Kaggle Steam Datasets

Nicolò Rossi

xx/yy/20zz

Introduzione

In questo documento viene mostrata l'analisi di dati estratti dalla piattaforma per videogiochi "Steam". I dataset utilizzati sono disponibili pubblicamente e sono stati ottenuti dal sito kaggle

Che cos'è Steam

Steam la più grande piattafroma per giocare, pubblicare ed acquistare giochi per computer ..... scriverò questa parte un altro giorno

Dataset utilizzati

Sono considerati i seguenti due dataset:

- Steam Store Games (Clean dataset)
- Steam Video Games

contenenti rispettivamente le informazioni relative ai comportamenti di gioco e di acquisto di 200.000 videogiocatori e le quelle sui giochi disponibili sulla piattaforma Steam. Nel corso di questa relazione per semplicità citeremo il primo chiamandolo "dataset del 2019" oppure "dei giochi" ed il secondo usando i nomi "200k" o "dataset dei giocatori".

Dati "grezzi"

In questa sezione verranno presntati i dataset utilizzati per l'analisi dati, così come essi sono reperibili dal Web.

Dati sui giocatori

Per prima cosa consideriamo il dataset riguardante 200.000 operazioni compiute dai un campione di giocatori. Il dataset è stato caricato su Kaggle tre anni fa, ma avremo modo di mostrare che sicuramente i dati sono stati acquisiti precedentemente e prenderemo in considerazione il problema della sua "datazione".

Questo è inizialmente strutturato come segue:

```
# lettura
players.data <- read.csv(
    "./data/steam-200k.csv",
    header=FALSE)[, -5]
colnames(players.data) <- c("player", "name", "activity", "time")
# rimozione punteggiatura
```

```
players.data <- clean.text(players.data, name, "[-.:®'’]")
players.data %>% head(10) %>% kable()
```

player	name	activity	time
151603712	the elder scrolls v skyrim	purchase	1.0
151603712	the elder scrolls v skyrim	play	273.0
151603712	fallout 4	purchase	1.0
151603712	fallout 4	play	87.0
151603712	spore	purchase	1.0
151603712	spore	play	14.9
151603712	fallout new vegas	purchase	1.0
151603712	fallout new vegas	play	12.1
151603712	left 4 dead 2	purchase	1.0
151603712	left 4 dead 2	play	8.9

Il tempo viene espresso in ore di gioco ed ha significato solo quando associato all'attività "play".

I seguenti semplici conteggi ci consentono di comprendere la reale dimensione del campionamento effettuato per realizzare questa collezione di dati:

Numero di giocatori considerati:

```
number.of.classes(players.data, player)
```

```
## [1] 12393
```

Numero di giochi giocati:

```
number.of.classes(players.data, name)
```

```
## [1] 5154
```

Attività di acquisto:

```
count.selected.lines(players.data, activity == "purchase" )
```

```
## [1] 129511
```

Attività di gioco:

```
count.selected.lines(players.data, activity == "play" )
```

```
## [1] 70489
```

Questa ripartizione non equa delle attività di gioco e acquisto non è semplicemente dovuta al caso in quanto è invece particolarmente frequente acquistare giochi e non giocarli. Questo fenomeno viene detto "problema del backlog" e verrà affrontato con più dettaglio in seguito.

Informazioni sui giochi

Il dataset con le informazioni aggiuntive sugli specifici giochi è più ricco e completo rispetto a quello precedente, che invece si limita a un campionamento su un relativamente piccolo numero di giocatori. In esso sono racchiuse tutte le informazioni riguardanti i giochi pubblicati sulla piattaforma Steam fino a Maggio 2019 e ad ogni gioco vengono associati i seguenti attributi:

- **appid**: identificativo unico del gioco, non essendo presente nell'altro dataset non ci è molto utile
- **name**: nome del gioco

- **release_date**: data di uscita
- **english**: se supporta la lingua inglese
- **developer**: casa sviluppatrice (a volte possono essere più di una, in quel caso le consideriamo come un unico team di sviluppo separato)
- **publisher**: chi ha pubblicato il gioco
- **platforms**: piattaforme su cui il gioco è disponibile (Windows, Mac e Linux)
- **required_age**: Indica l'età consigliata per il gioco (se pubblicata)
- **categories**: Lista delle categorie (indicate da Steam) che descrivono alcune caratteristiche di gioco
- **genres**: Lista che descrive la tipologia di gioco
- **steamspy_tags**: Ulteriori tag, più dettagliati
- **achievements**: Indica se il gioco supporta il sistema a obiettivi offerto da Steam (e con quanti)
- **positive_ratings**: Rating positivi per il gioco inviati dagli utenti
- **negative_ratings**: Rating negativi per il gioco inviati dagli utanti
- **average_playtime**: tempo medio di gioco
- **median_playtime**: tempo mediano di gioco
- **owners**: indica un intervallo in cui ricade il numero di persone che posseggono il gioco
- **price**: Prezzo di listino del prodotto

Il dataset offre quindi molte informazioni ma non tutti gli attributi risultano egualmente affidabili; in particolare si sono osservate delle discrepanze molto significative nel calcolo dei tempi di gioco, per cui si è deciso di escludere questi dati dall'analisi e di utilizzare, quando possibile, le informazioni presenti nel dataset dei giocatori, considerato in precedenza.

> Il sistema degli *achievements* è stato introdotto nel 2005 da Microsoft per la sua piattaforma Xbox Live e si basa sul proporre sfide con diversa difficoltà che siano comparabili fra i diversi giochi presenti sulla stessa piattaforma. Lo scopo è quello di aumentare la longevità dei giochi fornendo sfide che premino il giocatore quando completate e creando leaderboards virtuali per aumentare la competizione fra i diversi giocatori. Dato il successo di questo sistema, in breve tempo diverse compagnie come Sony, EA, Valve e Ubisoft, hanno introdotto delle loro versioni degli achievement sulle loro piattaforme con modalità molto simili a quelle proposte da Microsoft. Il concetto di "Gamification" che viene ora proposto in molti ambiti estenri al modo dei videogiochi, come l'e-learning, molto spesso presenta elementi analoghi a quelli degli achievement.

Questa è, ad esempio, una porzione di questo dataset:

```
games.data <- read.csv(
    "./data/steam.csv",
    colClasses = c("numeric", "character", "character", "factor", "factor", "factor",
                   "character", "factor", "character", "character", "character", "factor",
                   "integer", "integer", "numeric", "numeric", "factor", "numeric"))

# semplifico i nomi per poter effettuare il join con l'altro dataset
games.data <- clean.text(games.data, name, "[-.:®'’]")
games.data %>% head(10)
```

```
##    appid                         name release_date english        developer
## 1     10               counterstrike   2000-11-01       1            Valve
## 2     20        team fortress classic   1999-04-01       1            Valve
## 3     30               day of defeat   2003-05-01       1            Valve
## 4     40          deathmatch classic   2001-06-01       1            Valve
## 5     50      halflife opposing force   1999-11-01       1 Gearbox Software
## 6     60                     ricochet   2000-11-01       1            Valve
## 7     70                     halflife   1998-11-08       1            Valve
## 8     80 counterstrike condition zero   2004-03-01       1            Valve
## 9    130          halflife blue shift   2001-06-01       1 Gearbox Software
## 10   220                   halflife 2   2004-11-16       1            Valve
```

```
##    publisher         platforms required_age
## 1      Valve windows;mac;linux            0
## 2      Valve windows;mac;linux            0
## 3      Valve windows;mac;linux            0
## 4      Valve windows;mac;linux            0
## 5      Valve windows;mac;linux            0
## 6      Valve windows;mac;linux            0
## 7      Valve windows;mac;linux            0
## 8      Valve windows;mac;linux            0
## 9      Valve windows;mac;linux            0
## 10     Valve windows;mac;linux            0
##
## 1                                                    Multi-player;Online Multi-Player;Local Multi
## 2                                                    Multi-player;Online Multi-Player;Local Multi
## 3                                                                                          Multi
## 4                                                    Multi-player;Online Multi-Player;Local Multi
## 5                                                                          Single-player;Multi
## 6                                                                Multi-player;Online Multi
## 7                                              Single-player;Multi-player;Online Multi-Player;Stea
## 8                                                                          Single-player;Multi
## 9
## 10 Single-player;Steam Achievements;Steam Trading Cards;Captions available;Partial Controller Support
##    genres               steamspy_tags achievements positive_ratings
## 1  Action       Action;FPS;Multiplayer            0           124534
## 2  Action       Action;FPS;Multiplayer            0             3318
## 3  Action FPS;World War II;Multiplayer            0             3416
## 4  Action       Action;FPS;Multiplayer            0             1273
## 5  Action            FPS;Action;Sci-fi            0             5250
## 6  Action       Action;FPS;Multiplayer            0             2758
## 7  Action           FPS;Classic;Action            0            27755
## 8  Action       Action;FPS;Multiplayer            0            12120
## 9  Action            FPS;Action;Sci-fi            0             3822
## 10 Action            FPS;Action;Sci-fi           33            67902
##    negative_ratings average_playtime median_playtime            owners price
## 1              3339            17612             317 10000000-20000000  7.19
## 2               633              277              62  5000000-10000000  3.99
## 3               398              187              34  5000000-10000000  3.99
## 4               267              258             184  5000000-10000000  3.99
## 5               288              624             415  5000000-10000000  3.99
## 6               684              175              10  5000000-10000000  3.99
## 7              1100             1300              83  5000000-10000000  7.19
## 8              1439              427              43 10000000-20000000  7.19
## 9               420              361             205  5000000-10000000  3.99
## 10             2419              691             402 10000000-20000000  7.19
```

Per un totale di:

```
nrow(games.data)
```

```
## [1] 27075
```

giochi diversi.

Rendere i dataset Tidy

Consideriamo dapprima il dataset dei giochi presenti su Steam. Per evitare un incremento sostanziale nella dimensione del dataset manteniamo temporaneamente i campi lista ma li trasformiamo in vere e proprie liste R (anzichè del testo spaziato da ";"). Questo ci consentirà di usare *dplyr* nelle fasi successive in modo molto semplice per poter operare sui suddetti campi.

```
# trasformo le liste separate da ; in liste R
games.data <- games.data %>% mutate(platforms = strsplit(platforms, ";") ) %>%
    mutate(categories = strsplit(categories, ";") ) %>%
    mutate(genres = strsplit(genres, ";") ) %>%
    mutate(steamspy_tags = strsplit(steamspy_tags, ";") )
```

Interpretiamo correttamente il campo con la data di uscita considerandolo come una data R.

```
# trasformo le date da testo a data
games.data <- games.data %>% mutate(release_date = as.Date(release_date) )
```

Per concludere separiamo il campo owners nel suo lower e upper bound, in modo da poterli usare per operazioni di filtering basate su numeri.

```
# divido le info sul numero di giocatori in lower and upper bounds
games.data <- games.data %>% separate(owners, into=c("owners_lwb","owners_upb"), sep="-") %>%
    mutate(owners_lwb = as.integer(owners_lwb)) %>%
    mutate(owners_upb = as.integer(owners_upb))

games.data %>% head(10)
```

```
##    appid                         name release_date english        developer
## 1     10               counterstrike   2000-11-01       1            Valve
## 2     20      team fortress classic   1999-04-01       1            Valve
## 3     30               day of defeat   2003-05-01       1            Valve
## 4     40          deathmatch classic   2001-06-01       1            Valve
## 5     50     halflife opposing force   1999-11-01       1 Gearbox Software
## 6     60                    ricochet   2000-11-01       1            Valve
## 7     70                    halflife   1998-11-08       1            Valve
## 8     80 counterstrike condition zero   2004-03-01       1            Valve
## 9    130          halflife blue shift   2001-06-01       1 Gearbox Software
## 10   220                  halflife 2   2004-11-16       1            Valve
##    publisher           platforms required_age
## 1      Valve windows, mac, linux            0
## 2      Valve windows, mac, linux            0
## 3      Valve windows, mac, linux            0
## 4      Valve windows, mac, linux            0
## 5      Valve windows, mac, linux            0
## 6      Valve windows, mac, linux            0
## 7      Valve windows, mac, linux            0
## 8      Valve windows, mac, linux            0
## 9      Valve windows, mac, linux            0
## 10     Valve windows, mac, linux            0
##
## 1                                                    Multi-player, Online Multi-Player, Local
## 2                                                    Multi-player, Online Multi-Player, Local
## 3
## 4                                                    Multi-player, Online Multi-Player, Local
## 5                                                                        Single-player,
```

```
## 6                                                                                                  Multi-player, Online
## 7                                                                  Single-player, Multi-player, Online Multi-Player
## 8                                                                                                      Single-player,
## 9
## 10 Single-player, Steam Achievements, Steam Trading Cards, Captions available, Partial Controller Su
##    genres                  steamspy_tags achievements positive_ratings
## 1  Action       Action, FPS, Multiplayer            0           124534
## 2  Action       Action, FPS, Multiplayer            0             3318
## 3  Action FPS, World War II, Multiplayer            0             3416
## 4  Action       Action, FPS, Multiplayer            0             1273
## 5  Action            FPS, Action, Sci-fi            0             5250
## 6  Action       Action, FPS, Multiplayer            0             2758
## 7  Action           FPS, Classic, Action            0            27755
## 8  Action       Action, FPS, Multiplayer            0            12120
## 9  Action            FPS, Action, Sci-fi            0             3822
## 10 Action            FPS, Action, Sci-fi           33            67902
##    negative_ratings average_playtime median_playtime owners_lwb owners_upb
## 1              3339            17612             317   10000000   20000000
## 2               633              277              62    5000000   10000000
## 3               398              187              34    5000000   10000000
## 4               267              258             184    5000000   10000000
## 5               288              624             415    5000000   10000000
## 6               684              175              10    5000000   10000000
## 7              1100             1300              83    5000000   10000000
## 8              1439              427              43   10000000   20000000
## 9               420              361             205    5000000   10000000
## 10             2419              691             402   10000000   20000000
##    price
## 1   7.19
## 2   3.99
## 3   3.99
## 4   3.99
## 5   3.99
## 6   3.99
## 7   7.19
## 8   7.19
## 9   3.99
## 10  7.19
```

Separazione delle informazioni su gioco e acquisto

Consideriamo ora il dataset dei giocatori, per renderelo tidy lo suddividiamo in due dataset separati, uno per le operazioni di acquisto dei giochi e uno per le attività di gioco:

```
players.play <- players.data %>% filter(activity == "play") %>% select(-activity)
players.buy  <- players.data %>% filter(activity == "purchase") %>% select(-activity, -time)
players.play %>% head(10)
```

```
##       player                        name  time
## 1  151603712 the elder scrolls v skyrim 273.0
## 2  151603712                  fallout 4  87.0
## 3  151603712                      spore  14.9
## 4  151603712          fallout new vegas  12.1
## 5  151603712              left 4 dead 2   8.9
```

```
## 6  151603712                  huniepop   8.5
## 7  151603712              path of exile   8.1
## 8  151603712                poly bridge   7.5
## 9  151603712                left 4 dead   3.3
## 10 151603712            team fortress 2   2.8
```

```
players.buy %>% head(10)
```

```
##       player                        name
## 1  151603712 the elder scrolls v skyrim
## 2  151603712                  fallout 4
## 3  151603712                      spore
## 4  151603712          fallout new vegas
## 5  151603712              left 4 dead 2
## 6  151603712                   huniepop
## 7  151603712              path of exile
## 8  151603712                poly bridge
## 9  151603712                left 4 dead
## 10 151603712            team fortress 2
```

Analisi esplorativa

In questa sezione risponderemo ad alcune semplici domande che possiamo possiamo porci sui dataset considerati, in modo di poterli comprendere meglio per le analisi successive.

Considerazioni sul tempo di gioco

In questa sottosezione consideriamo il dataset sulle attività di gioco dei giocatori, in particolare ci chiediamo quali siano i giochi più giocati nel campione analizzato e come questa sia distribuita:

Per prima cosa quindi ordiniamo i giochi per tempo assoluto e cumulativo di gioco:

```
games.mostplayed <- players.play %>%
    group_by(name) %>%
    summarise(totalTime = sum(time)) %>%
    arrange(desc(totalTime))
games.mostplayed %>% head(10)
```

```
## # A tibble: 10 x 2
##    name                                    totalTime
##    <chr>                                       <dbl>
##  1 dota 2                                    981685.
##  2 counterstrike global offensive            322772.
##  3 team fortress 2                           173673.
##  4 counterstrike                             134261.
##  5 sid meiers civilization v                  99821.
##  6 counterstrike source                       96076.
##  7 the elder scrolls v skyrim                 70889.
##  8 garrys mod                                 49725.
##  9 call of duty modern warfare 2 multiplayer  42010.
## 10 left 4 dead 2                              33597.
```

Si noti che i primi quattro prodotti più giocati sono realizzati da Valve, la compagnia responsabile della piattaforma Steam. Questo è piuttosto significativo e mostra come una piattaforma proprietaria consenta

poi di incrementare l'utilizzo dei propri giochi nonostante questi non siano gli unici offerti. E'da considerare inoltre che nel 2018 Steam era largamente la più grande piattaforma del mercato PC dei videogiochi, in grado di coinvolgere gran parte dell'utenza e degli sviluppatori. Solo verso la fine dell'anno, con l'apertura di un primo store generalista veramente concorrente, l'Epic Games Store, si è venuta a creare della concorrenza nel campo del publishing dei giochi PC, prima sostanzialmente monopolizzato da Valve in particolare per la sfera degli sviluppatori indipendenti. Se da un lato la disponibilità di diverse piattaforme sia un vantaggio per il consumatore e gli sviluppatori dal punto di vista economico, questo crea delle problematiche non indifferenti nella gestione degli, ormai estremamente diffusi, giochi multi-giocatore, nei quali possono crearsi situazioni di incompatibilità tra versioni offerte da piattaforme differenti dello stesso gioco, suddividendo così la "userbase" (ossia i giocatori) in insiemi separati e più piccoli. Questo fenomeno era classicamente invece limitato al mondo dei giochi su console proprio per la loro natura "chiusa", ossia completamente dettata dalla casa produttrice. In ogni caso gli sviluppatori stanno agendo da diversi anni per mitigare il problema e sono sempre di più i giochi che supportano il "cross-play" ossia la possibilità, per giocatori di piattaforme diverse, di giocare allo stesso gioco.

