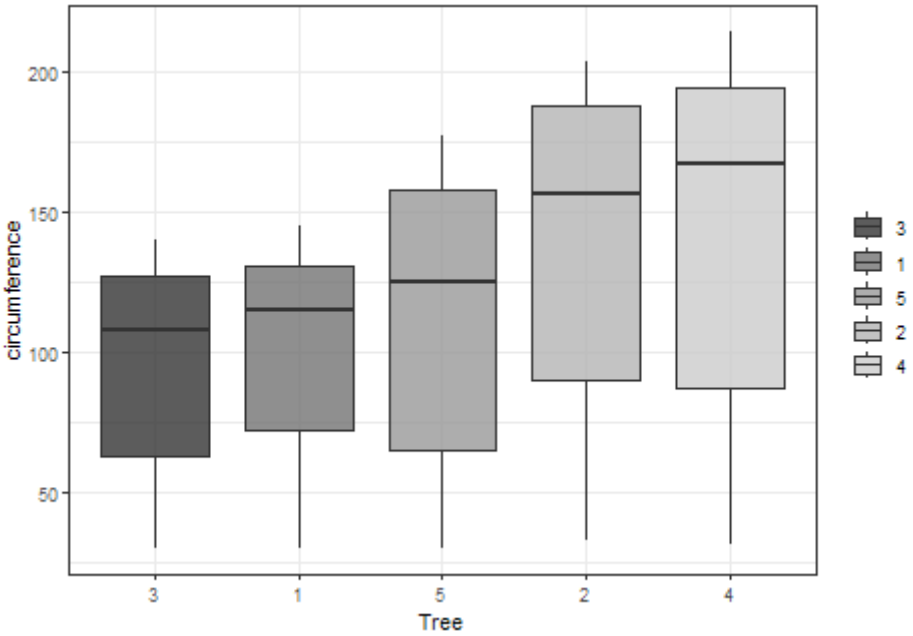



Fig. 4: Toni di grigio, variabile ordinata

oppure

```
Orange %>%
  ggplot(aes(x = Tree, y = circumference,
             fill = Tree)) +
  geom_boxplot(alpha = 0.8) +
  labs(fill = NULL) +
  scale_fill_brewer(palette = "Greys") +
  theme_bw()
```

# Definire i colori manualmente

Per i valori continui, le funzioni da usare per definire “a mano” i colori per valori continui sono quelle della famiglia `scale_color_gradient()` / `scale_fill_gradient()`, che collegano le intensità dei valori e intensità di uno o più colori.

- **Gradiente a due colori:**

```
scale_color_gradient(low = "orange", high = "darkblue")
```

- **Gradiente a tre colori, con valore intermedio:**

```
scale_color_gradient2(low = "orange", mid = "white", high =
"darkblue")
```

- **Scala di grigi, scegliendo la gamma di grigi**

```
scale_color_gradient(low = "grey80", high = "grey10")
```

Per i valori discreti, useremo `scale_fill_manual()`:

- **Indicare un colore o un vettore di colori**

```
scale_fill_manual(values = c("red", "blue", "green"))
```

In tutti i casi, i colori potranno essere indicati

- **ricorrendo alla palette in uso:**

```
scale_color_gradient(low = 1, high = 5)
```

```
scale_fill_manual(values = palette()[c(4,3,5)])
```

## Valori continui: gradient

La funzione è `scale_color_gradient()` / `scale_fill_gradient()`. Crea una **palette sequenziale** (a gradiente), e richiede indicare i colori per i valori `low` e `high` della variabile mappata:

```
# gradient
mtcars %>%
  ggplot(aes(wt, mpg)) +
  geom_point(aes(col = hp), size = 2) +
  scale_color_gradient(low = "lightgrey", high = "purple") +
  theme_minimal()
```