Osserviamo subito che molti giochi risultano quindi acquistati e non giocati:

```
# giochi totali
number.of.classes(players.data, name)
```

```
## [1] 5154
```

```
# giochi acquistati
number.of.classes(players.buy, name)
```

```
## [1] 5154
```

```
# giochi giocati
number.of.classes(players.play, name)
```

```
## [1] 3600
```

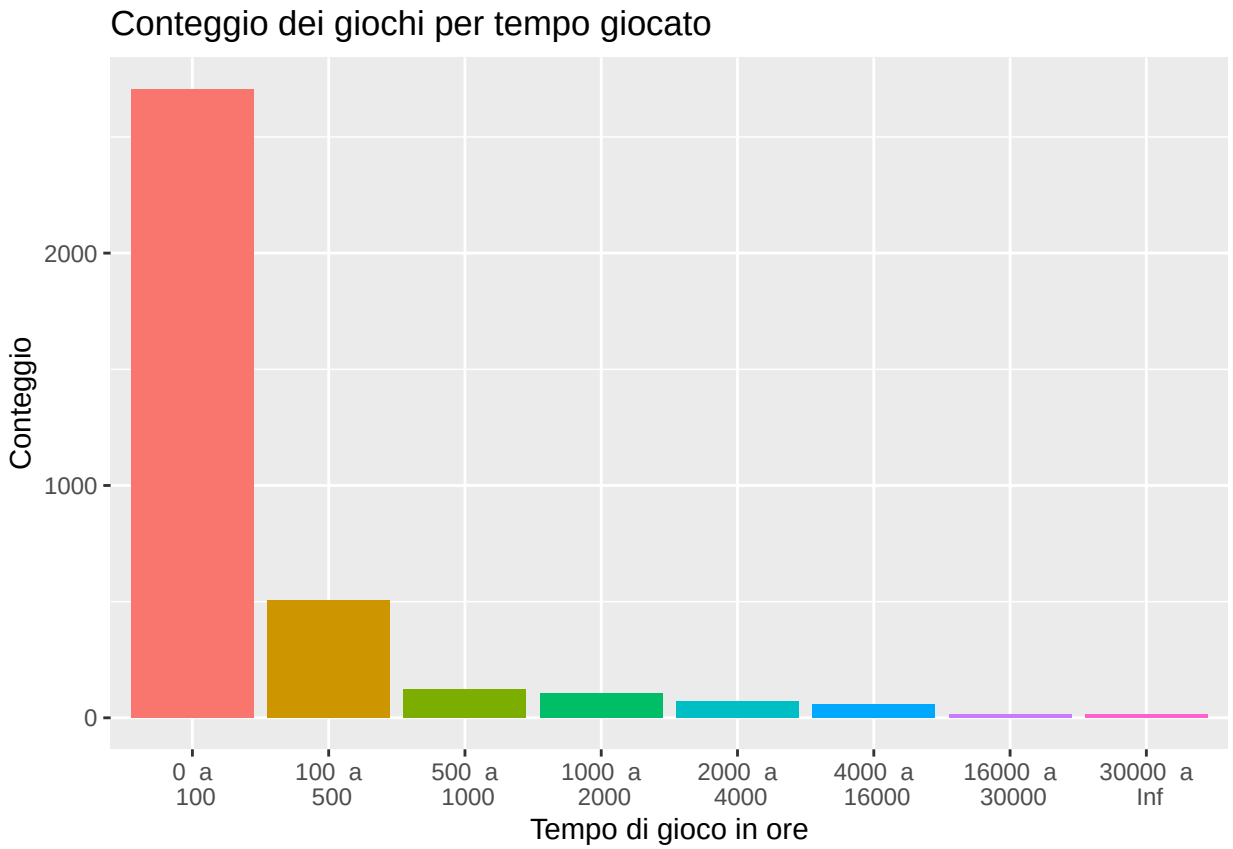
```
# differenza
number.of.classes(players.buy, name) - number.of.classes(players.play, name)
```

```
## [1] 1554
```

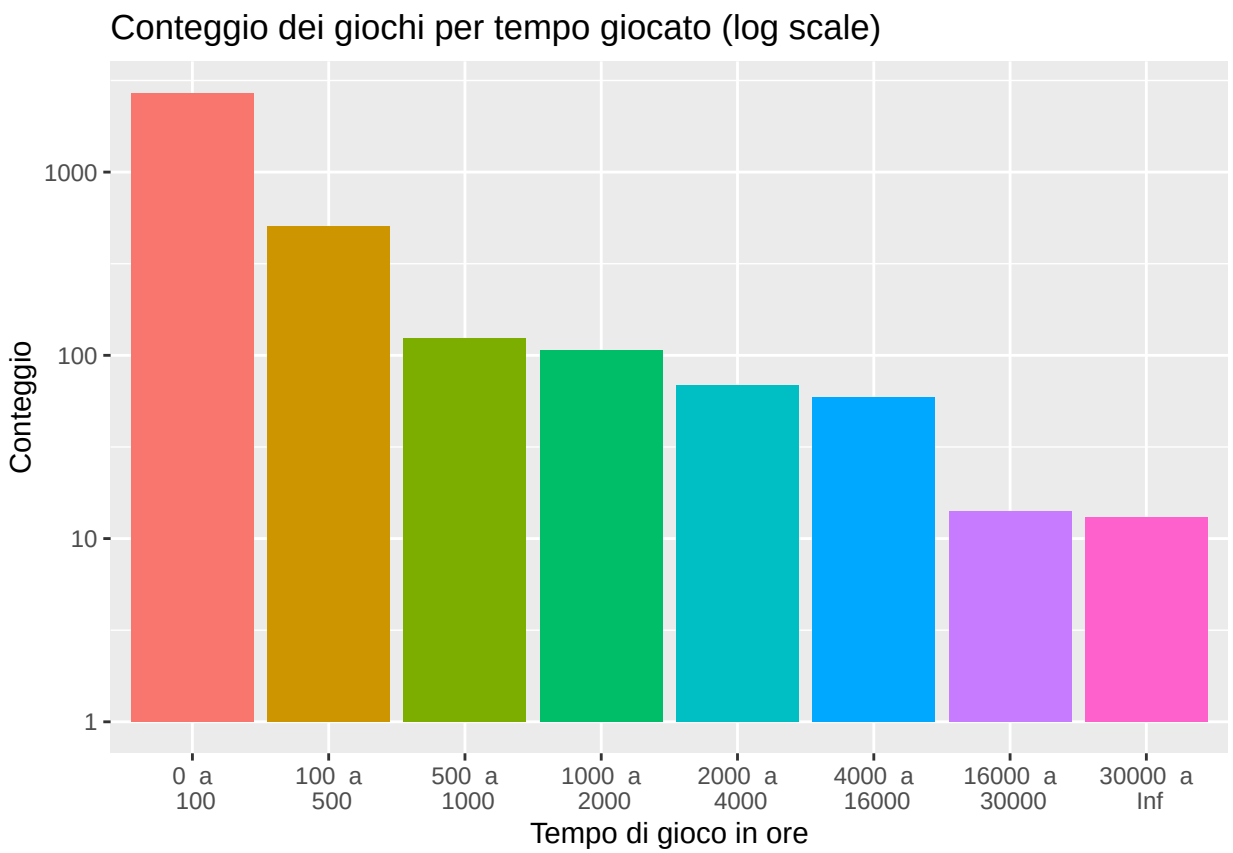
Questo ci permette di verificare che effettivamente il dataset è consistente per la regola "giocato → acquistato", come sarebbe atteso.

Consideriamo quindi la distribuzione dei tempi di gioco:

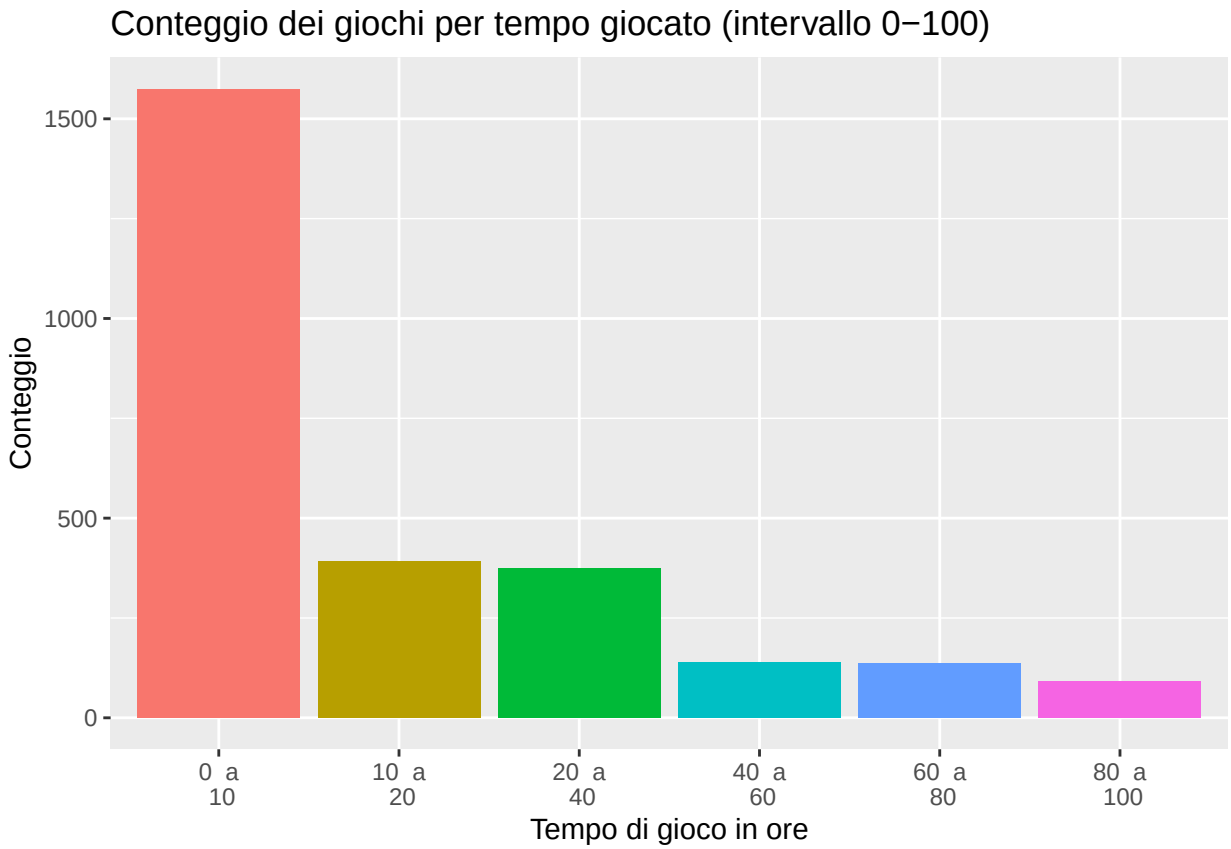
```
# distribuzione del tempo di gioco
temp <- games.mostplayed %>% factorise(totalTime, c(0,100,500,1000,2000,4000,16000,30000,Inf))
levels(temp$totalTime) = paste(c(0,100,500,1000,2000,4000,16000,30000), " a\n", c(100,500,1000,2000,400
ggplot(temp, legend=FALSE) +
  labs(title = "Conteggio dei giochi per tempo giocato") +
  ylab("Conteggio") +
  xlab("Tempo di gioco in ore") +
  geom_bar(aes(x=totalTime, fill=totalTime)) +
  theme(legend.position = "none")
```




```
# con scala logaritmica
ggplot(temp, legend=FALSE) +
  labs(title = "Conteggio dei giochi per tempo giocato (log scale)") +
  ylab("Conteggio") +
  xlab("Tempo di gioco in ore") +
  geom_bar(aes(x=totalTime, fill=totalTime)) +
  theme(legend.position = "none") +
  scale_y_log10()
```




```
# concentransosi sull'intervallo 0-100
temp <- games.mostplayed %>% filter(totalTime<100) %>% factorise(totalTime, c(0,10,20,40,60,80,100))
levels(temp$totalTime) = paste(c(0,10,20,40,60,80), " a\n", c(10,20,40,60,80,100) )
ggplot(temp, legend=FALSE) +
  labs(title = "Conteggio dei giochi per tempo giocato (intervallo 0-100)") +
  ylab("Conteggio") +
  xlab("Tempo di gioco in ore") +
  geom_bar(aes(x=totalTime, fill=totalTime)) +
  theme(legend.position = "none")
```



Viene naturale ora chiedersi se queste distribuzioni assolute si rispecchiano passando al tempo medio di gioco.

Numero di Giocatori

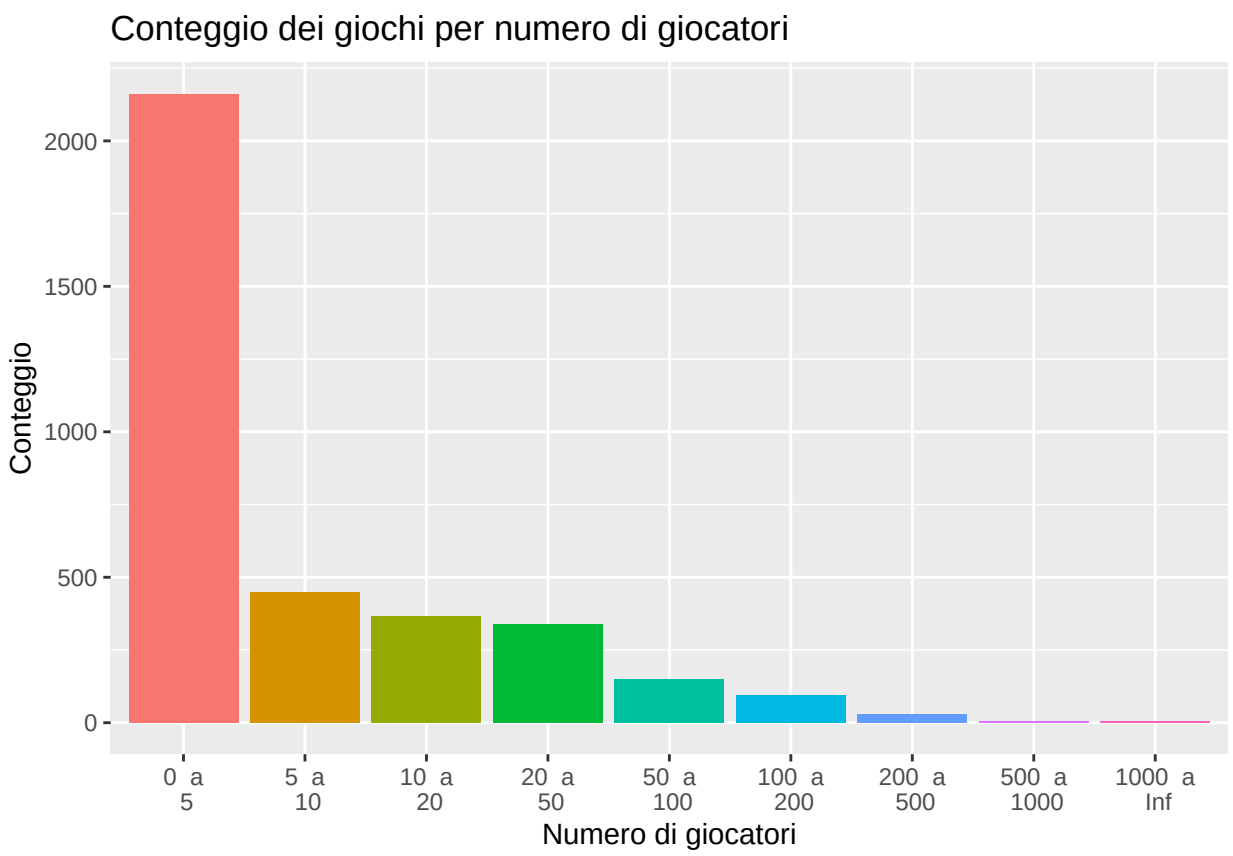
Calcoliamo quindi per prima cosa il numero di giocatori per ogni gioco e ne visualizziamo la distribuzione in modo similare al caso precedente:

```
games.nplayers <- players.play %>%
    group_by(name) %>%
    summarise(totalPlayers = n()) %>%
    arrange(desc(totalPlayers))
games.nplayers %>% head(10)
```

```
## # A tibble: 10 x 2
##    name                           totalPlayers
##    <chr>                                 <int>
##  1 dota 2                                 4841
##  2 team fortress 2                        2323
##  3 counterstrike global offensive         1377
##  4 unturned                               1069
##  5 left 4 dead 2                           801
##  6 counterstrike source                    715
##  7 the elder scrolls v skyrim              677
##  8 garrys mod                              666
##  9 counterstrike                           568
## 10 sid meiers civilization v               554
```

Visualizziamo quindi i risultati

```
# distribuzione del numero di giocatori
temp <- games.nplayers %>% factorise(totalPlayers, c(0,5,10,20,50,100,200,500,1000,Inf))
levels(temp$totalPlayers) = paste(c(0,5,10,20,50,100,200,500,1000), " a\n", c(5,10,20,50,100,200,500,100
ggplot(temp, legend=FALSE) +
  labs(title = "Conteggio dei giochi per numero di giocatori") +
  ylab("Conteggio") +
  xlab("Numero di giocatori") +
  geom_bar(aes(x=totalPlayers, fill=totalPlayers)) +
  theme(legend.position = "none")
```



Anche qui si nota la tendenza ad affermarsi di soli pochi giochi. Possiamo ora valutare il tempo medio di gioco.

Tempo medio di gioco

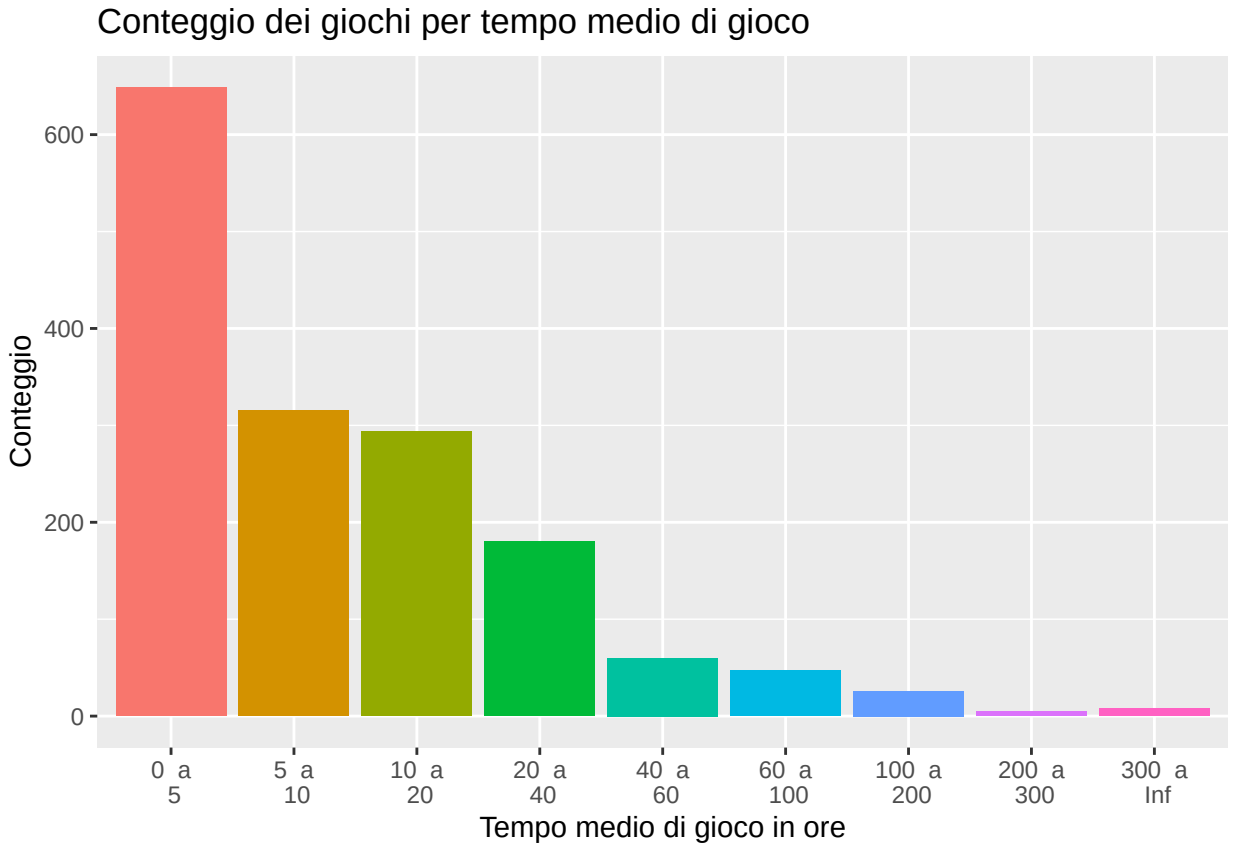
```
games.avgplaytime <- players.play %>%
    group_by(name) %>%
    summarise(avgPlayTime = mean(time), players = n() ) %>%
    arrange(desc(avgPlayTime))
games.avgplaytime %>% head(10)
```

```
## # A tibble: 10 x 3
##    name                            avgPlayTime players
##    <chr>                                 <dbl>   <int>
##  1 eastside hockey manager                1295        1
```

```
##  2 baldurs gate ii enhanced edition        475.       9
##  3 fifa manager 09                         411        1
##  4 perpetuum                               401.       4
##  5 football manager 2014                   392.      78
##  6 football manager 2012                   390.      79
##  7 football manager 2010                   375.      35
##  8 football manager 2011                   366.      31
##  9 freaking meatbags                       331        1
## 10 out of the park baseball 16             330.       2
```

Si può notare da questa lista che molti dei giochi con tempo medio più elevato sono giocati da pochi giocatori che si appassionano particolarmente a un gioco specifico. Per evitare di prendere in considerazione casi limite eccezionali (come "Eastside Hockey Manager"), filtriamo i risultati per accettare esclusivamente i giochi con almeno 5 giocatori.

```
# con più di cento giocatori
temp <- games.avgplaytime %>% filter(players >= 5) %>%
  factorise(avgPlayTime, c(0,5,10,20,40,60,100,200,300,Inf))
levels(temp$avgPlayTime) = paste(c(0,5,10,20,40,60,100,200,300), " a\n", c(5,10,20,40,60,100,200,300,In
ggplot(temp, legend=FALSE) +
  labs(title = "Conteggio dei giochi per tempo medio di gioco") +
  ylab("Conteggio") +
  xlab("Tempo medio di gioco in ore") +
  geom_bar(aes(x=avgPlayTime, fill=avgPlayTime)) +
  theme(legend.position = "none")
```

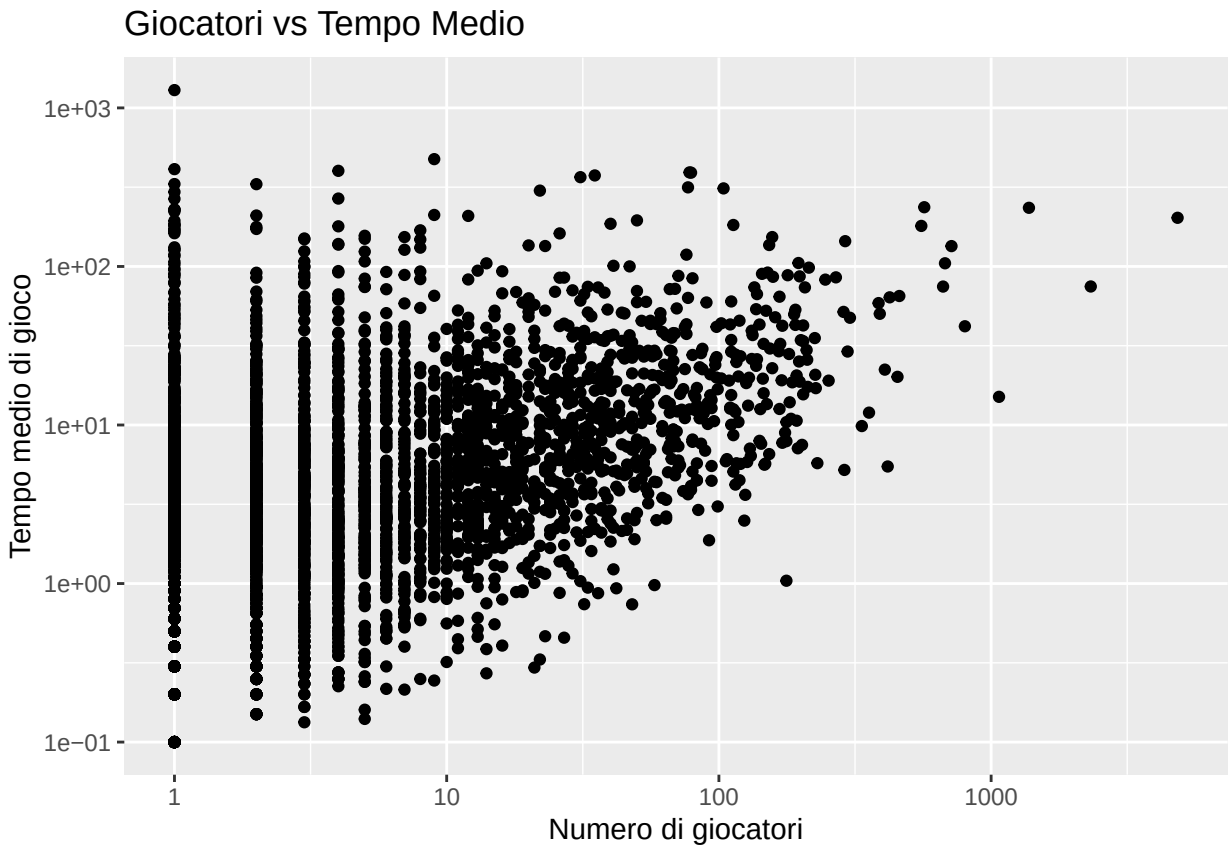


E'abbastanza probabile, anche se non certo, che i giochi nel range di ore da 0 a 5 siano stati in media provati e poi abbandonati dai giocatori. Il calo drastico dopo le 40 ore probabilmente è dovuto al fatto che molti

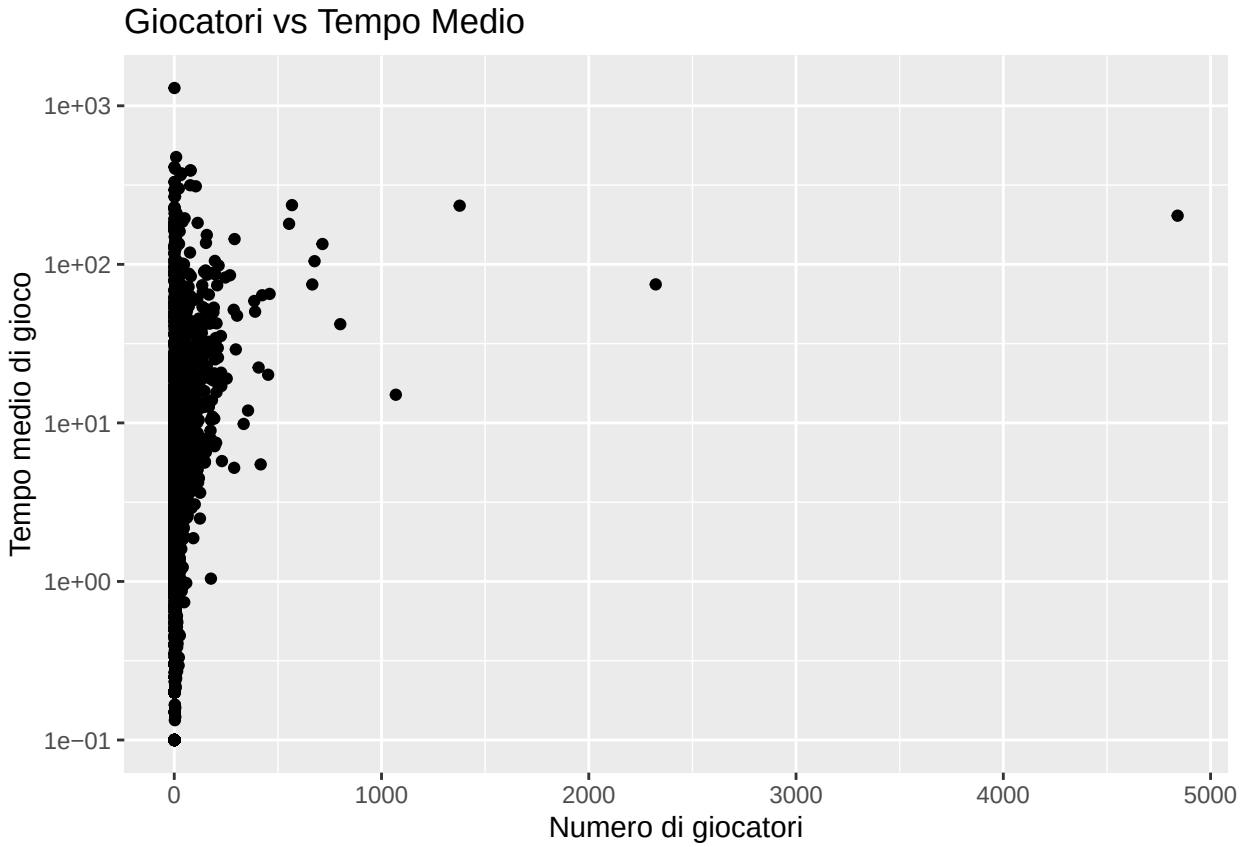
giochi a giocatore singolo terminano in meno tempo, indagheremo su questo in segito.

Concludiamo questa sezione valutando quanta correlazione ci sia tra i tempi medi di gioco e il numero di giocatori, per verificare se la selezione effettuata sul dataset sia o meno risultata sensata.

```
ggplot(games.avgplaytime, legend=FALSE) +
  labs(title = "Giocatori vs Tempo Medio") +
  ylab("Tempo medio di gioco") +
  xlab("Numero di giocatori") +
  geom_point(aes(y=avgPlayTime, x=players)) +
  theme(legend.position = "none") +
  scale_x_log10()+
  scale_y_log10()
```




```
ggplot(games.avgplaytime, legend=FALSE) +
  labs(title = "Giocatori vs Tempo Medio") +
  ylab("Tempo medio di gioco") +
  xlab("Numero di giocatori") +
  geom_point(aes(y=avgPlayTime, x=players)) +
  theme(legend.position = "none") +
  scale_y_log10()
```



I grafici mostrano chiaramente come con l'aumento del numero di giocatori la variabilità rispetto al tempo medio di gioco risulti diminuita:

```
# completo
var(games.avgplaytime$avgPlayTime)
```

```
## [1] 1530.496
```

```
mean(games.avgplaytime$avgPlayTime)
```

```
## [1] 13.33164
```

```
# >= 5
var((games.avgplaytime %>% filter(players >= 5))$avgPlayTime)
```

```
## [1] 1296.342
```

```
mean((games.avgplaytime %>% filter(players >= 5))$avgPlayTime)
```

```
## [1] 17.22091
```

```
# <= 5
var((games.avgplaytime %>% filter(players < 5))$avgPlayTime)
```

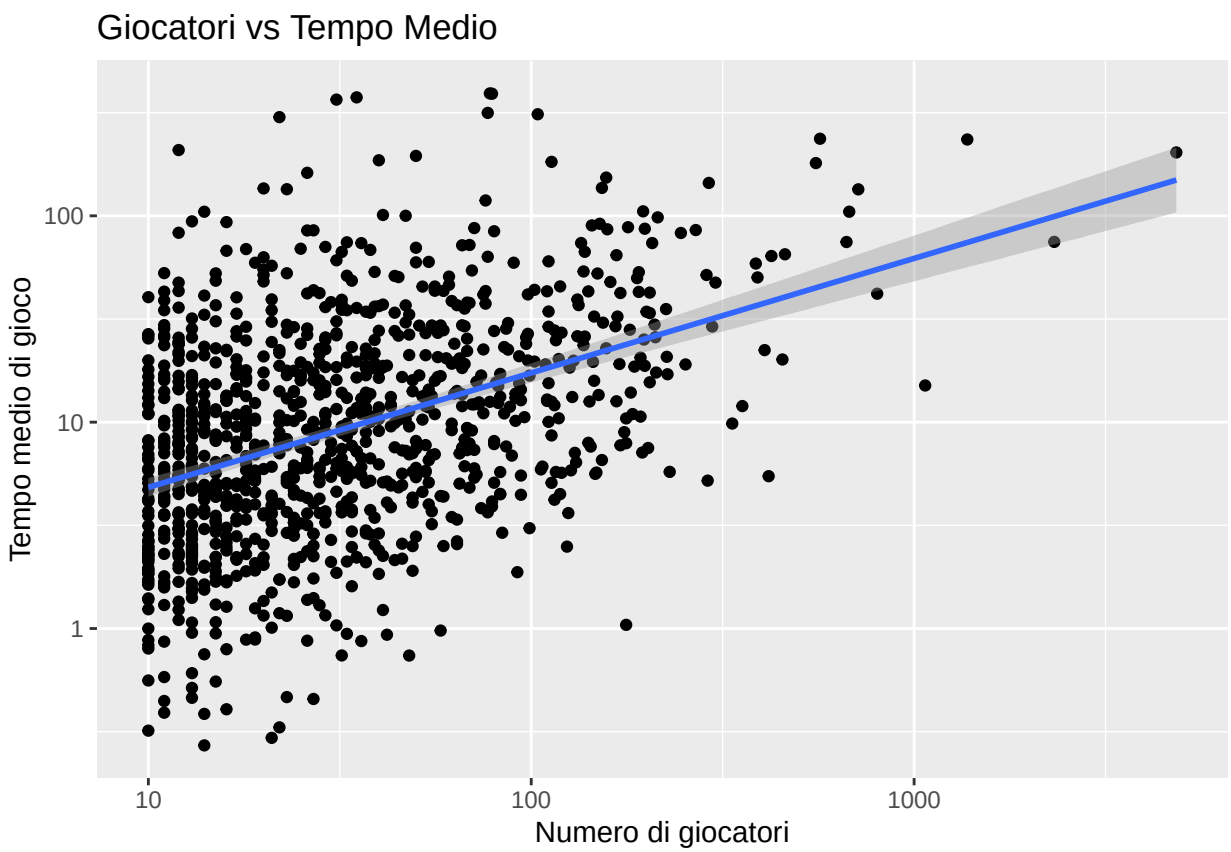
```
## [1] 1693.974
```

```
mean((games.avgplaytime %>% filter(players < 5))$avgPlayTime)
```

```
## [1] 10.27578
```

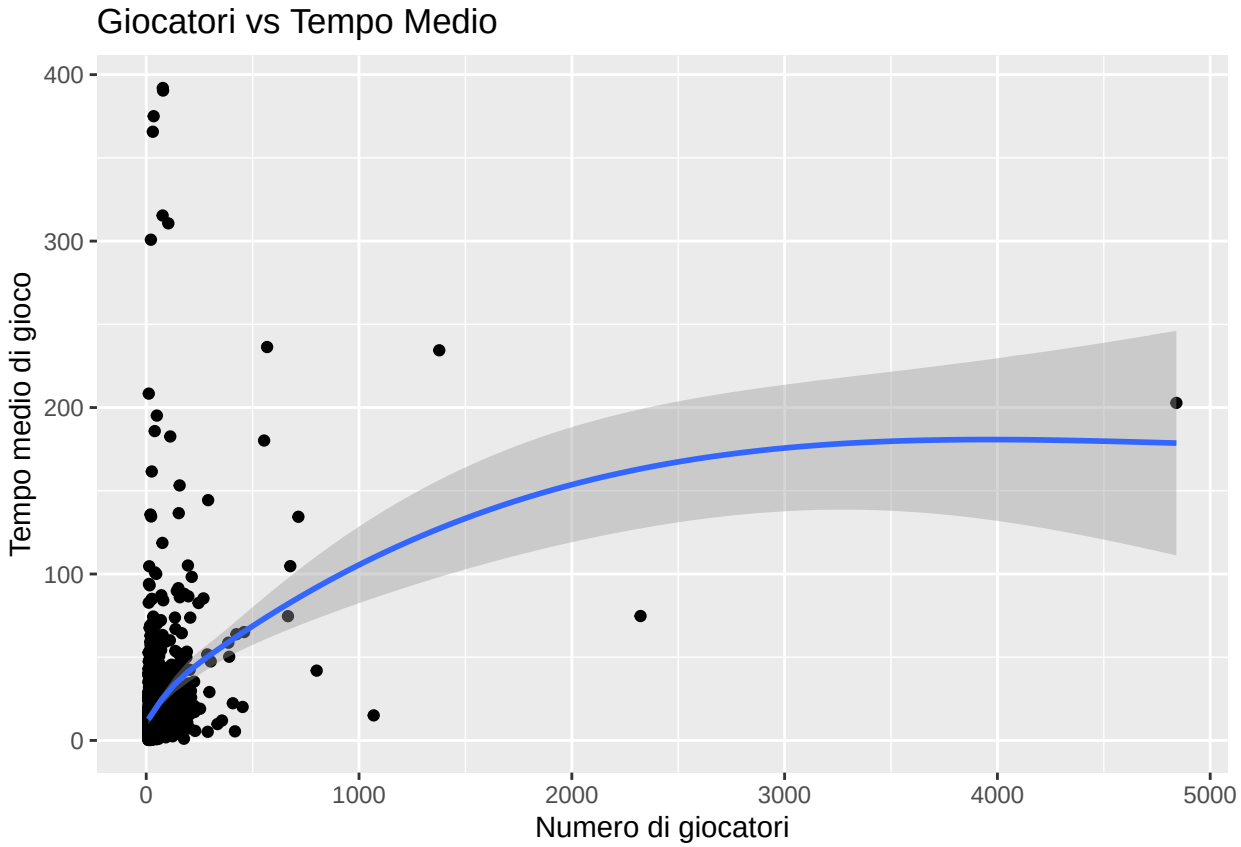
Questo conferma l'ipotesi precedente, concentrandosi sui giochi con più di dieci giocatori il trend risulta più chiaro.

```
ggplot(games.avgplaytime %>% filter(players >= 10), legend=FALSE) +
  labs(title = "Giocatori vs Tempo Medio") +
  ylab("Tempo medio di gioco") +
  xlab("Numero di giocatori") +
  geom_point(aes(y=avgPlayTime, x=players)) +
  theme(legend.position = "none") +
  geom_smooth(aes(y=avgPlayTime, x=players),method = "lm", formula=y~x) +
  scale_x_log10()+
  scale_y_log10()
```




```
ggplot(games.avgplaytime %>% filter(players >= 10), legend=FALSE) +
  labs(title = "Giocatori vs Tempo Medio") +
  ylab("Tempo medio di gioco") +
  xlab("Numero di giocatori") +
  geom_point(aes(y=avgPlayTime, x=players)) +
  theme(legend.position = "none") +
  geom_smooth(aes(y=avgPlayTime, x=players))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



mostrando quindi una tendenza all'aumento del tempo medio di gioco a seconda del numero di giocatori. Ritengo questo risultato particolarmente interessante, in quanto si potrebbe effettivamente pensare che i giochi con più giocatori tendano ad attirare anche molte persone con scarso interesse che successivamente andrebbero ad abbandonare il gioco, facendo diminuire così notevolmente la media. Potrebbe essere di interesse valutare se questo sia il trend anche nei giochi per dispsitivi mobili (android/iOS) dove il mercato e gli utenti sono solitamente molto diversi.

Dati congiunti

Dopo questa prima fase di analisi esplorativa, possiamo arricchire le informazioni sui giochi aquistati e giocati con quelle delle caratteristiche dei singoli giochi. Questo è possibile unendo le due tabelle per nome.

```
# Giochi e tempo di gioco (se disponibile)
play.data <- na.omit(full_join(by=c("name"), players.play, games.data))
number.of.classes(players.play, name)
```

```
## [1] 3600
```

```
number.of.classes(play.data, name)
```

```
## [1] 2516
```

Questi numeri ci indicano che non stiamo considerando più di mille giochi che erano stati giocati dai giocatori presenti nel dataset 200k. Questo perdita di informazioni è dovuta in primo luogo al fatto che stiamo cercando innanzitutto di unire due tabelle su un campo "nome" testuale, cosa inevitabile dato che nel dataset 200k non viene riportato l'appid associato ai giochi (un identificativo unico rilasciato da Steam per ogni gioco pubblicato). Certi casi invece riguardano il ritiro di vecchie versioni dei giochi dal commercio, per far spazio

a versioni rivisitate o migliorate (spesso chiamate remastered). Consideriamo due casi nello specifico, le due serie "Civilization" e "BioShock" per avere un'idea migliore della problematica:

```
q <- "civilization"
# dataset dei giochi
string.query(games.data, name, q) %>% get.unique(name) %>% arrange(desc(name)) %>% head(10)
```

```
##                                     name
## 1               sid meiers civilization vi
## 2                sid meiers civilization v
## 3  sid meiers civilization iv colonization
## 4               sid meiers civilization iv
## 5     sid meiers civilization iii complete
## 6     sid meiers civilization beyond earth
## 7                         lost civilization
## 8                         idle civilization
## 9                  heroes of civilizations
## 10              galactic civilizations iii
```

```
# dataset dei giocatori
string.query(players.play, name, q) %>% get.unique(name) %>% arrange(desc(name)) %>% head(10)
```

```
##                                       name
## 1                    sid meiers civilization v
## 2          sid meiers civilization iv warlords
## 3      sid meiers civilization iv colonization
## 4  sid meiers civilization iv beyond the sword
## 5                    sid meiers civilization iv
## 6         sid meiers civilization iii complete
## 7         sid meiers civilization beyond earth
## 8                   precivilization marble age
## 9                   galactic civilizations iii
## 10  galactic civilizations ii ultimate edition
```

```
# dataset uniti
string.query(play.data, name, q) %>% get.unique(name) %>% arrange(desc(name)) %>% head(10)
```

```
##                                       name
## 1                   sid meiers civilization v
## 2    sid meiers civilization iv colonization
## 3                  sid meiers civilization iv
## 4       sid meiers civilization iii complete
## 5       sid meiers civilization beyond earth
## 6                  galactic civilizations iii
## 7 galactic civilizations ii ultimate edition
## 8  galactic civilizations i ultimate edition
```

```
q <- "bioshock"
# dataset dei giochi
string.query(games.data, name, q) %>% get.unique(name) %>% arrange(desc(name)) %>% head(10)
```

```
##                    name
## 1   bioshock remastered
## 2     bioshock infinite
## 3 bioshock 2 remastered
```

```
# dataset dei giocatori
string.query(players.play, name, q) %>% get.unique(name) %>% arrange(desc(name)) %>% head(10)
```

```
##                name
## 1 bioshock infinite
## 2        bioshock 2
## 3          bioshock
```

```
# dataset uniti
string.query(play.data, name, q) %>% get.unique(name) %>% arrange(desc(name)) %>% head(10)
```

```
##                name
## 1 bioshock infinite
```

Nel caso di "Civilization" osserviamo che "sid meier's civilization iv warlords" e "sid meier's civilization iv beyond the sword" erano presenti tra i giochi giocati ma non tra i giochi di Steam noti, questo probabilmente è dato dal fatto che queste due sono espansioni di "civilization iv" e dunque ora non sono più considerate un gioco separato; ovviamente è anche possibile che il dataset con i giochi di Steam del 2019 sia incompleto. "lost" e "idle civilization" sono giochi meno noti, e non sono stati giocati da nessuno dei giocatori campionati. "Precivilization marble age" è presente solo come gioco giocato, a quanto sembra il suo nome è stato cambiato in solo "Marble age". In totale quindi non è stato possibile aggiungere le ulteriori informazioni a tre diversi giochi. Il caso della serie "BioShock" è invece emblematico per la seconda problematica citata in precedenza: in questo caso si è passati da versioni "base" a versioni "remastered" dello stesso gioco, con conseguente cambio del nome.

Continueremo le analisi con questo dataset ridotto ma informativo, quantifihiamo ora la perdita di giochi e giocatori rispetto al dataset originale:

```
# giocatori persi
number.of.classes(players.play, player) - number.of.classes(play.data, player)
```

```
## [1] 800
```

```
# giochi persi
number.of.classes(players.play, name) - number.of.classes(play.data, name)
```

```
## [1] 1084
```

Il dataset rimane comunque abbastanza ricco per le successive analisi:

```
# giocatori finali
number.of.classes(play.data, player)
```

```
## [1] 10550
```

```
# giochi finali
number.of.classes(play.data, name)
```

```
## [1] 2516
```

Datazione del dataset:

Individuare la problematica che riguarda i giochi "BioShock" ci permette di determinare un intervallo all'interno del quale i dati sono stati acquisiti in quanto i giochi originali sono stati completamente eliminati da steam e sostituiti per tutti gli utenti dalle nuove versioni nel Settembre 2016.