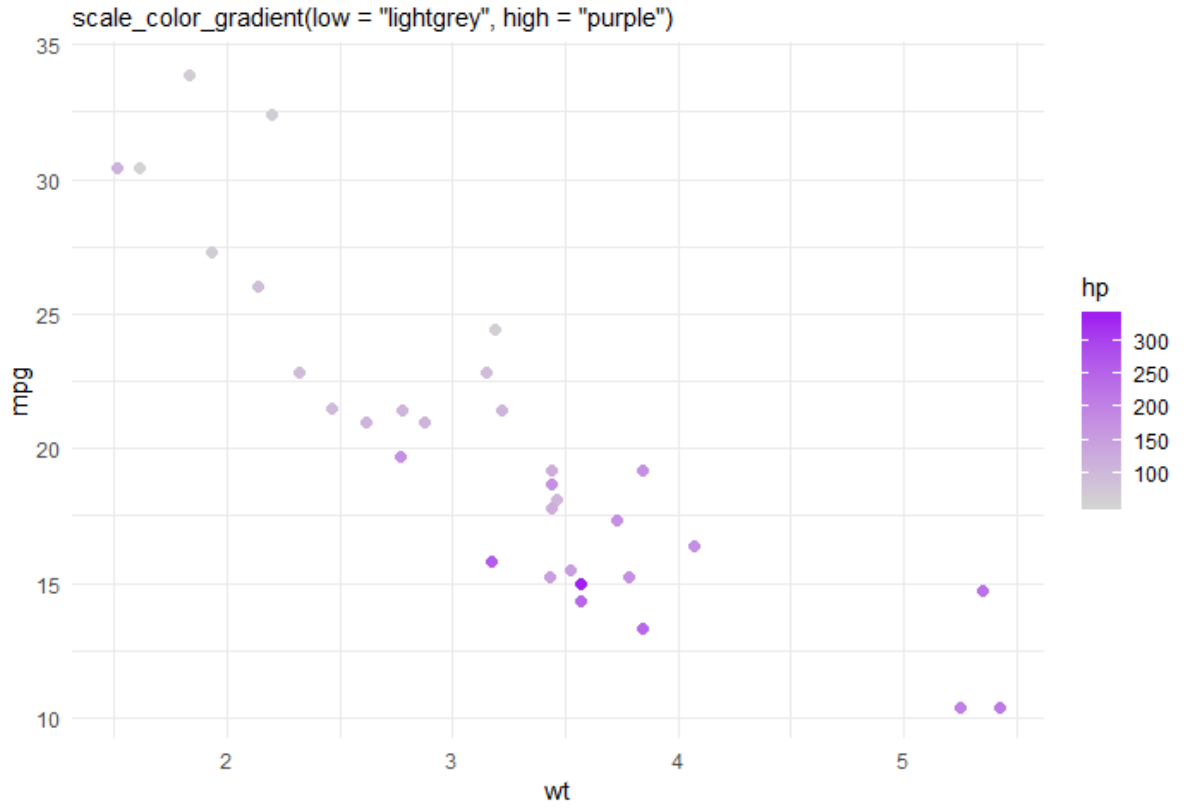



Fig. 5: Gradient: paletta sequenziale

Si possono indicare anche i colori della palette in uso:

```
scale_color_gradient(low = 1, high = 5)
```

Di conseguenza, è possibile cambiare la palette:

```
palette("Accent")
mtcars %>%
  ggplot(aes(wt, mpg)) +
  geom_point(aes(col = hp), size = 2) +
  scale_color_gradient(low = palette()[3], high = palette()[5]) +
  theme_minimal()
palette("default")
```

## Valori continui: gradient2

La funzione `scale_color_gradient2()` / `scale_fill_gradient2()` consente di creare una **palette divergente**, e richiede di indicare i colori per i valori `low`, `high` e anche `mid` della variabile mappata. Di default il valore per `mid` è 0; nell'esempio che segue ho indicato la media, e ho tradotto il nome delle variabili, anche cambiando il titolo della legenda:

```
# gradient 2
mtcars %>%
  ggplot(aes(wt, mpg)) +
```

```
  geom_point(aes(col = hp), size = 2) +
  labs(x = "peso", y = "miglia") +
  scale_color_gradient2('cavalli',      # titolo della legenda
                        low = 'darkgreen', mid = 'yellow', high =
'red',
                        midpoint = mean(mtcars$hp))
```