```
play.data %>% select(name, release_date) %>% unique() %>% arrange(desc(release_date)) %>% head(10)
```

```
##                      name release_date
## 1          space engineers   2019-02-28
## 2               flashback   2019-02-28
## 3                    snow   2019-02-14
```

```
## 4       farlight explorers   2019-02-01
## 5  uncharted waters online   2018-12-19
## 6               miscreated   2018-12-18
## 7                     dayz   2018-12-13
## 8                   kenshi   2018-12-06
## 9                     maia   2018-11-23
## 10                requiem   2018-11-02
```

come si può vedere, l'analisi sembra mostrarci diversi giochi che effettivamente risultano essere successivi al 2016, questa situazione è dovuta al fatto che molti giochi approdano su Steam in "Early Access" ossia come giochi incompleti ma già acquistabili e giocabili, che successivamente verranno definitivamente pubblicati quando pronti. Questo significa che spesso i giocatori giocano ad alcuni giochi prima della loro vera e propria uscita e, quando gli sviluppatori prendono alla lettera il concetto di *beta perpetua*, anche diversi anni prima. Si consideri il caso di "space engineers", è entrato in "Early Access" nel 2013 ma è stato pubblicato solamente all'inizio del 2019. Quindi è necessario cercare un videogioco che non abbia avuto questa fase di pre-lancio o che l'abbia avuta poco prima del settembre 2016. Scegliamo quindi il gioco "out there somewhere" pubblicato su Steam nel il 2016-03-14 e che non ha avuto una fase ad accesso anticipato in quanto porting per la piattaforma Steam di un gioco realizzato nel 2012.

Se non si è convinti che le considerazioni riguardanti la remasterizzazione di "BioShock" siano attendibili, si consideri che il gioco "Civilization VI" è uscito nell'ottobre del 2016 e che non è stato individuato nessun giocatore nel campione analizzato che lo abbia giocato. Possiamo calcolare (una sottostima) di quale sia la probabilità di un tale evento nel caso in cui il campionamento sia stato effettuato dopo l'uscita di questo gioco. Per farlo usiamo due dati non disponibili direttamente sul dataset:

- Il massimo numero di utenti contemporaneamente giocanti a Civilization VI nel solo mese di Ottobre 2016 è stato di 162.310 utenti (una sottostima degli utenti totali, considerando che in sole due settimane aveva superato il milione di copie vendute)
- Nel 2018 si sono registrati circa 90 Milioni di giocatori attivi mensilmente, assumeremo che i dati siano stati ottenuti da utenti attivi. Se così non fosse si stima che gli utenti Steam complessivi possano aver raggiunto il miliardo nel 2019, ma questo numero è molto difficile da provare e racchiude certamente molti utenti "inesistenti".

```
# numero di campioni
number.of.classes(players.play, player)
```

```
## [1] 11350
```

```
# probabilità di non estrazione (caso realistico)
(1-(162310/90000000))**number.of.classes(players.play, player)
```

```
## [1] 1.265784e-09
```

```
# probabilità di non estrazione (caso estremo)
(1-(162310/1000000000))**number.of.classes(players.play, player)
```

```
## [1] 0.1584418
```

```
# probabilità di non estrazione (caso estremo, con milione di copie vendute)
(1-(1000000/1000000000))**number.of.classes(players.play, player)
```

```
## [1] 1.170284e-05
```

Il che conferma come sia molto difficile pensare che non aver individuato giocatori di "Civilization VI" possa essere dovuto al caso. Possiamo concludere con una certa convinzione quindi che il dataset 200k risalga a più o meno la metà del 2016.

Valutazione dei punteggi degli utenti

Per mostrare le possibilità del nuovo dataset creato, consideriamo ora la seguente query che ci mostra la frequenza dei voti positivi per ogni gioco della serie "Sid Meier's Civilization" giocato nel dataset 200k:

```
string.query(games.data, name, "sid meiers civilization") %>%
  filter.by.tag.or(steamspy_tags, c("Strategy")) %>%
  mutate(score = ifelse( positive_ratings > 100, (positive_ratings)/(positive_ratings+negative_ratings)
  select(name,score) %>%
  arrange(desc(score)) %>% head(10)
```

```
##                                     name     score
## 1               sid meiers civilization v 0.9586537
## 2              sid meiers civilization iv 0.9152988
## 3    sid meiers civilization iii complete 0.8642306
## 4 sid meiers civilization iv colonization 0.7764706
## 5              sid meiers civilization vi 0.6810381
## 6    sid meiers civilization beyond earth 0.5374318
```

E' interessante notare come, dopo "Civilization V", i due seguiti "beyond earth" e "VI" risultino avere un punteggio decisamente più basso. Possiamo chiederci se questi siano casi del cosiddetto fenomeno del "Review Bombing", per il quale un gioco viene bersagliato in modo sistematico da valutazioni negative da parte degli utenti. Questo può accadere per molti motivi diversi: non raggiungimento delle aspettative, politiche aziendali non accettate dai fan, problemi dal punto di vista dell'implementazione del gioco (per bug o ottimizzazione), eccetera ... . Per quanto questa pratica possa sembrare scorretta in quanto valuta il gioco in un determinato contesto e momento del suo ciclo di vita, è in realtà l'unica mossa da parte degli appassionati per poter far effettivamente sentire la loro opinione. La stampa specializzata che si occupa invece della recensione sistematica dei giochi in uscita normalmente non è affetta da queste situazioni e valuta i diversi giochi con il proprio metro di giudizio normale. Ovviamente, anche fra la stampa specializzata, possono esserci opinioni diverse nella valutazione dei giochi in quanto questi sono prodotti con una componente artistica e dunque prettamente soggettiva, similmente a quanto accade normalmente ad esempio in ambito cinematografico. Per questo motivo, per poter avere una valutazione più oggettiva dei prodotti che risenta in minor modo delle opinioni personali dei vari recensori si è sviluppato il sito *metacritic*, che ha lo scopo di raccogliere le recensioni ufficiali di diversi media, tra i quali anche i videogiochi, e di compararle e mediarle. Il sito fornisce infine un numero per ogni prodotto, detto *metascore*, che rappresenta la media delle valutazioni che ha ricevuto. Questa media viene pesata anche secondo l'autorità del recensore, associata principalmente al numero di recensioni redatte. Si noti che il *metascore* non include alcuna valutazione da parte degli utenti, che invece sono trattate in modo separato nel sito. (per altre informazioni, si consulti questo link).

Per accedere a questi dati, consideriamo un altro dataset da Kaggle, Metacritic all time games stats. Si noti che solamente i giochi con un po'di rilievo riescono ad essere considerati dai recensori e così ad ottenere un *metascore*, quindi ci attendiamo l'assenza di questa informazione per alcuni dei giochi considerati, in quel caso sarà necessario affidarsi solamente agli utenti.

```
metacritic.data <- read.csv(
    "./data/metacritic_games.csv")
metacritic.data %>% head(10)
```

```
##                          name platform              developer
## 1           Command & Conquer       PC       Westwood Studios
## 2               Full Throttle       PC               LucasArts
## 3      Battle Arena Toshinden       PS                 Tamsoft
## 4  Sid Meier's Civilization II       PC                MPS Labs
## 5                       Quake       PC             id Software
## 6                      Diablo       PC Blizzard Entertainment
## 7               Super Mario 64      N64                Nintendo
```

```
## 8                  Wipeout XL       PS              Psygnosis
## 9                Wave Race 64      N64               Nintendo
## 10                Tomb Raider       PS       Core Design Ltd.
##                 publisher         genre.s.  players rating attribute
## 1      Virgin Interactive          Sci-Fi     1-4       T
## 2               LucasArts        Adventure
## 3                    SCEA           Action     1-2       T
## 4              MicroProse         Strategy 1 Player    K-A
## 5             id Software           Action    1-16       M
## 6  Blizzard Entertainment     Role-Playing     1-4       M
## 7                Nintendo           Action 1 Player      E
## 8                    SCEE          Driving     1-2
## 9                Nintendo          Driving     1-2       E
## 10      Eidos Interactive Action Adventure 1 Player      T
##    release_date                                    link critic_positive
## 1  Aug 31, 1995                 /game/pc/command-conquer               5
## 2  Apr 30, 1995                   /game/pc/full-throttle               6
## 3  Sep  9, 1995 /game/playstation/battle-arena-toshinden               1
## 4  Feb 29, 1996      /game/pc/sid-meiers-civilization-ii               7
## 5  Jun 22, 1996                           /game/pc/quake               9
## 6  Dec 31, 1996                         /game/pc/diablo               12
## 7  Sep 26, 1996         /game/nintendo-64/super-mario-64               13
## 8  Sep 30, 1996             /game/playstation/wipeout-xl               8
## 9  Nov  1, 1996           /game/nintendo-64/wave-race-64               12
## 10 Nov 15, 1996            /game/playstation/tomb-raider               13
##    critic_neutral critic_negative metascore user_positive user_neutral
## 1               0               0        94            47            0
## 2               2               0        86            18            1
## 3               3               0        69             1            0
## 4               0               0        94            46            0
## 5               0               0        94            84            4
## 6               0               0        94            84            8
## 7               0               0        94           220           14
## 8               0               0        93             4            1
## 9               1               0        92            15            2
## 10              0               0        91            31            4
##    user_negative user_score
## 1              1        8.9
## 2              0        8.7
## 3              1        5.8
## 4              1        8.9
## 5              1        8.8
## 6              7        8.7
## 7             10        9.2
## 8              1        8.5
## 9              0        8.2
## 10             1        8.4
```

```
# la colonna user_score deve essere numerica
metacritic.data <- mutate(metacritic.data, user_score = as.numeric(gsub("\\.", "",user_score)))
```

```
## Warning: si è prodotto un NA per coercizione
```

```
# semplifico i nomi per poter effettuare il join con l'altro dataset
metacritic.data <- clean.text(metacritic.data, name, "[-.:®'’]")
```

```r
# considero solo la piattaforma PC e filtro alcune colonne non necessarie
metacritic.data <- metacritic.data %>% filter(platform == "PC") %>%
  select(name, genre.s., players, rating, metascore, user_score, release_date,
         critic_positive, critic_neutral, critic_negative, user_positive, user_neutral, user_negative) %
  mutate(critic_total = critic_positive + critic_neutral + critic_negative, user_total = user_positive +
  mutate(rating_metacritic = rating) %>% select(-rating)
# gestisco le date
Sys.setlocale("LC_TIME", "C")
```

```
## [1] "C"
```

```r
metacritic.data <- metacritic.data %>% mutate(release_date = tolower(gsub(",", " ",release_date))) %>%
  mutate(release_date = format(as.Date(strptime( release_date, "%b %d %Y")), "%d-%m-%y"))
metacritic.data %>% head(10)
```

```
##                           name     genre.s.  players metascore user_score
## 1             command & conquer       Sci-Fi      1-4         94         89
## 2                 full throttle    Adventure                  86         87
## 3   sid meiers civilization ii     Strategy 1 Player         94         89
## 4                         quake       Action     1-16         94         88
## 5                        diablo Role-Playing      1-4         94         87
## 6  command & conquer red alert     Strategy      1-6         90         88
## 7                star control 3     Strategy      1-2         89         50
## 8                 duke nukem 3d       Action      1-4         89         87
## 9                    descent ii       Action      1-8         89         87
## 10                     obsidian    Adventure 1 Player         85         71
##    release_date critic_positive critic_neutral critic_negative user_positive
## 1      31-08-95               5              0               0            47
## 2      30-04-95               6              2               0            18
## 3      29-02-96               7              0               0            46
## 4      22-06-96               9              0               0            84
## 5      31-12-96              12              0               0            84
## 6      31-10-96               7              0               0            25
## 7      31-08-96               5              0               0             1
## 8      29-01-96               8              0               0            39
## 9      29-02-96               6              0               0             5
## 10     31-12-96               8              1               0             1
##    user_neutral user_negative critic_total user_total rating_metacritic
## 1             0             1            5         48                 T
## 2             1             0            8         19
## 3             0             1            7         47               K-A
## 4             4             1            9         89                 M
## 5             8             7           12         99                 M
## 6             3             0            7         28                 T
## 7             4             6            5         11                 E
## 8             3             2            8         44                 M
## 9             0             0            6          5                 T
## 10            0             0            9          1               K-A
```

Soffermiamoci ad analizzare questo dataset prima di continuare rispondendo alla domanda che ci siamo posti.

Il numero di giochi recensiti è:

```r
number.of.classes(metacritic.data, name)
```

```
## [1] 5436
```

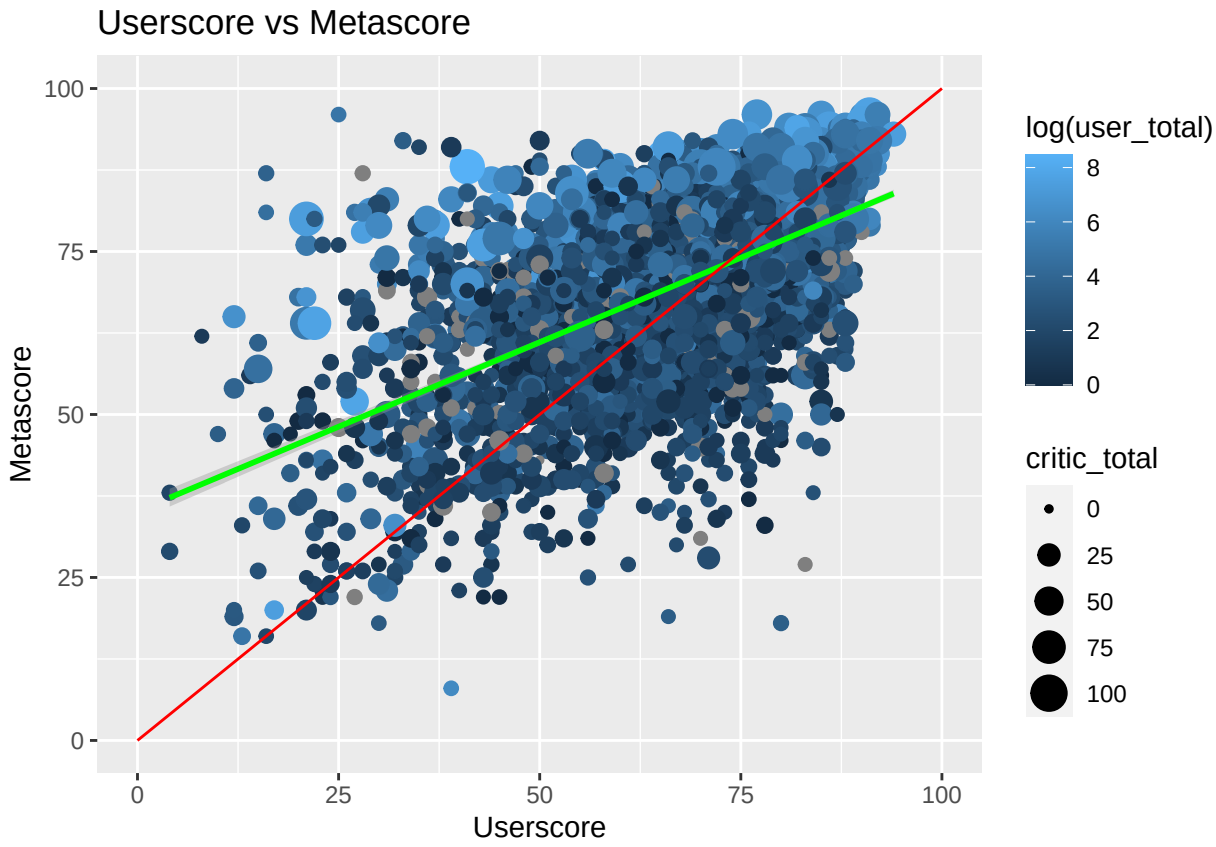
Il che ci indica come esistano giochi con lo stesso nome, di questi selezioniamo solo il più recente:

```
metacritic.data <- metacritic.data %>% group_by(name, release_date) %>% arrange(name, release_date) %>%
```

Per prima cosa proviamo a vedere internamente al sito metacritic la correlazione tra voto degli utenti e dei recensori:

```
ggplot(na.omit(metacritic.data)) +
  geom_point(aes(x=user_score, y=metascore, col=log(user_total), size=critic_total)) +
  geom_smooth(aes(x=user_score, y=metascore), method = "lm", color = "green") +
  geom_line(data = data.frame(x = seq(0,100), y = seq(0,100)), aes(x=x,y=y), color = "red")+
  labs(title = "Userscore vs Metascore") +
  ylab("Metascore") +
  xlab("Userscore")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# mostrare i nomi degli outliers
```

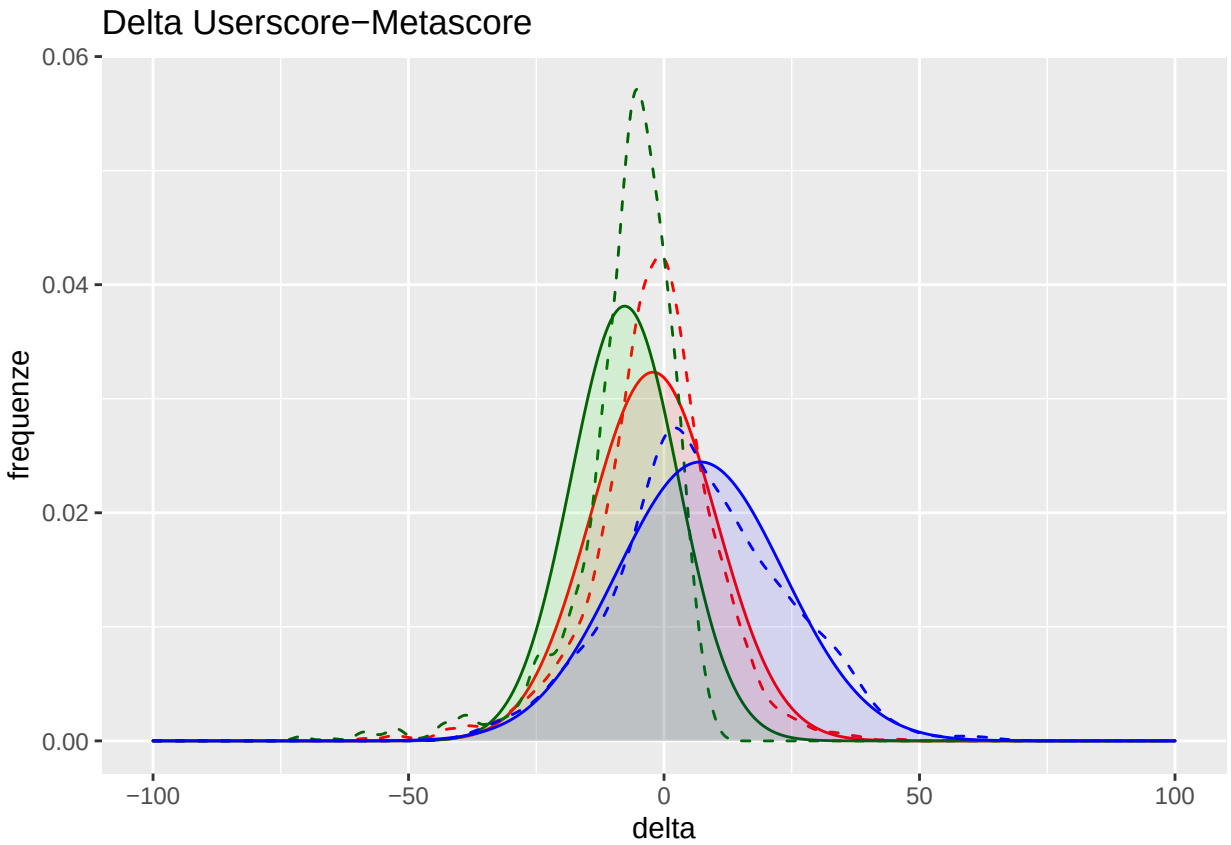
Si noti che i punti al di sopra della retta rossa sono stati valutati meglio dalla critica che dagli utenti, viceversa per i punti al di sotto.

Visiualizziamo ora la distribuzione della differenza fra punteggi della critica e degli utenti:

```
md.no.NA <- na.omit(metacritic.data)
md.no.NA.filtered.hi <- filter(md.no.NA, metascore >= 80)
md.no.NA.filtered.lo <- filter(md.no.NA, metascore <= 50)

ggplot(md.no.NA) +
  geom_density(aes(x=user_score-metascore), col="red", linetype = "dashed") +
```

```
  geom_area( data = data.frame(x=seq(-100,100,0.1), y=dnorm(seq(-100,100,0.1),
                mean=mean(md.no.NA$user_score-md.no.NA$metascore),
                sd=sd(md.no.NA$user_score-md.no.NA$metascore))),
             aes(x=x,y=y), col="red", fill="red", alpha = 0.1) +
  geom_density(data = md.no.NA.filtered.hi, aes(x=user_score-metascore), col="darkgreen", linetype = "da
  geom_area( data = data.frame(x=seq(-100,100,0.1), y=dnorm(seq(-100,100,0.1),
                mean=mean(md.no.NA.filtered.hi$user_score-md.no.NA.filtered.hi$metascore),
                sd=sd(md.no.NA.filtered.hi$user_score-md.no.NA.filtered.hi$metascore))),
             aes(x=x,y=y), col="darkgreen", fill="green", alpha = 0.1) +
  geom_density(data = md.no.NA.filtered.lo, aes(x=user_score-metascore), col="blue", linetype = "dashed
  geom_area(data = data.frame(x=seq(-100,100,0.1), y=dnorm(seq(-100,100,0.1),
                mean=mean(md.no.NA.filtered.lo$user_score-md.no.NA.filtered.lo$metascore),
                sd=sd(md.no.NA.filtered.lo$user_score-md.no.NA.filtered.lo$metascore))),
             aes(x=x,y=y), col="blue", fill="blue", alpha = 0.1) +
  labs(title = "Delta Userscore-Metascore") +
  ylab("frequenze") +
  xlab("delta")
```



Ove verde indica i giochi di maggior successo (metascore >= 80), Blu di minor successo (metascore <= 50), mentre le curve rosse sono associate all'intero dataset. Per ogni selezione sono state mostrate distribuzione empirica e approssimata a gaussiana.

si osseva una certa tendenza per gli utenti a proporre voti lievemente più negativi rispetto alla critica in genreale, le situazioni si estremizzano nel caso dei giochi con alto punteggio e si invertono con quelli con basso punteggio:

```
# discrepanza media (tutto il dataset)
mean(md.no.NA$user_score-md.no.NA$metascore)
```

```
## [1] -2.148363
```

```
# discrepanza media (metascore >= 80)
mean(md.no.NA.filtered.hi$user_score-md.no.NA.filtered.hi$metascore)
```

```
## [1] -7.701713
```

```
# discrepanza media (metascore <= 50)
mean(md.no.NA.filtered.lo$user_score-md.no.NA.filtered.lo$metascore)
```

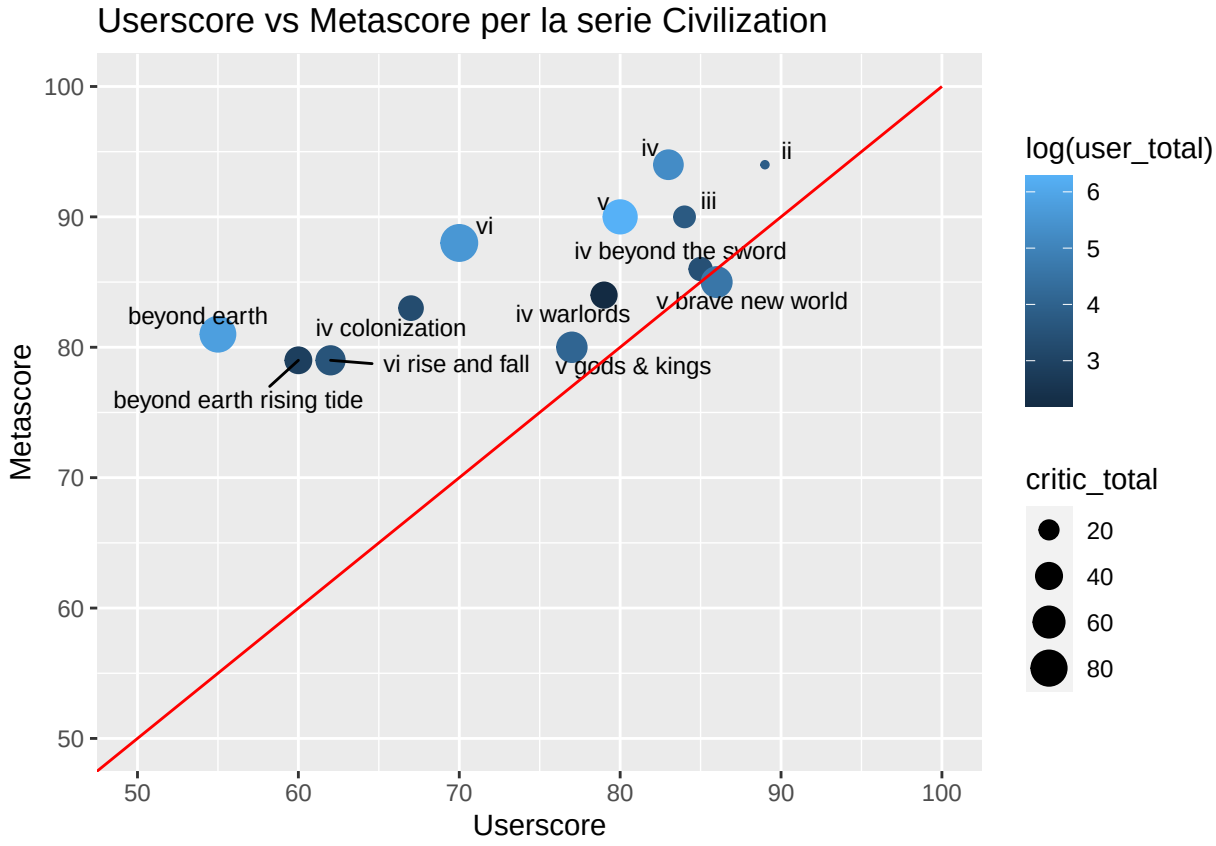
```
## [1] 7.098361
```

Si notano inoltre delle "gobbe" evidenti sulla coda sinistra della distribuzione empirica associata ai giochi con metascore molto positivo, probabilmente possono essere proprio effetti legati al "Review bombing".

Questo ci mostra come i giochi con uno score basso siano valutati più positivamente dai critici che dagli utenti e vice-versa per i punteggi più alti, questo probabilmente potrebbe essere dovuto al fatto che il voto degli utenti è meno ragionato in media e più basato sulle sensazioni dirette, quindi semplicemente se un gioco è piaciuto avrà una valutazione alta, altrimenti una bassa.

Concentriamoci sui giochi della serie "Civilization":

```
ggplot(na.omit(metacritic.data %>% string.query(name, "sid meiers civilization"))) +
  geom_point(aes(x=user_score, y=metascore, col=log(user_total), size=critic_total)) +
  geom_node_text( aes(x=user_score, y=metascore, label = gsub("sid meiers civilization","",name)), colo
  geom_line(data = data.frame(x = seq(0,100), y = seq(0,100)), aes(x=x,y=y), color = "red")+
  labs(title = "Userscore vs Metascore per la serie Civilization") +
  ylab("Metascore") +
  xlab("Userscore") +
  coord_cartesian(xlim=c(50,100),ylim=c(50,100))
```



Come si può notare, il metascore della serie "Civilization" si attesta a vaolri pittosto alti e consistenti, lo userscore invece spazia dall'insufficiente al molto buono.

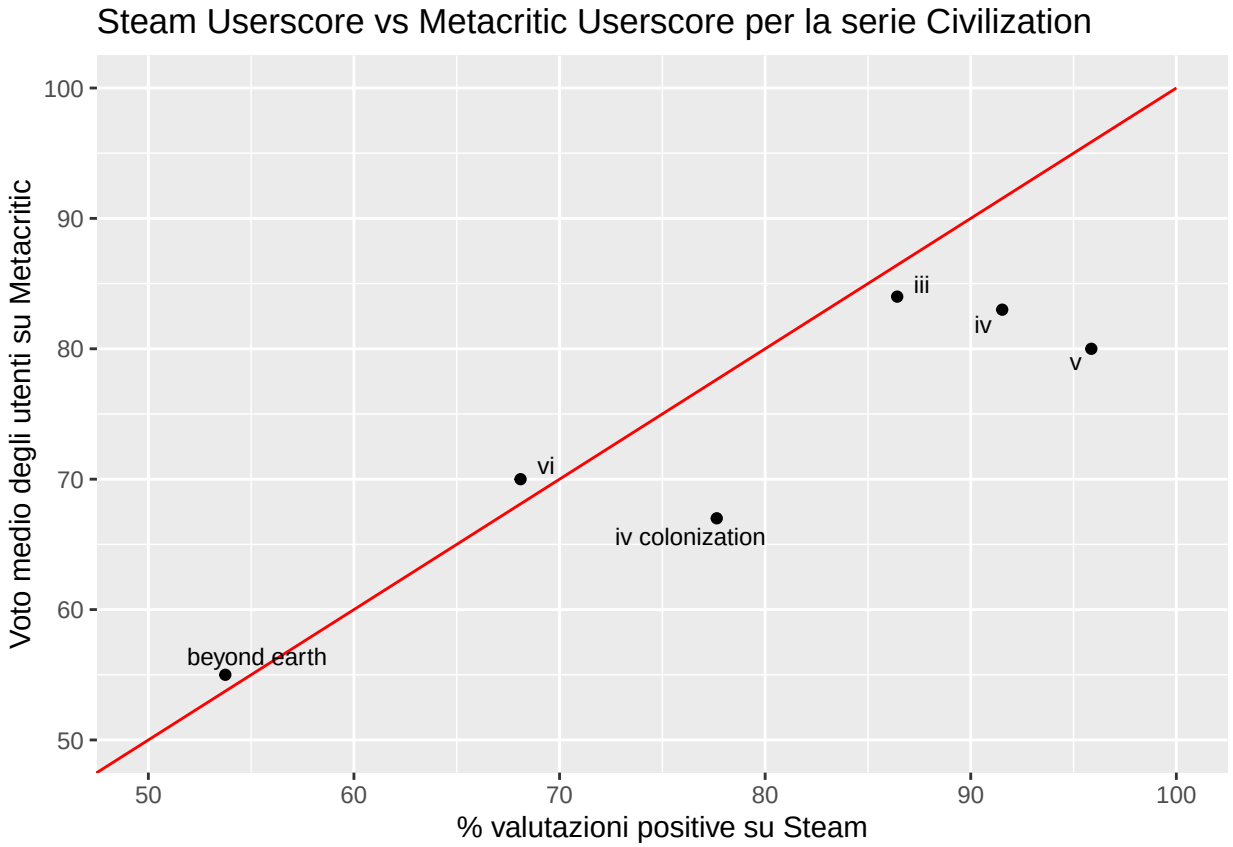
Uniamo queste informazioni ai dati di gioco:

```
played.civs <- md.no.NA %>% full_join(games.data %>% mutate(
    name = unlist(map(name, ~ gsub("sid meiers civilization iii complete","sid meiers civilization iii"
    by=c("name")) %>%
  string.query(name, "sid meiers civilization") %>% na.omit()
played.civs
```

```
## # A tibble: 6 x 33
## # Groups:   name [6]
##   name  genre.s. players metascore user_score release_date.x critic_positive
##   <chr> <fct>    <fct>       <int>      <dbl> <chr>                    <int>
## 1 sid ~ Strategy ""             81         55 23-10-14                    65
## 2 sid ~ Strategy "1 Pla~        90         84 30-10-01                    22
## 3 sid ~ Strategy "Onlin~        94         83 25-10-05                    50
## 4 sid ~ Strategy ""             83         67 22-09-08                    28
## 5 sid ~ Strategy "Up to~        90         80 21-09-10                    66
## 6 sid ~ Strategy "Onlin~        88         70 20-10-16                    79
## # ... with 26 more variables: critic_neutral <int>, critic_negative <int>,
## #   user_positive <int>, user_neutral <int>, user_negative <int>,
## #   critic_total <int>, user_total <int>, rating_metacritic <fct>, appid <dbl>,
## #   release_date.y <date>, english <fct>, developer <fct>, publisher <fct>,
## #   platforms <list>, required_age <fct>, categories <list>, genres <list>,
## #   steamspy_tags <list>, achievements <fct>, positive_ratings <int>,
## #   negative_ratings <int>, average_playtime <dbl>, median_playtime <dbl>,
## #   owners_lwb <int>, owners_upb <int>, price <dbl>
```

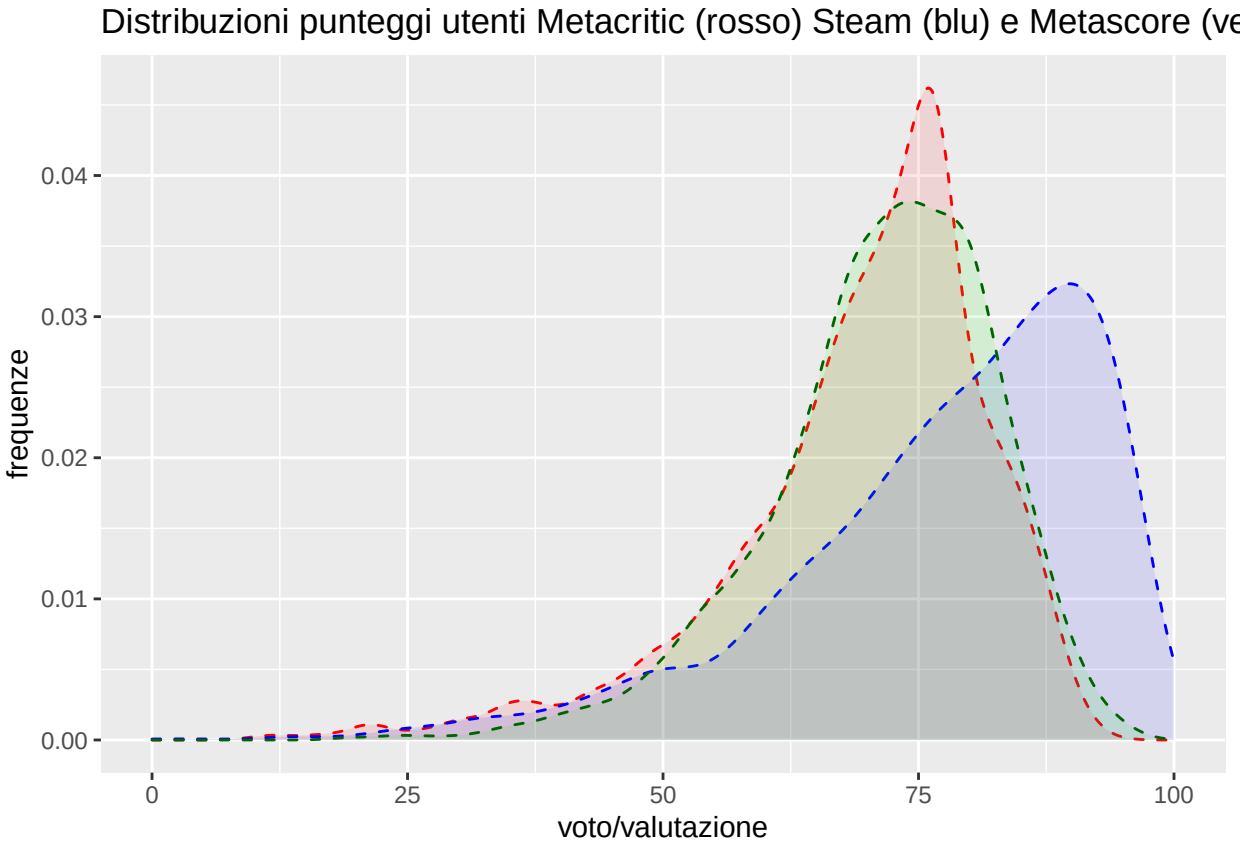
Confrontiamo ora le valutazioni degli utenti di Metacritic e Steam:

```
ggplot(played.civs %>% mutate(steam_user_score = 100*(positive_ratings)/(positive_ratings+negative_ratin
  geom_point(aes(x=steam_user_score, y=user_score)) +
  geom_line(data = data.frame(x = seq(0,100), y = seq(0,100)), aes(x=x,y=y), color = "red") +
  geom_node_text( aes(x=steam_user_score, y=user_score,
                      label = gsub("sid meiers civilization","",name)), color = "black", size = 3, repel
  labs(title = "Steam Userscore vs Metacritic Userscore per la serie Civilization") +
  ylab("Voto medio degli utenti su Metacritic") +
  xlab("% valutazioni positive su Steam") +
  coord_cartesian(xlim=c(50,100),ylim=c(50,100))
```



Come si può vedere, su "Civilization VI" sembra esserci un discreto consenso. Ci si può chiedere come si comparino le distribuzioni dei punteggi basati su Metascore, utenti Metacritic e utenti Steam.

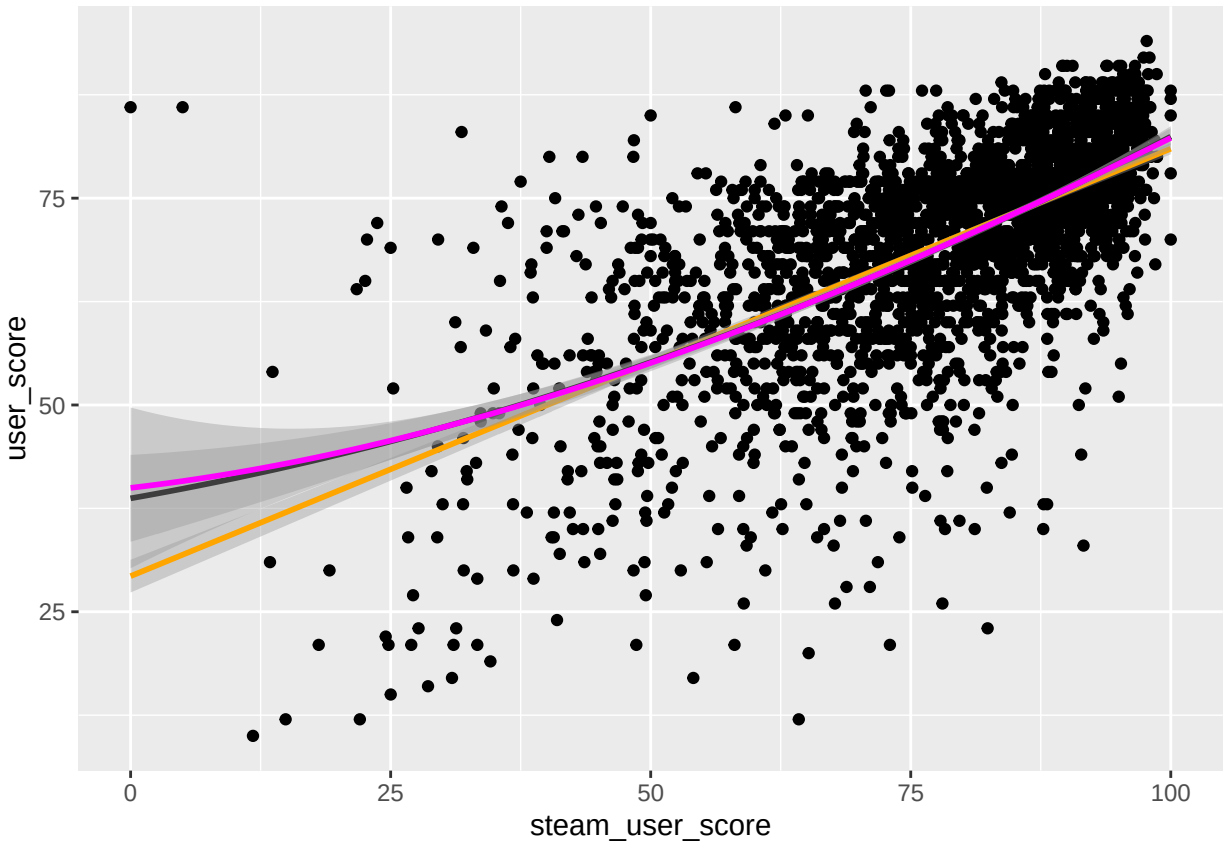
```
ggplot(md.no.NA %>% full_join(games.data,by=c("name")) %>% na.omit() %>%
         mutate(steam_user_score = 100*(positive_ratings)/(positive_ratings+negative_ratings)) ) +
  geom_density(aes(x=user_score), col="red", fill="red", linetype = "dashed", alpha=0.1) +
  geom_density(aes(x=steam_user_score), col="blue", fill="blue", linetype = "dashed", alpha=0.1) +
  geom_density(aes(x=metascore), col="darkgreen", fill="green", linetype = "dashed", alpha=0.1) +
  labs(title = "Distribuzioni punteggi utenti Metacritic (rosso) Steam (blu) e Metascore (verde)") +
  ylab("frequenze") +
  xlab("voto/valutazione")
```



Si osserva come la percentuale di valutazioni di apprezzamento degli utenti di steam non ha una distribuzione simile a quella dei voti. Costruiamo un modello di regressione lineare generalizzato per per cercare di rendere voti e valutazioni direttamente comparabili.