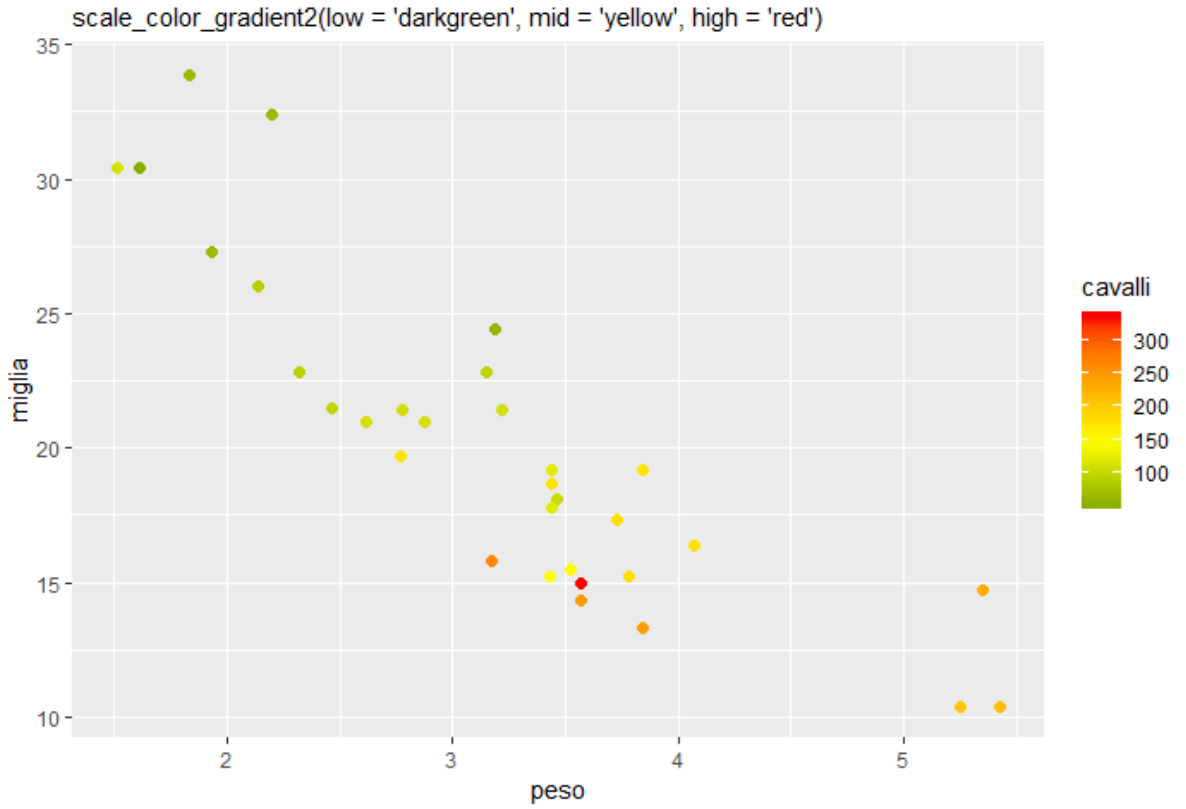



Fig. 6: Gradient2: palette divergente

## Valori discreti: manual

Abbastanza semplice, per i valori discreti, è indicare i colori con `scale_fill_manual` (o `scale_fill_manual()`)

```
# manual
SLID %>%
  ggplot(aes(sex)) +
  geom_bar(aes(fill = sex), width = 0.5) +
  scale_fill_manual(values=c("magenta", "blue"),
                    guide=FALSE)
```