```
df <- md.no.NA %>% full_join(games.data,by=c("name")) %>% na.omit() %>%
        mutate(steam_user_score = 100*(positive_ratings)/(positive_ratings+negative_ratings)) %>% sele

ggplot( df ) +
  geom_point(aes(x=steam_user_score, y=user_score)) +
  geom_smooth(aes(y=user_score, x=steam_user_score), color="orange", method = "glm", formula = y~x) +
  geom_smooth(aes(y=user_score, x=steam_user_score), color="black", method = "glm", formula = y~x+I(x^2)
  geom_smooth(aes(y=user_score, x=steam_user_score), color="magenta", method = "glm", formula = y~x+I(x
```




```
l1 <- lm(user_score~steam_user_score , data = df)
l2 <- lm(user_score~steam_user_score + I(steam_user_score^2) , data = df)
l3 <- lm(user_score~steam_user_score + I(steam_user_score^2) + I(steam_user_score^3) , data = df)

AIC(l1,l2,l3) %>% head(10)
```

```
##    df      AIC
## l1  3 18839.77
## l2  4 18827.47
## l3  5 18829.38
```

```
BIC(l1,l2,l3) %>% head(10)
```

```
##    df      BIC
## l1  3 18857.29
## l2  4 18850.82
## l3  5 18858.57
```

```
summary(l1)
```

```
##
## Call:
## lm(formula = user_score ~ steam_user_score, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.501  -4.871   0.942   6.169  56.690
##
## Coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      29.31019    1.00870   29.06   <2e-16 ***
## steam_user_score  0.51668    0.01275   40.51   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.936 on 2533 degrees of freedom
## Multiple R-squared:  0.3932, Adjusted R-squared:  0.393
## F-statistic:  1641 on 1 and 2533 DF,  p-value: < 2.2e-16
```

```
summary(l2)
```

```
##
## Call:
## lm(formula = user_score ~ steam_user_score + I(steam_user_score^2),
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.839  -4.822   1.046   6.201  47.288
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.871e+01  2.680e+00  14.447  < 2e-16 ***
## steam_user_score      2.213e-01  7.906e-02   2.799 0.005166 **
## I(steam_user_score^2) 2.160e-03  5.705e-04   3.785 0.000157 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.91 on 2532 degrees of freedom
## Multiple R-squared:  0.3966, Adjusted R-squared:  0.3961
## F-statistic: 832.2 on 2 and 2532 DF,  p-value: < 2.2e-16
```

```
summary(l3)
```

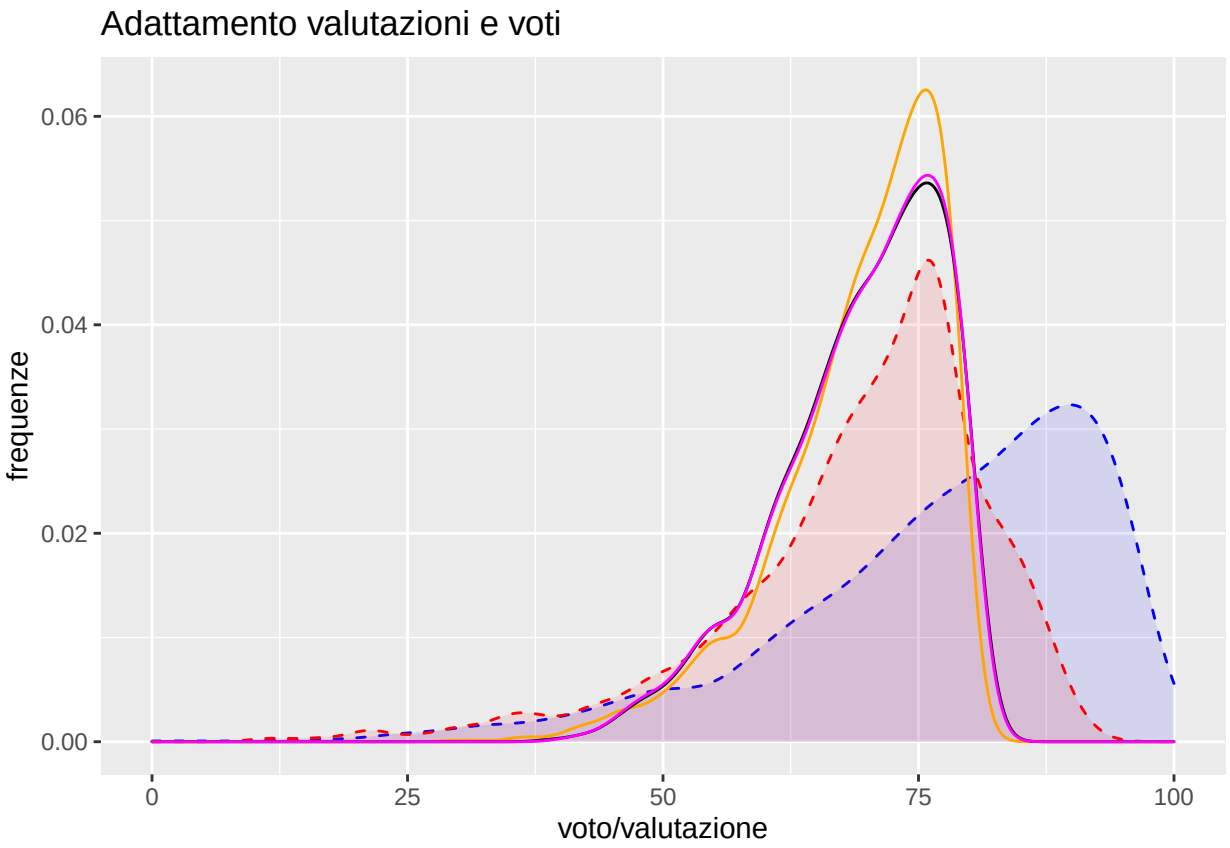
```
##
## Call:
## lm(formula = user_score ~ steam_user_score + I(steam_user_score^2) +
##     I(steam_user_score^3), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.821  -4.819   1.055   6.203  46.021
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.998e+01  4.956e+00   8.067  1.1e-15 ***
## steam_user_score       1.463e-01  2.588e-01   0.565    0.572
## I(steam_user_score^2)  3.460e-03  4.313e-03   0.802    0.423
## I(steam_user_score^3) -6.929e-06  2.278e-05  -0.304    0.761
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.912 on 2531 degrees of freedom
## Multiple R-squared:  0.3966, Adjusted R-squared:  0.3959
```

```
## F-statistic: 554.6 on 3 and 2531 DF,  p-value: < 2.2e-16
```

```
f1 <- function(x) l1$coefficients[1] + l1$coefficients[2]*x
f2 <- function(x) l2$coefficients[1] + l2$coefficients[2]*x + l2$coefficients[3]*(x**2)
f3 <- function(x) l3$coefficients[1] + l3$coefficients[2]*x + l3$coefficients[3]*(x**2) + l3$coefficien
```

Il secondo modello (nero) risulta essere quello migliore secondo BIC, AIC, e significatività dei parametri. Il grafico seguente illustra le nuove distribuzioni ottenute adattando quella delle valutazioni prese da Steam.

```
ggplot( df ) +
  geom_density(aes(x=steam_user_score), col="blue", fill="blue", linetype = "dashed", alpha=0.1) +
  geom_density(aes(x=user_score), col="red", fill="red", linetype = "dashed", alpha=0.1) +
  geom_density(aes(x=map_dbl(steam_user_score, f1)), col="orange", alpha=0.1) +
  geom_density(aes(x=map_dbl(steam_user_score, f2)), col="black", alpha=0.1) +
  geom_density(aes(x=map_dbl(steam_user_score, f3)), col="magenta", alpha=0.1) +
  labs(title = "Adattamento valutazioni e voti") +
  ylab("frequenze") +
  xlab("voto/valutazione")
```



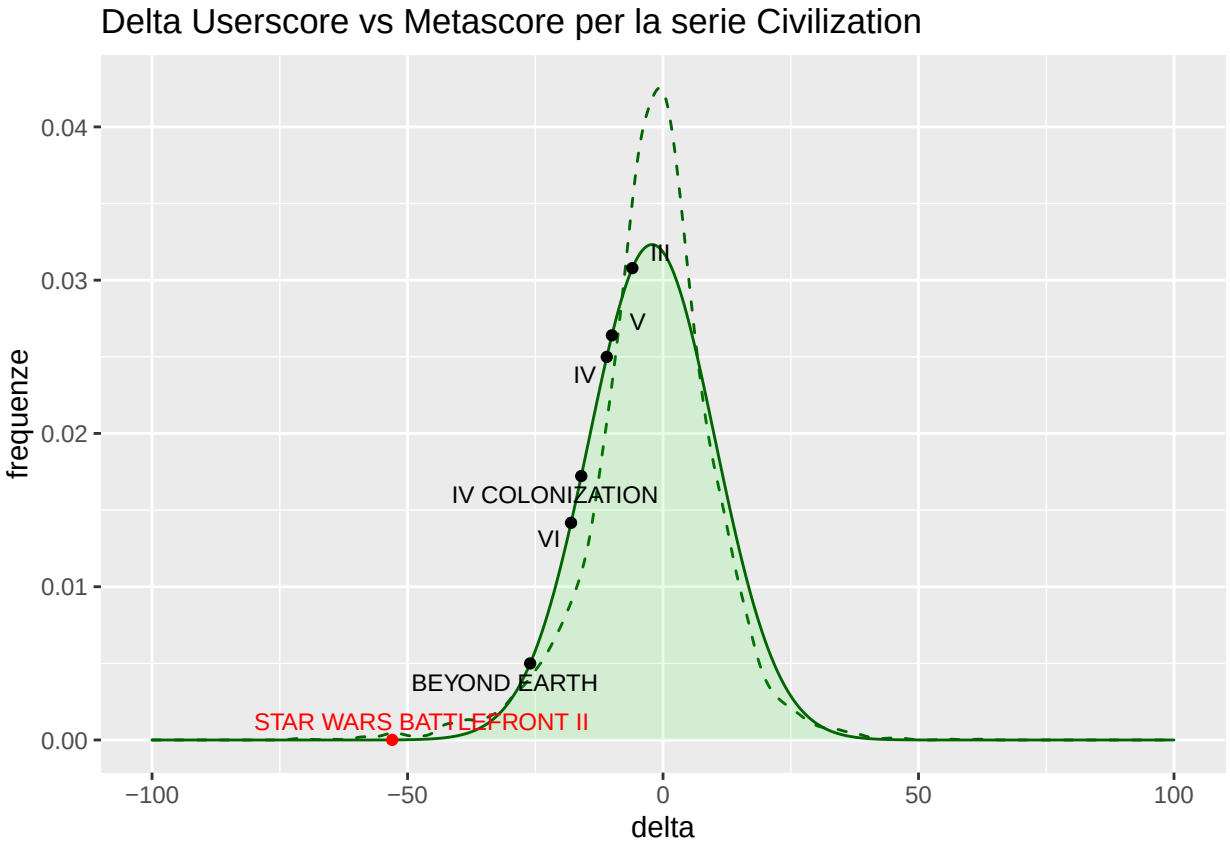
Queste nuove distribuzioni saranno utili successivamente per valutare in modo attendibile i giochi che non dispongono di un voto direttamente presente sul portale di Metacritic.

Osservare che le distribuzioni valutazioni degli utenti Metacritic e quelle del metascore sono molto simili è molto interessante. Calcolare un punteggio a partire dal solo rapporto apprezzamenti/valutazioni sembra invece sovrastimare abbondantemente la valutazione. Procediamo quindi infine a vedere dove si posizionano i giochi della serie "Civilization" rispetto alle distribuzioni empiriche calcolate per i delta di valutazione. Includiamo in questa fase anche un caso noto e decisamente marcato di "review bombing", quello del gioco "Star Wars Battlefront II":

```
md.no.NA %>% string.query(name,"battlefront ii")
```

```
## # A tibble: 1 x 15
## # Groups:   name [1]
##   name  genre.s. players metascore user_score release_date critic_positive
##   <chr> <fct>    <fct>       <int>      <dbl> <chr>                  <int>
## 1 star~ Action   ""             65         12 17-11-17                   6
## # ... with 8 more variables: critic_neutral <int>, critic_negative <int>,
## #   user_positive <int>, user_neutral <int>, user_negative <int>,
## #   critic_total <int>, user_total <int>, rating_metacritic <fct>
```

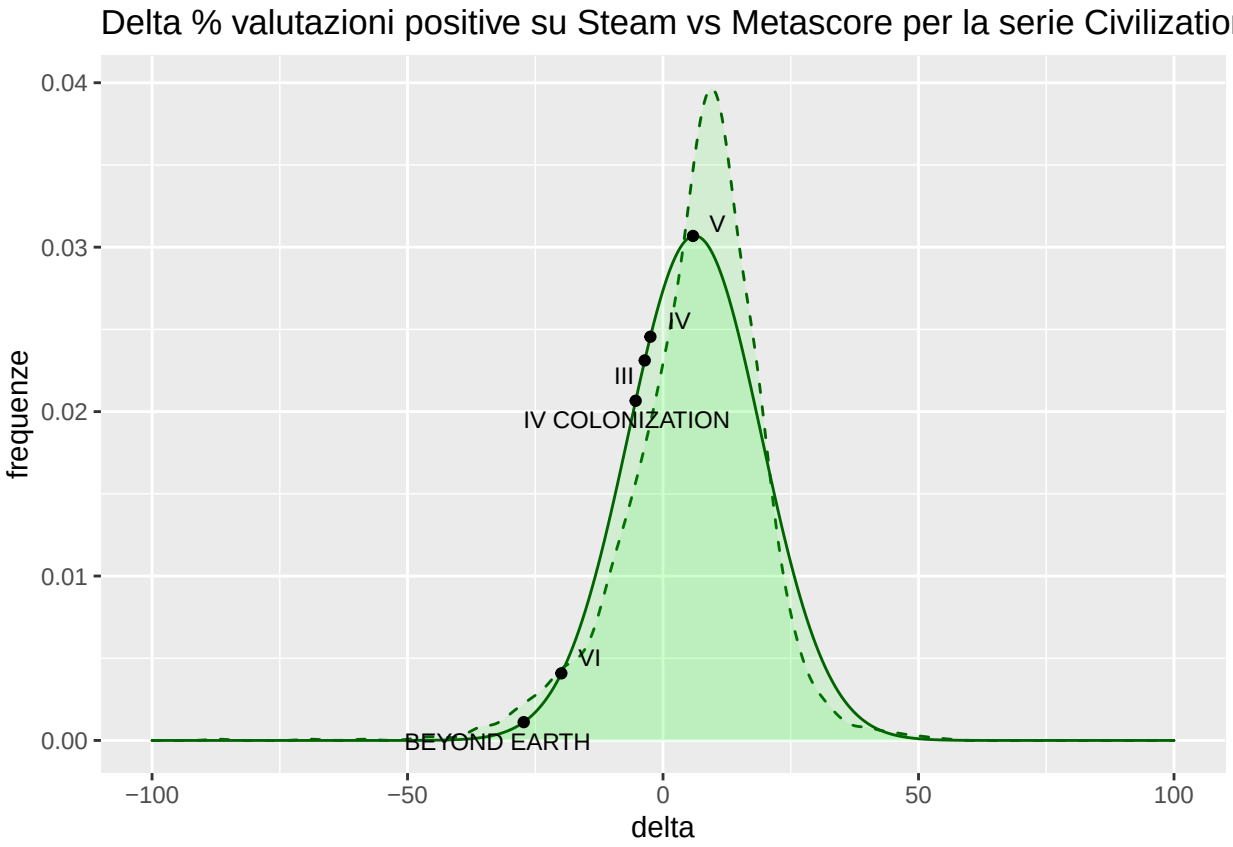
```
ggplot(md.no.NA) +
  geom_density(aes(x=user_score-metascore), col="darkgreen", linetype = "dashed") +
  geom_area( data = data.frame(x=seq(-100,100,0.1), y=dnorm(seq(-100,100,0.1),
                mean=mean(md.no.NA$user_score-md.no.NA$metascore),
                sd=sd(md.no.NA$user_score-md.no.NA$metascore))),
             aes(x=x,y=y), col="darkgreen", fill="green", alpha = 0.1) +
  geom_point(data = played.civs, aes(x=user_score-metascore,
              y=dnorm(user_score-metascore,
                 mean=mean(md.no.NA$user_score-md.no.NA$metascore),
                 sd=sd(md.no.NA$user_score-md.no.NA$metascore)))) +
  geom_node_text(data = played.civs, aes(x=user_score-metascore,
      y=dnorm(user_score-metascore,
        mean=mean(md.no.NA$user_score-md.no.NA$metascore),
        sd=sd(md.no.NA$user_score-md.no.NA$metascore)),
      label = toupper(gsub("sid meiers civilization","",name))), color = "black", size = 3, repel=TRUE )
  geom_point(data = md.no.NA %>% string.query(name,"battlefront ii"), aes(x=user_score-metascore,
              y=dnorm(user_score-metascore,
                 mean=mean(md.no.NA$user_score-md.no.NA$metascore),
                 sd=sd(md.no.NA$user_score-md.no.NA$metascore))), col="red") +
  geom_node_text(data = md.no.NA %>% string.query(name,"battlefront ii"), aes(x=user_score-metascore,
      y=dnorm(user_score-metascore,
        mean=mean(md.no.NA$user_score-md.no.NA$metascore),
        sd=sd(md.no.NA$user_score-md.no.NA$metascore)),
      label = toupper(gsub("sid meiers civilization","",name))), color = "red", size = 3, repel=TRUE ) +
  labs(title = "Delta Userscore vs Metascore per la serie Civilization") +
  ylab("frequenze") +
  xlab("delta")
```



Da questo grafico possiamo concludere che effettivamente "Civilization VI" ha ricevuto molte valutazioni negative da parte degli utenti, anche se il gioco della serie che effettivamente ha subito il trattamento peggiore risulta essere "beyond earth". Calandoci nel contesto reale questo è effettivamente facilmente comprensibile: "beyond earth" è uno spinoff della serie principale con una visione prettamente fantascientifica, ben diversa da quella degli altri titoli della serie. E' evidente che un cambio così marcato non sia quindi stato apprezzato dagli appassionati che hanno quindi valutato negativamente il prodotto. Concludiamo mostrando l'effetto ancora più marcato avuto nel rapporto valutazioni positive/negative su Steam. (Non riportiamo dati su "Star Wars battlefront II" in quanto questo è un gioco EA e non era disponibile su Steam).

```
df <- md.no.NA %>% full_join(games.data,by=c("name")) %>% na.omit() %>%
      mutate(steam_user_score = 100*(positive_ratings)/(positive_ratings+negative_ratings))
df.civ <- played.civs %>%
      mutate(steam_user_score = 100*(positive_ratings)/(positive_ratings+negative_ratings))
ggplot( df ) +
  geom_density(aes(x=steam_user_score-metascore), col="darkgreen", fill="green", linetype = "dashed", al
  geom_area( data = data.frame(x=seq(-100,100,0.1),
                 y=dnorm(seq(-100,100,0.1),
                   mean=mean(df$steam_user_score-df$metascore),
                   sd=sd(df$steam_user_score-df$metascore))),
             aes(x=x,y=y), col="darkgreen", fill="green", alpha = 0.1) +
  geom_point(data = df.civ, aes(x=steam_user_score-metascore,
                 y=dnorm(steam_user_score-metascore,
                    mean=mean(df$steam_user_score-df$metascore),
                    sd=sd(df$steam_user_score-df$metascore)))) +
  geom_node_text(data = df.civ, aes(x=steam_user_score-metascore,
                 y=dnorm(steam_user_score-metascore,
                    mean=mean(df$steam_user_score-df$metascore),
                    sd=sd(df$steam_user_score-df$metascore)),
```

```
    label = toupper(gsub("sid meiers civilization","",name))), color = "black", size = 3, repel=TRUE )
  labs(title = "Delta % valutazioni positive su Steam vs Metascore per la serie Civilization") +
  ylab("frequenze") +
  xlab("delta")
```



La distribuzione degll percentuali di valutazioni positive non è normale, qui viene comunque approssimata così per visualizzare più chiaramente la problematica

> "Star Wars Battlefront II" è stato ampliamente criticato dagli utenti in quanto il gioco proponeva una enorme quantità di "microtransazioni", in quel periodo infatti EA (non da sola) stava spingendo per introdurre delle metodologie legate ai software come servizio (SaS) all'interno dei propri giochi in quanto decisamente remunerative. Questo ha creato gradualmente una situazione di malcontento nei giocatori che è letteralmente esplosa al lancio di questo gioco. Questa, per così dire, ribellione ha portato a un ridimensionamento notevole della quantità di microtransazioni nei giochi successivi. Ovviamente la situazione è molto complessa e articolata e necessiterebbe senza dubbio di una trattazione molto più esaustiva che esula dalle finalità di questa relazione.

Nota sull'incompletezza unendo i dati di gioco con quelli di metacritic

Calcoliamo quanto impatta accettare solamente dati che abbiano disponibili tutti i campi offerti da Metacritic:

```
# giocatori iniziali
number.of.classes(play.data, player)
```

```
## [1] 10550
```

```
# giochi iniziali
number.of.classes(play.data, name)
```

```
## [1] 2516
```

```
# delta giocatori finali
number.of.classes(play.data, player) - number.of.classes(play.data %>% full_join(md.no.NA,by=c("name"))
```

```
## [1] 817
```

```
# delta giochi finali
number.of.classes(play.data, name) - number.of.classes(play.data %>% full_join(md.no.NA,by=c("name")) %
```

```
## [1] 1321
```

Si può certamente fare di meglio tenendo conto che non tutti i giochi hanno abbastanza recensioni utente o specializzate da ottenere un metascore. Utilizzeremo quindi queste informazioni solo quando sono effettivamente disponibili, nei restanti casi operiamo sui giochi e giocatori già selezionati e stimiamo le valutazioni utilizzando il modello presentato in precedenza.

RIMUOVI!!