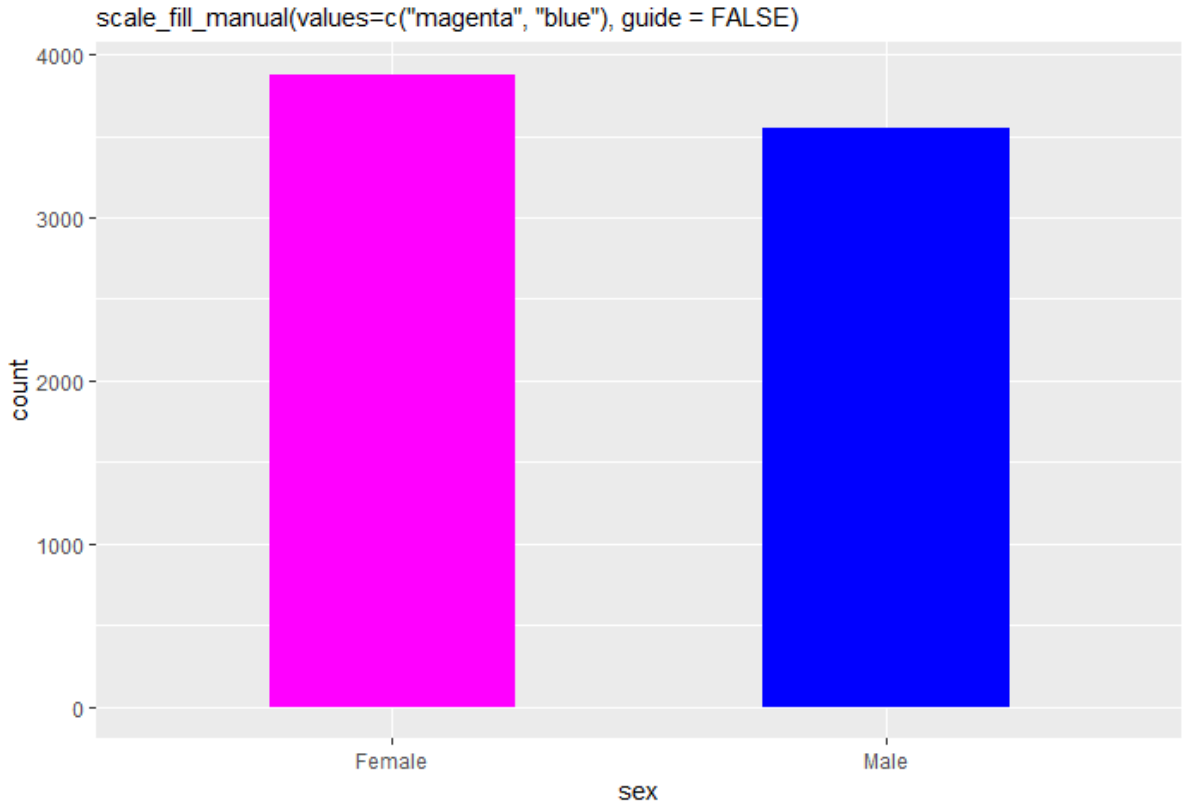
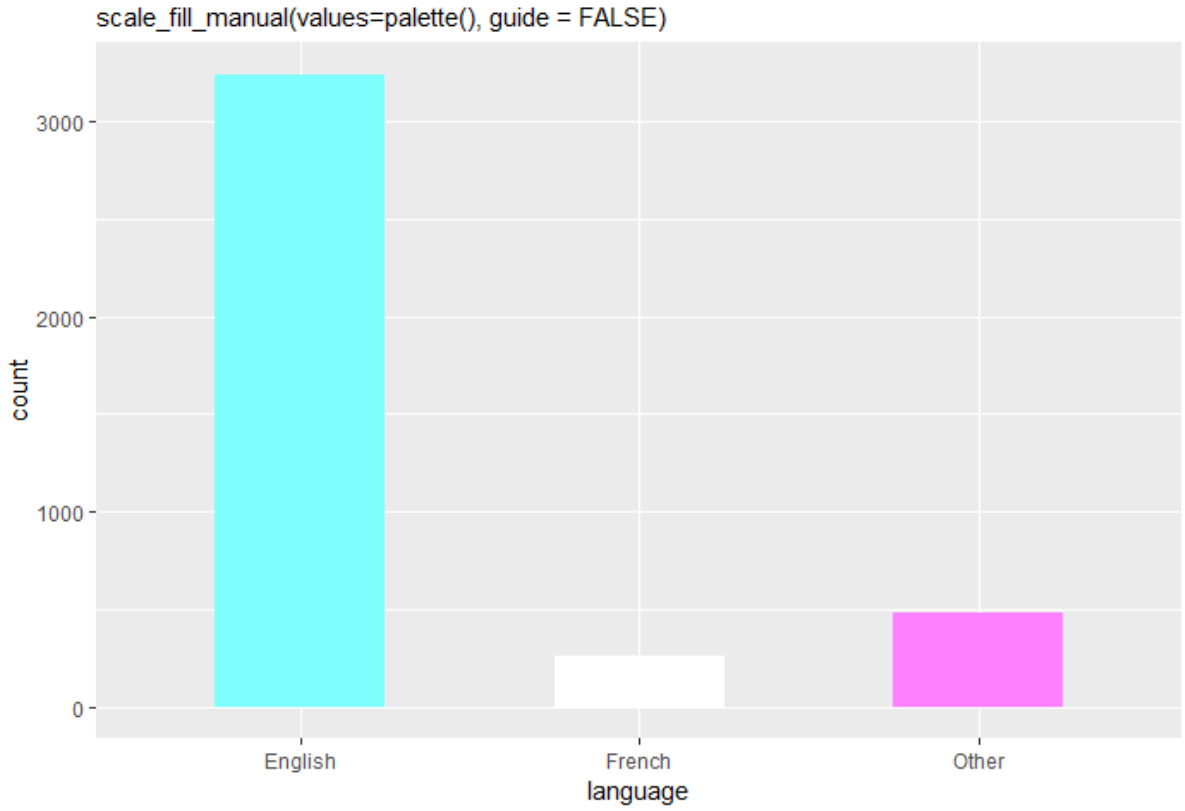



Fig. 7: manual

In questo modo, è anche molto semplice cambiare la palette (vedi Le palette).

```
palette(cm.colors(3))
SLID %>%
  na.omit(language) %>%
  ggplot(aes(language)) +
  geom_bar(aes(fill = language), width = 0.5) +
  scale_fill_manual(values=palette(),
                    guide=FALSE)

palette("default")
```




oppure

```
palette(cm.colors(3))
SLID %>%
  na.omit(language) %>%
  ggplot(aes(language)) +
  geom_bar(aes(fill = language),
           width = 0.5, col = "black") +
  scale_fill_manual(values=palette()[c(3,1,2)],
                    guide=FALSE)
palette("default")
```

## Valori discreti: hue

La funzione `scale_color_hue()` non è molto semplice da usare, in quanto fa riferimento alla ruota dei colori. Con l'argomento `h.start`, si indica il punto di inizio (in gradi). È inoltre possibile specificare come dividere i colori, la luminosità, la saturazione ecc.

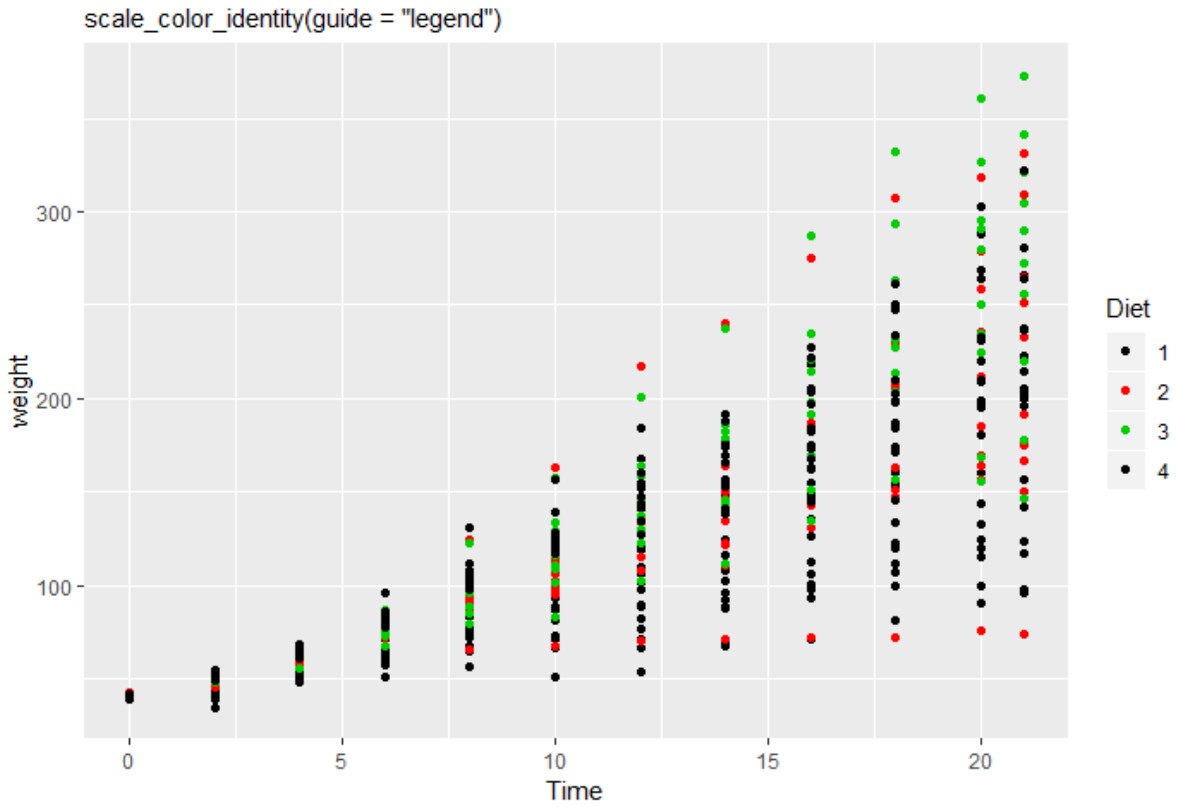
```
# hue
SLID %>%
  ggplot(aes(education, wages)) +
  geom_point(aes(col = sex)) +
  scale_color_hue(h.start = 90)       # cambio il punto sulla color
wheel
```