```
metacritic.data %>% full_join(games.data,by=c("name")) %>%
      mutate(steam_user_score = 100*(positive_ratings)/(positive_ratings+negative_ratings)) %>%
  select(name, metascore, user_score, genres, categories, steamspy_tags) %>% unnest(genres) %>% filter(
```

```
## # A tibble: 10 x 6
## # Groups:   name [10]
##    name                      metascore user_score genres categories steamspy_tags
##    <chr>                         <int>      <dbl> <chr>  <list>     <list>
##  1 the witcher 3 wild hunt          93         94 RPG    <chr [5]>  <chr [3]>
##  2 system shock 2                   92         91 RPG    <chr [3]>  <chr [3]>
##  3 star wars knights of th~         93         90 RPG    <chr [2]>  <chr [3]>
##  4 final fantasy ix                 84         89 RPG    <chr [5]>  <chr [3]>
##  5 to the moon                      81         89 RPG    <chr [5]>  <chr [3]>
##  6 gleaner heights                  65         88 RPG    <chr [7]>  <chr [3]>
##  7 herou rogue to redempti~         80         88 RPG    <chr [2]>  <chr [3]>
##  8 lisa                             78         88 RPG    <chr [5]>  <chr [3]>
##  9 mass effect 2                    94         88 RPG    <chr [1]>  <chr [3]>
## 10 mount & blade warband            78         88 RPG    <chr [5]>  <chr [3]>
```

Reti delle associazioni gioco-giocatore

In questa sezione prenderemo in considerazione le reti descritte dalle relazioni di gioco e di acquisto, che sono direttamente rappresentate dal dataset 200k.

```
totalTime.game <- play.data %>%
    group_by(name) %>%
    summarise(totalGameTime = sum(time)) %>%
    select(name, totalGameTime) %>% arrange(desc(totalGameTime))

totalTime.player <- play.data %>%
    group_by(player) %>%
    summarise(totalGameTime = sum(time)) %>%
    select(player, totalGameTime) %>% arrange(desc(totalGameTime))

totalTime.game %>% head(10)
```

```
## # A tibble: 10 x 2
##    name                          totalGameTime
```

```
##    <chr>                                <dbl>
##  1 dota 2                              981685.
##  2 counterstrike global offensive      322772.
##  3 team fortress 2                     173673.
##  4 counterstrike                       134261.
##  5 sid meiers civilization v            99821.
##  6 counterstrike source                 96076.
##  7 the elder scrolls v skyrim           70889.
##  8 garrys mod                           49725.
##  9 left 4 dead 2                        33597.
## 10 terraria                             29952.
```

```
totalTime.player %>% head(10)
```

```
## # A tibble: 10 x 2
##       player totalGameTime
##        <int>         <dbl>
##  1  73017395        11754
##  2 100630947        10851.
##  3 153382649         9640
##  4  26762388         8465.
##  5  10599862         8267.
##  6  48798067         8194.
##  7  52731290         8151.
##  8  14544587         7919.
##  9  52567955         7836.
## 10 130882834         7801
```

```
games <- play.data %>% select(name) %>% arrange(name) %>% unique() %>% mutate(is_game = TRUE) %>%
  inner_join(games.data, by=c("name")) %>% inner_join(totalTime.game, by=c("name")) %>%
  arrange(name, release_date) %>% group_by(name) %>% slice(1)

players <- play.data %>% select(player) %>% arrange(player) %>% unique() %>%
  mutate(is_game = FALSE) %>% mutate(name = player) %>% select(-player) %>%
  full_join(totalTime.player %>% mutate(name=player) %>% select(-player), by=c("name")) %>%
  mutate(name = paste("__",name,"__") )

play.relation <- play.data %>% mutate(player = paste("__",player,"__") ) %>%
  mutate(from=name, to=player) %>%
  semi_join(rbind.fill(games,players), by=c("name")) %>% select(from, to, time)

players.games.graph <- graph_from_data_frame(play.relation, vertices=rbind.fill(games,players), directe
  as_tbl_graph() %>%
  mutate(centrality = centrality_authority())

#players.games.graph

# verifico se il grafo è bipartito (come dovrebbe essere)
as.data.frame(get.edgelist(players.games.graph)) %>%
  full_join(games %>% select(name,is_game),by=c("V1" = "name") ) %>%
  full_join(players %>% select(name,is_game),by=c("V2" = "name") ) %>% filter(is_game.x == is_game.y)
```

```
## Warning: Column `V1`/`name` joining factor and character vector, coercing into
## character vector
```

```
## Warning: Column `V2`/`name` joining factor and character vector, coercing into
```

```
## character vector
```

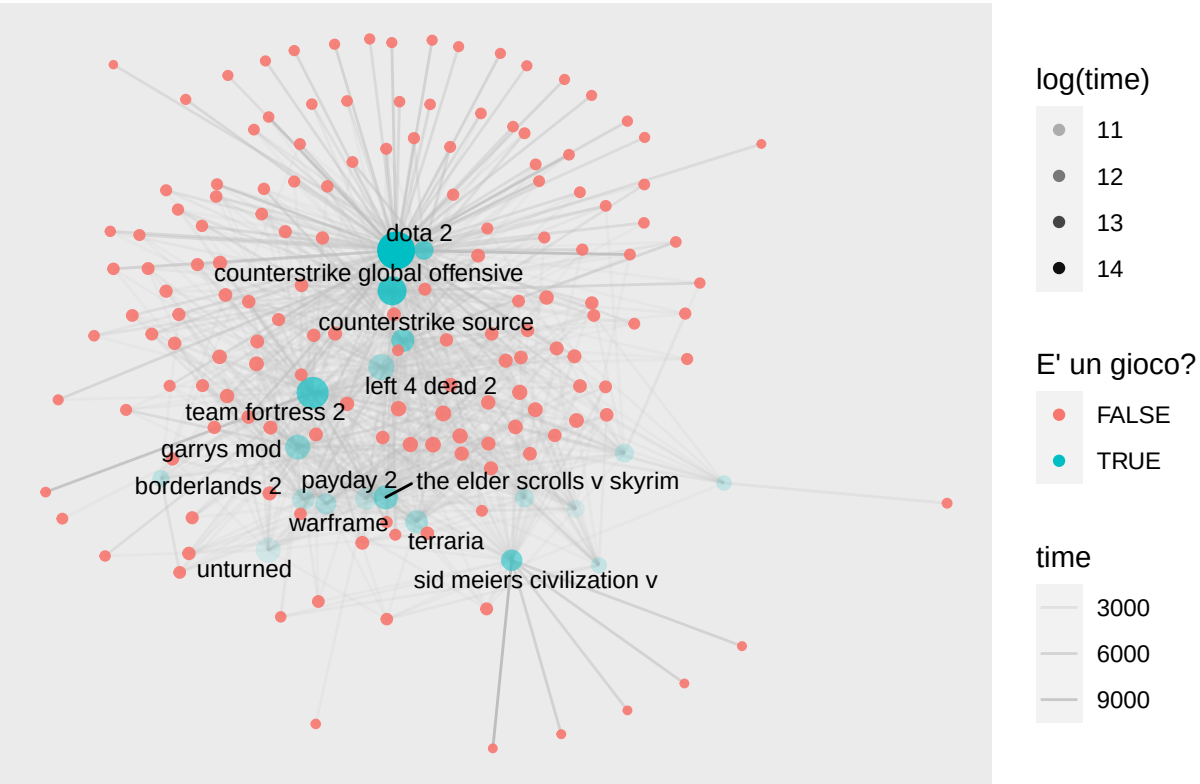
```
## [1] V1        V2        is_game.x is_game.y
## <0 rows> (or 0-length row.names)
```

```
# questo dataframe deve essere vuoto
```

```
players.games.graph.subgraph <- to_subgraph(players.games.graph, ifelse(is_game, totalGameTime > 15000,

#players.games.graph.subgraph
```

```
ggraph(players.games.graph.subgraph, layout = 'kk') +
  geom_edge_link(aes(alpha=time), color="gray") +
  geom_node_point(aes(col=is_game, size=centrality, alpha=ifelse(is_game,log(totalGameTime)+0.5,max(log
  geom_node_text( aes( filter = centrality > 0.15, label = name), color = "black", size = 3, repel=TRUE
  labs(title = "Rete giochi (>15000 ore di gioco) giocatori (>4000 ore di gioco)",
       color = "E' un gioco?", alpha="log(time)") +
  scale_size(guide="none")
```



dove la dimensione dei nodi è data dalla centralità. Valutiamo quindi alcune delle proprietà di questo grafo. Iniziamo dalla distribuzione dei gradi:

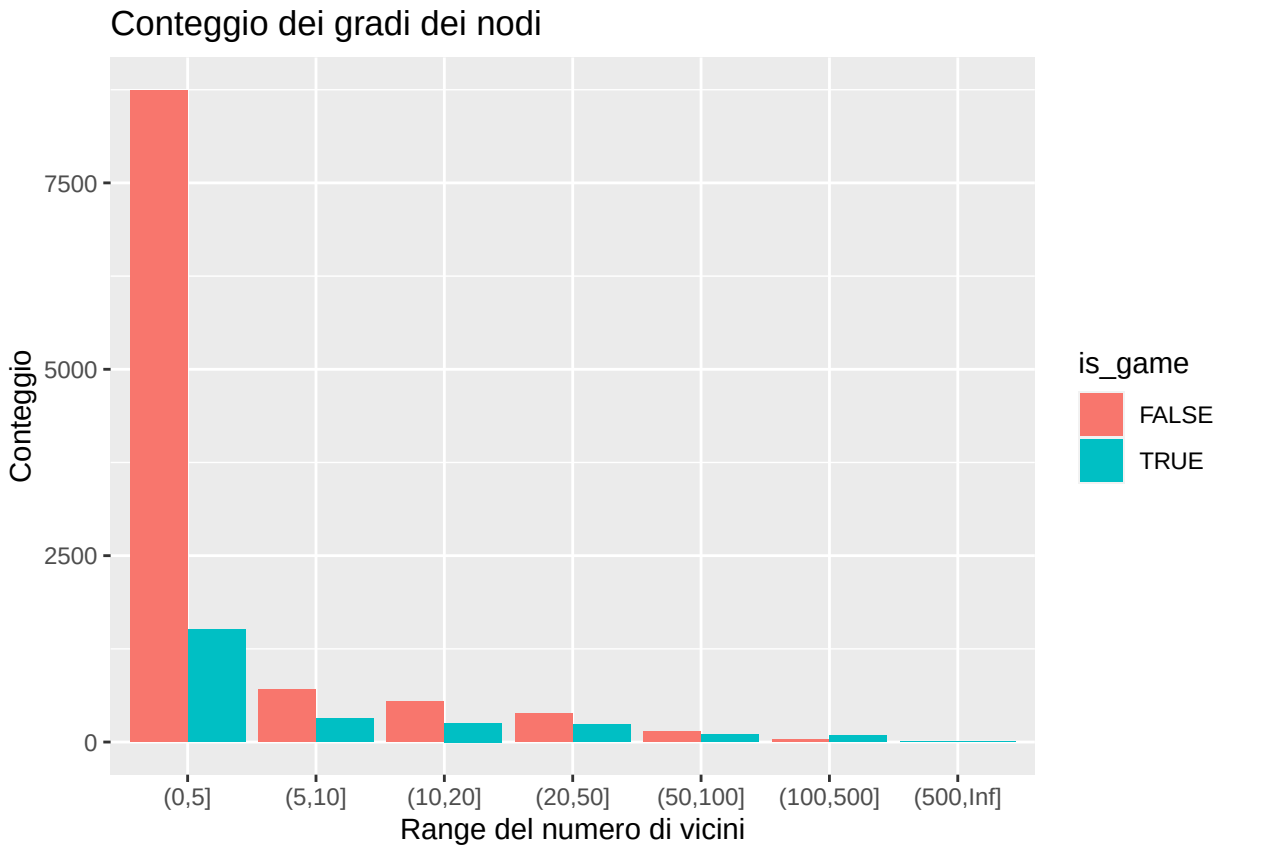
```
players.games.graph.degrees <- as.data.frame(degree(players.games.graph)) %>% mutate(name = names(degre
  rename(degree = "degree(players.games.graph)") %>% join(as.data.frame(players.games.graph) %>% select

players.games.graph.degrees %>% head(10)
```

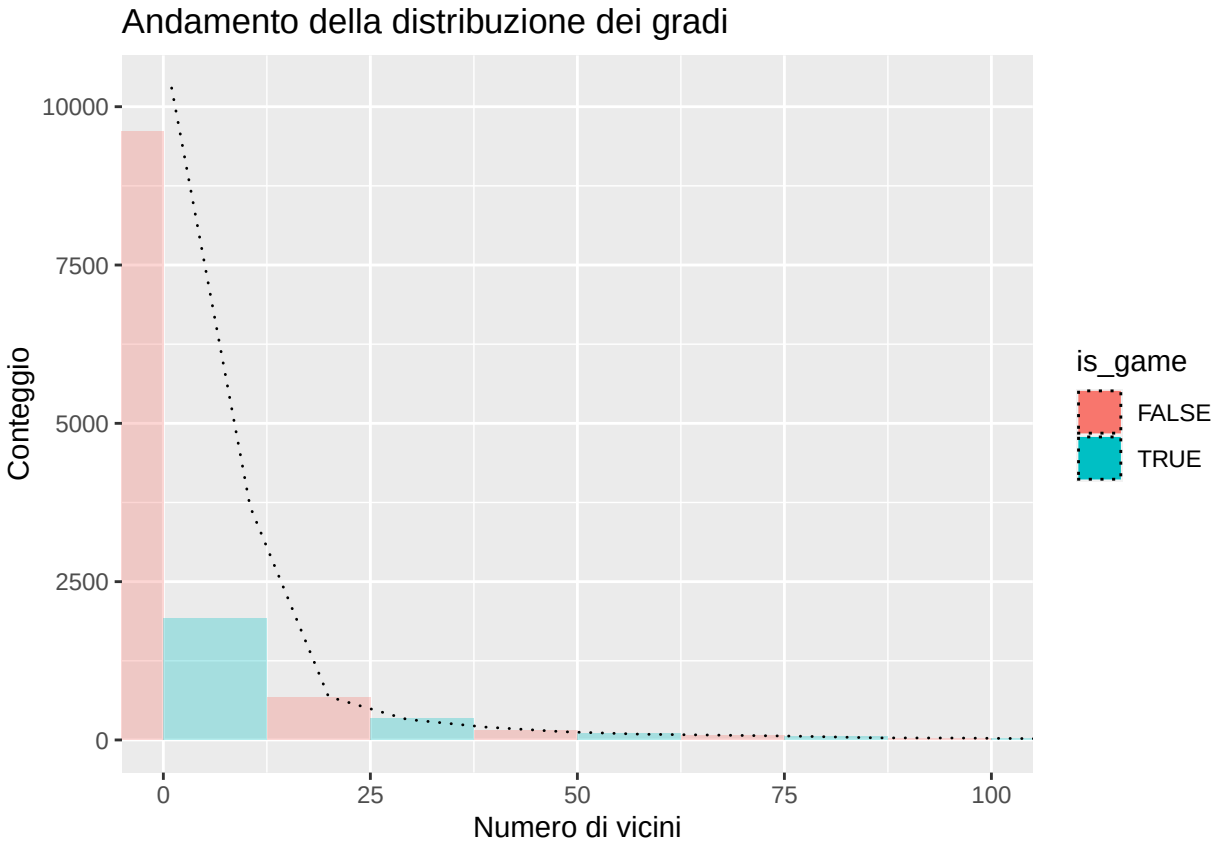
```
##    degree                                name is_game
## 1       3                            Orbitalis    TRUE
## 2       2                       10 second ninja    TRUE
```

```
## 3       1                                10,000,000    TRUE
## 4       9                          100% orange juice    TRUE
## 5       1                                1000 amps    TRUE
## 6       7                     12 labours of hercules    TRUE
## 7       8 12 labours of hercules ii the cretan bull    TRUE
## 8       3     12 labours of hercules iii girl power    TRUE
## 9       5                                       140    TRUE
## 10      1                                 15 days    TRUE
```

```
ggplot(players.games.graph.degrees %>% factorise(degree, c(0,5,10,20,50,100,500,Inf))) +
  geom_bar(aes(x=degree, fill=is_game), position = "dodge") +
  labs(title = "Conteggio dei gradi dei nodi") +
  ylab("Conteggio") +
  xlab("Range del numero di vicini")
```




```
ggplot(players.games.graph.degrees) +
  geom_histogram(aes(x=degree, fill=is_game), binwidth = 25, alpha=0.3, position="dodge") +
  geom_density(aes(x=degree, y=..count..), linetype = "dotted") +
  coord_cartesian(xlim=c(0,100)) +
  labs(title = "Andamento della distribuzione dei gradi") +
  ylab("Conteggio") +
  xlab("Numero di vicini")
```



Questa distribuzione dei gradi segue quella di una distribuzione a coda lunga, possiamo quindi concludere che questo grafo segue il modello a *collegamento preferenziale* di Barabasi-Albert. Questa ipotesi viene confermata anche dall'istogramma delle distanze tra i nodi:

```
# commentato perchè ci impiega qualche minuto
#df <- as.data.frame(as.table(players.games.graph %>% distances())) %>% filter(Freq != Inf & Freq != 0)
#ggplot(df %>% factorise(Freq, c(0,1,2,3,4,5,10,Inf))) +
#  geom_bar(aes(x=Freq)) +
#  labs(title = "Valutazione delle distanze tra i nodi") +
#  ylab("Conteggio") +
#  xlab("Distanza")
```

Possiamo facilmente vedere, come intuibile, che i giochi sono più centrali, secondo il criterio di autorità, rispetto ai giocatori:

```
players.games.graph %>% arrange(desc(centrality)) %>% select(name,centrality,is_game) %>% as.data.frame
```

```
##                             name centrality is_game
## 1                       dota 2  1.0000000    TRUE
## 2                team fortress 2  0.6292968    TRUE
## 3  counterstrike global offensive  0.4677942    TRUE
## 4                  left 4 dead 2  0.3431805    TRUE
## 5                     garrys mod  0.2959488    TRUE
## 6                       unturned  0.2938680    TRUE
## 7     the elder scrolls v skyrim  0.2721326    TRUE
## 8           counterstrike source  0.2269820    TRUE
## 9                       terraria  0.2269631    TRUE
## 10                      portal 2  0.2230797    TRUE
```

```
players.games.graph %>% filter(is_game == FALSE) %>% arrange(desc(centrality)) %>% select(name,centrali
```

```
##              name centrality is_game
## 1   __ 49893565 __ 0.05161991   FALSE
## 2   __ 11403772 __ 0.04725789   FALSE
## 3     __ 298950 __ 0.04472673   FALSE
## 4   __ 36546868 __ 0.04262805   FALSE
## 5   __ 62990992 __ 0.04220094   FALSE
## 6   __ 55906572 __ 0.04206402   FALSE
## 7   __ 48798067 __ 0.04164756   FALSE
## 8   __ 17530772 __ 0.03851630   FALSE
## 9   __ 86055705 __ 0.03798089   FALSE
## 10  __ 51557405 __ 0.03757577   FALSE
```

Ovviamente questo è dovuto principalmente al fatto che i giocatori sono molti di più dei giochi e che questo è un grafo bipartito, che ammette quindi solo relazioni giochi-giocatori. Valutiamo ora la presenza o meno della classica *componente gigante* che viene a formarsi nei grafi non direatti a *collegamento preferenziale* come questo:

```
ncc <- count_components(players.games.graph)
ncc
```

```
## [1] 16
```

```
component_distribution(players.games.graph) %>% imap(~ c(.x*ncc,.y) ) %>% keep(~ (.[1] != 0) ) %>% map(
```