# Identity

Se la variabile mappata (o una diversa) è codificata in modo compatibile con la notazione dei colori (numeri o nomi dei colori), è possibile utilizzare `scale_color_identity()` o `scale_fill_identity()`.

La legenda va aggiunta, perché di default non viene prodotta.

```
# identity
ChickWeight %>%
  ggplot(aes(Time, weight)) +
  geom_point(aes(col = Diet)) +
  scale_color_identity(guide = "legend")      # aggiungo la legenda
```

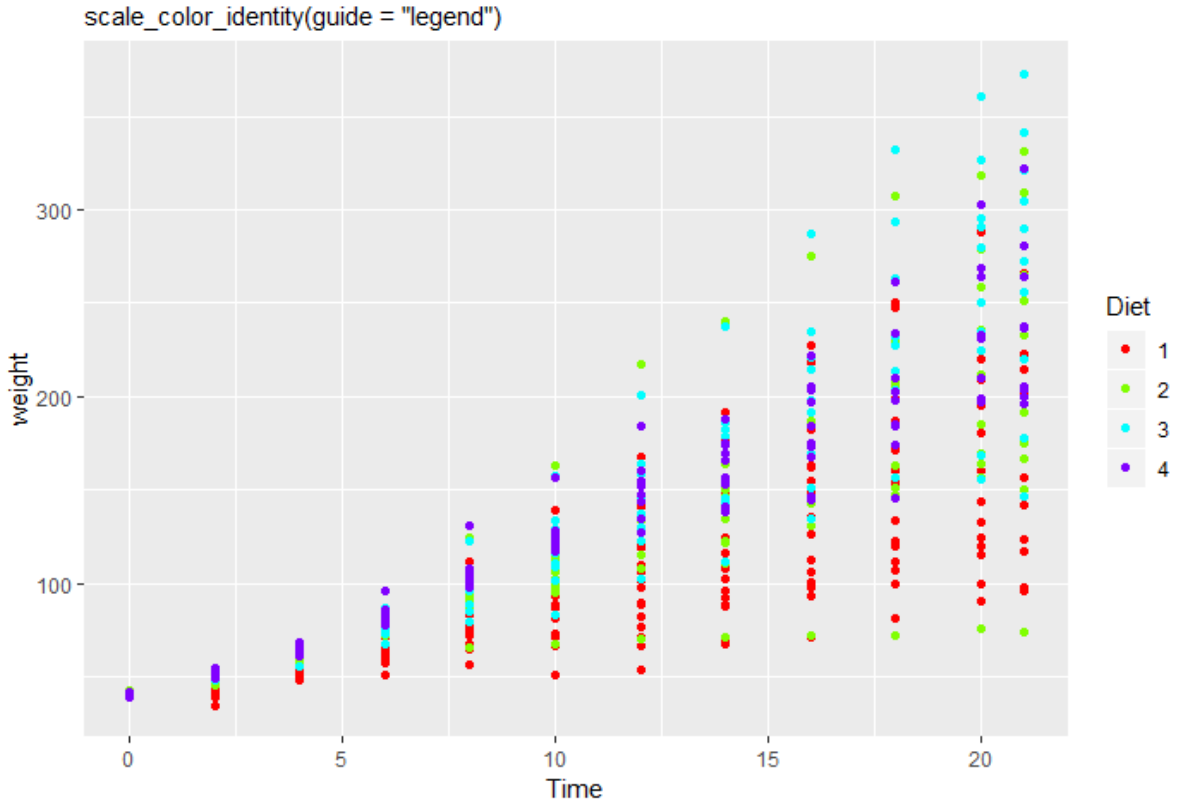



Anche in questo caso, possiamo cambiare la palette:

```
palette(rainbow(4))
ChickWeight %>%
  ggplot(aes(Time, weight)) +
  geom_point(aes(col = Diet)) +
  scale_color_identity(guide = "legend")

# torno alla palette di default
```

```
palette("default")
```




# Script di esempio

E' possibile scaricare ed eseguire lo script dell'esempio:

ggplot_base.R

```
# Pacchetti
library(tidyverse)
library(carData)

# distiller
Orange %>%
  mutate("scarti" = circumference - mean(circumference)) %>%
  ggplot(aes(x = Tree, y = scarti)) +
  geom_col(width = 0.7, aes(fill = scarti)) +
  ylab("circumference (scarti)") +
  theme_minimal() +
  coord_flip() +
  scale_fill_distiller(palette = "Greens",
                       guide = FALSE)       # senza legenda

# brewer
```

```
SLID %>%
  ggplot(aes(education, wages)) +
  # valore discreto da rappresentare con i colori
  geom_point(aes(col = sex)) +
  scale_color_brewer(palette = "Set1")

Orange %>%
  ggplot(aes(Tree, circumference)) +
  geom_boxplot(aes(fill = Tree)) +
  scale_fill_brewer(palette = "PuBuGn")

# toni di grigio
cars %>%
  ggplot(aes(speed, dist, col = speed)) +
  geom_point() +
  scale_color_distiller(palette = "Greys") +
  theme_bw()

Orange %>%
  ggplot(aes(x = Tree, y = circumference,
             fill = Tree)) +
  geom_boxplot(alpha = 0.8) +
  labs(fill = NULL) +
  scale_fill_grey() +
  theme_bw()

# gradient
mtcars %>%
  ggplot(aes(wt, mpg)) +
  geom_point(aes(col = hp), size = 2) +
  scale_color_gradient(low = "lightgrey", high = "purple") +
  theme_minimal()

# gradient2
mtcars %>%
  ggplot(aes(wt, mpg)) +
  geom_point(aes(col = hp), size = 2) +
  labs(x = "peso", y = "miglia") +
  scale_color_gradient2('cavalli',      # titolo della legenda
                        low = 'darkgreen', mid = 'yellow', high
= 'red',
                        midpoint = mean(mtcars$hp))

# manual
SLID %>%
  ggplot(aes(sex)) +
  geom_bar(aes(fill = sex), width = 0.5) +
```

```
  scale_fill_manual(values=c("magenta", "blue"),
                    guide=FALSE)


# hue
SLID %>%
  ggplot(aes(education, wages)) +
  geom_point(aes(col = sex)) +
  scale_color_hue(h.start = 90)       # cambio il punto sulla
colorwheel

# identity
ChickWeight %>%
  ggplot(aes(Time, weight)) +
  geom_point(aes(col = Diet)) +
  scale_color_identity(guide = "legend")      # aggiungo la
legenda
```

Grafici, ggplot2, Tidyverse

From:
https://www.agnesevardanega.eu/wiki/ - **Ricerca Sociale con R**

Permanent link:
**https://www.agnesevardanega.eu/wiki/r/ggplot2/gestione_colori**

Last update: **06/09/2025 23:36**