```
## [[1]]
## [1] "Dimensione: 3 Numero: 14"
##
## [[2]]
## [1] "Dimensione: 4 Numero: 1"
##
## [[3]]
## [1] "Dimensione: 13036 Numero: 1"
```

La componente gigante quindi è presente e raccoglie pressochè tutti i nodi presenti nella rete.

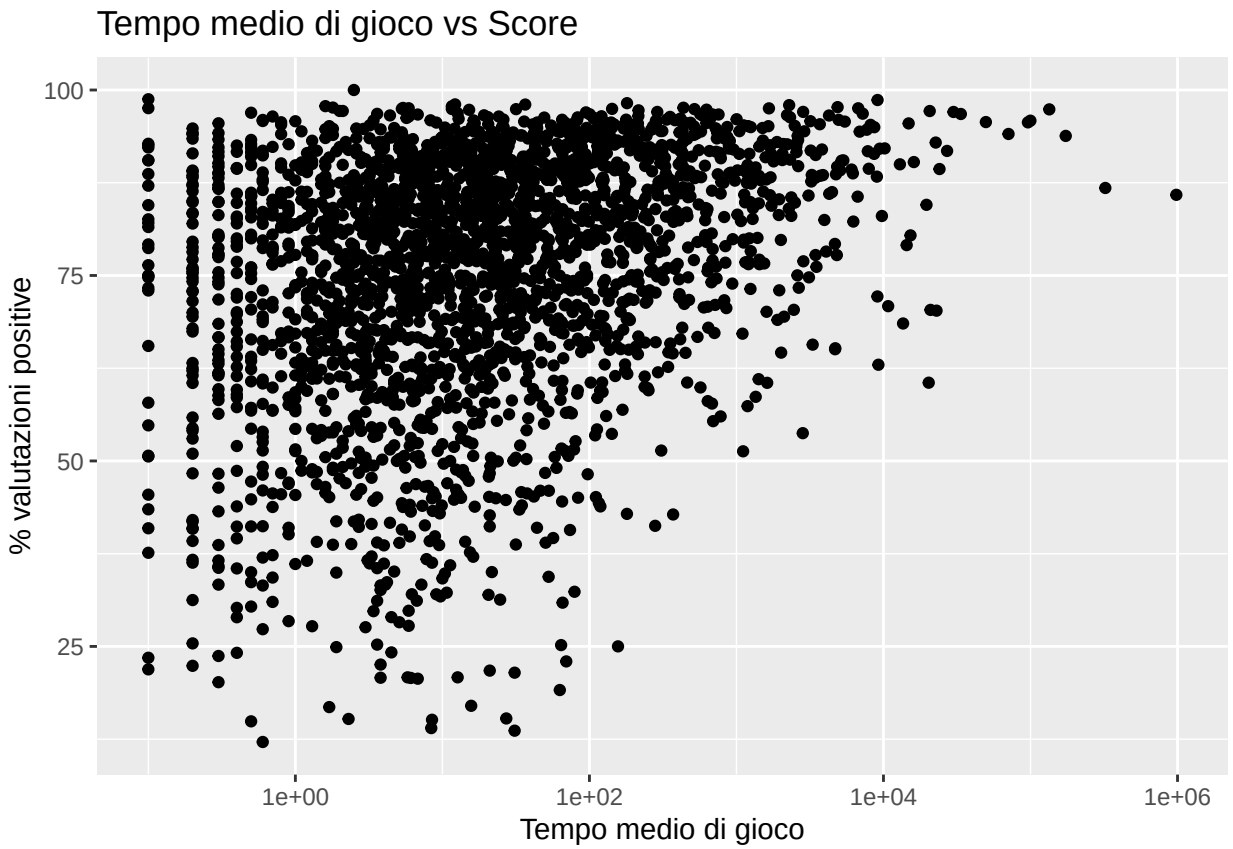
Possiamo chiederci se i giochi più giocati siano anche quelli più graditi:

```
df <- players.games.graph.degrees %>% inner_join(as.data.frame(players.games.graph)) %>%
  mutate(score = 100*(positive_ratings)/(positive_ratings+negative_ratings)) %>% select(degree,name,tota
```

```
## Joining, by = c("name", "is_game")
```

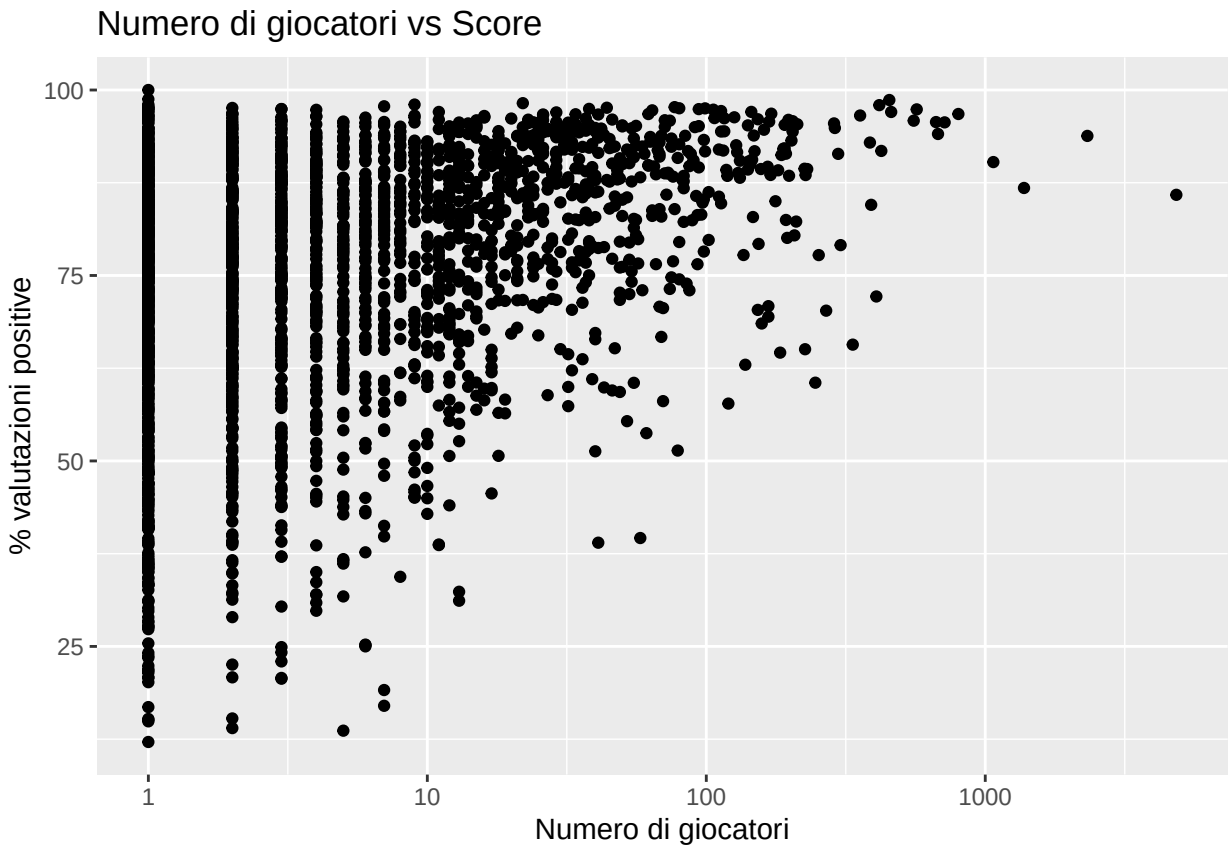
```
# i giocatori sono eliminati automaticamente
ggplot(df) +
  geom_node_point(aes(x=totalGameTime, y=score)) +
  scale_x_log10() +
  labs(title = "Tempo medio di gioco vs Score") +
  ylab("% valutazioni positive") +
  xlab("Tempo medio di gioco")
```

```
## Warning: Removed 10550 rows containing missing values (geom_point).
```



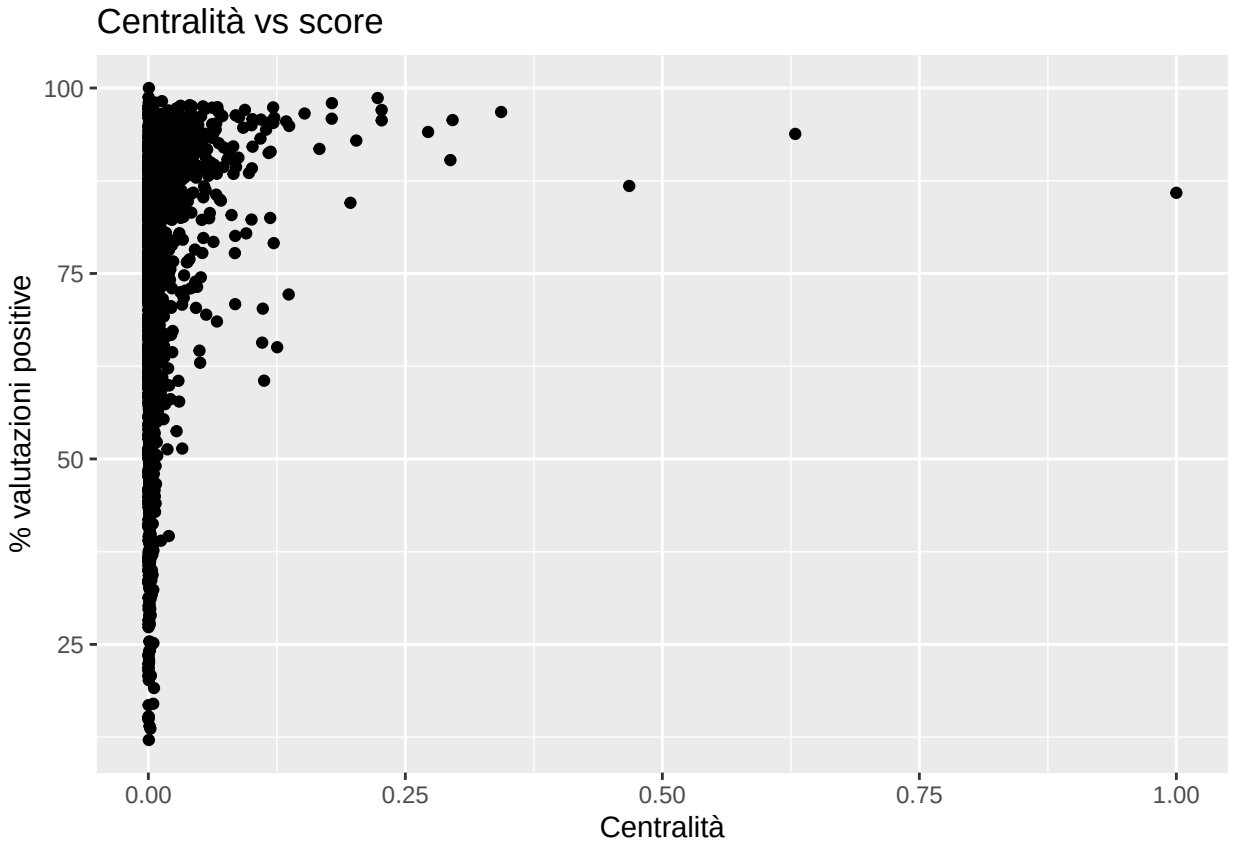

```
ggplot(df) +
  geom_node_point(aes(x=degree, y=score)) +
  scale_x_log10() +
  labs(title = "Numero di giocatori vs Score") +
  ylab("% valutazioni positive") +
  xlab("Numero di giocatori")
```

```
## Warning: Removed 10550 rows containing missing values (geom_point).
```



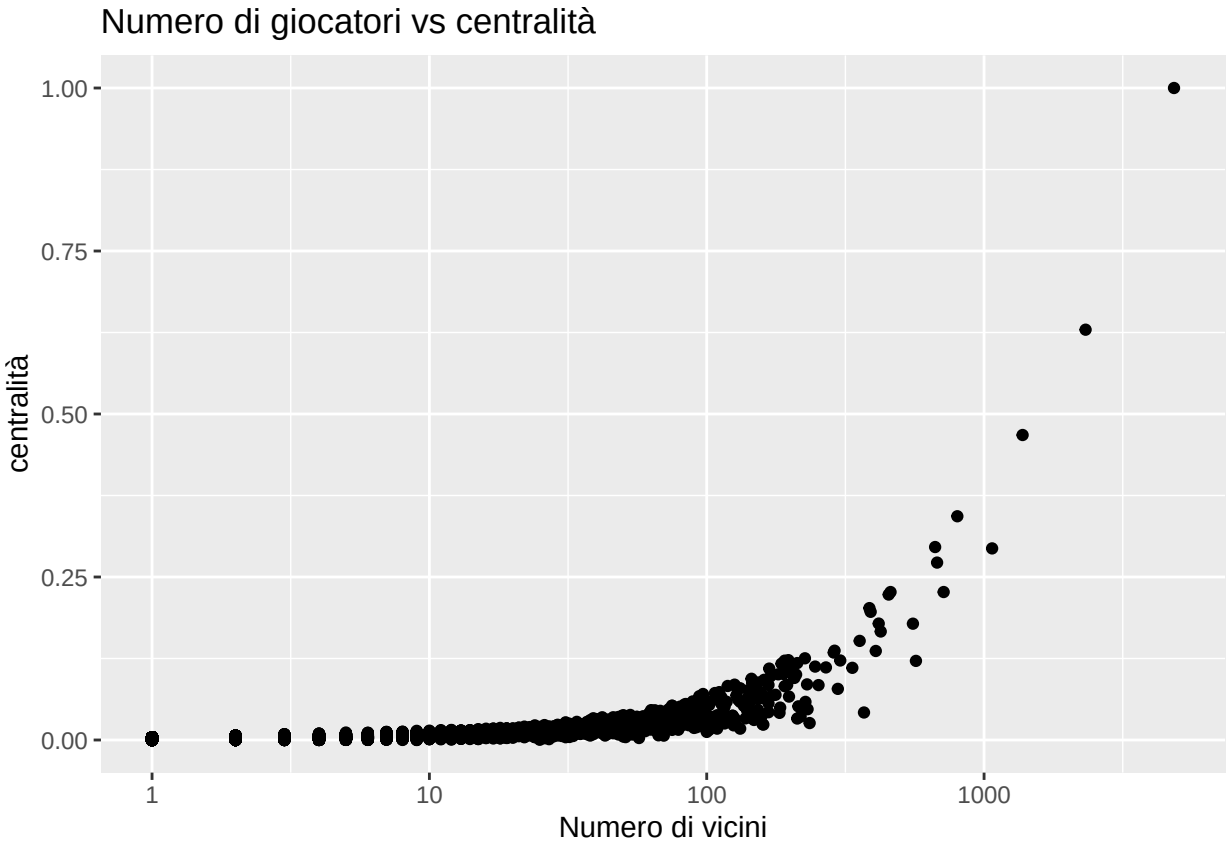

```
ggplot(df) +
  geom_node_point(aes(x=centrality, y=score)) +
  labs(title = "Centralità vs score") +
  xlab("Centralità") +
  ylab("% valutazioni positive")
```

```
## Warning: Removed 10550 rows containing missing values (geom_point).
```



Si può osservare quindi come, se accade che un gioco si affermi, ossia che abbia un alto tempo di gioco, livello di autorità o numero di giocatori, allora difficilmente questo avrà un basso score.

```
ggplot(df) +
  geom_node_point(aes(x=degree, y=centrality)) +
  scale_x_log10() +
  labs(title = "Numero di giocatori vs centralità") +
  ylab("centralità") +
  xlab("Numero di vicini")
```



Ovviamente, data la formula per il calcolo dell'autorità, questa correla fortemente con il numero di nodi adiacenti.

Grafo della userbase condivisa

Più maneggevole e interessante per il nostro scopo di creare un sistema per consigliare nuovi giochi da giocare è il *grafo della userbase condivisa*, ossia il grafo pesato che mostra per ogni coppia di giochi il numero di giocatori che li giocano entrambi. Questo grafo è particolarmente significativo perchè mette in relazione diretta i giochi senza dover tener più direttamente traccia degli utenti.

```
reduced <- play.data %>% select(name,player,time)

game.game.df <- reduced %>% inner_join(reduced, by=c("player")) %>% select(name.x, name.y, time.x) %>% 
  group_by(name.x,name.y) %>% summarise(absTime = sum(time.x), avgTime = sum(time.x)/n(),
                                        avgTimeCorrected = (sum(time.x))/(n()+5), sharedUserbase = n())
  arrange(desc(sharedUserbase))

game.game.relation <- game.game.df %>% rename(from = name.x, to = name.y)

# uso "games" che contiene i nomi dei giochi
game.game.graph <- graph_from_data_frame(game.game.relation, vertices=games, directed=TRUE) %>%
  as_tbl_graph() %>%
  mutate(centrality = centrality_authority())

#game.game.graph
```

RIMUOVI!!

```
game.game.graph %>% as.data.frame() %>% filter.by.tag.or(genres, c("RPG")) %>% arrange(desc(centrality))
```

```
##                                                name is_game  appid
## 1                           the elder scrolls v skyrim    TRUE  72850
## 2                                              terraria    TRUE 105600
## 3                                         borderlands 2    TRUE  49520
## 4                                               magicka    TRUE  42910
## 5  the witcher 2 assassins of kings enhanced edition    TRUE  20920
## 6                                          torchlight ii    TRUE 200710
## 7                                 the binding of isaac    TRUE 113200
## 8                                    fallout new vegas    TRUE  22380
## 9                                    dungeon defenders    TRUE  65800
## 10                                          torchlight    TRUE  41500
##    release_date english                                    developer
## 1         15288       1                      Bethesda Game Studios
## 2         15110       1                                  Re-Logic
## 3         15603       1 Gearbox Software;Aspyr (Mac);Aspyr (Linux)
## 4         14999       1                      Arrowhead Game Studios
## 5         15446       1                              CD PROJEKT RED
## 6         15603       1                                 Runic Games
## 7         15245       1           Edmund McMillen and Florian Himsl
## 8         14903       1                      Obsidian Entertainment
## 9         15265       1                        Trendy Entertainment
## 10        14544       1                                 Runic Games
##                       publisher           platforms required_age
## 1            Bethesda Softworks             windows           16
## 2                     Re-Logic windows, mac, linux            0
## 3  2K;Aspyr (Mac);Aspyr (Linux) windows, mac, linux           18
## 4           Paradox Interactive             windows            0
## 5    CD PROJEKT RED;1C-SoftClub windows, mac, linux            0
## 6                   Runic Games windows, mac, linux            0
## 7               Edmund McMillen        windows, mac            0
## 8            Bethesda Softworks             windows           18
## 9          Trendy Entertainment windows, mac, linux            0
## 10                  Runic Games        windows, mac            0
##
## 1                                                                Single-player, Steam Achievements
## 2                             Single-player, Multi-player, Online Multi-Player, Co-op, Online Co-op,
## 3                                                                      Single-player, Co-op,
## 4                                   Single-player, Multi-player, Co-op, Shared/Split Screen, Steam
## 5                                                  Single-player, Steam Achievements, Ste
## 6                                       Single-player, Multi-player, Co-op, Cross-Platform Mul
## 7
## 8                                                                                              Sing
## 9  Single-player, Multi-player, Co-op, Shared/Split Screen, Steam Achievements, Full controller suppo
## 10
##                           genres                    steamspy_tags achievements
## 1                            RPG          Open World, RPG, Fantasy           75
## 2  Action, Adventure, Indie, RPG      Sandbox, Adventure, Survival           88
## 3                    Action, RPG                  FPS, Co-op, RPG           69
## 4                    Action, RPG             Comedy, Action, Co-op           88
## 5                            RPG              RPG, Fantasy, Mature           52
## 6  Action, Adventure, Indie, RPG   RPG, Action RPG, Hack and Slash          119
## 7  Action, Adventure, Indie, RPG   Rogue-like, Indie, Replay Value           99
```

```
## 8                    Action, RPG Open World, RPG, Post-apocalyptic          75
## 9   Action, Indie, RPG, Strategy         Tower Defense, RPG, Co-op         114
## 10                           RPG   RPG, Action RPG, Hack and Slash          66
##    positive_ratings negative_ratings average_playtime median_playtime
## 1            237303            14951             7089            3885
## 2            255600             7797             5585            1840
## 3            144595            11021             3276            1139
## 4             20912             1964              589             369
## 5             36427             4710              838             320
## 6             29539             1757             1037             508
## 7             43227             1923             1849             567
## 8             66756             3149             2485            1431
## 9             10693              932             1993            1205
## 10             3825              317             1108            1144
##    owners_lwb owners_upb price totalGameTime centrality
## 1    10000000   20000000  9.99       70889.3  1.0000000
## 2     5000000   10000000  6.99       29951.8  0.9879322
## 3     5000000   10000000 19.99       22667.9  0.9753134
## 4     2000000    5000000  7.99        2053.8  0.9625058
## 5     2000000    5000000 14.99        3254.3  0.9378128
## 6     2000000    5000000 14.99        6891.9  0.9259551
## 7     2000000    5000000  3.99        5458.1  0.8919096
## 8     2000000    5000000  7.99       14832.9  0.8884692
## 9     1000000    2000000  9.99        3816.2  0.8694643
## 10    1000000    2000000 10.99        1688.9  0.8632280
```

```
game.game.graph.subgraph <- to_subgraph(game.game.graph, sharedUserbase >= 100, subset_by = "edges")$sub
V(game.game.graph.subgraph)$degree = unlist(as.data.frame(degree(game.game.graph.subgraph)))
game.game.graph.subgraph <- to_subgraph(game.game.graph.subgraph, degree > 0, subset_by = "nodes")$subgr

#game.game.graph.subgraph

ggraph(game.game.graph.subgraph, layout="kk") +
  geom_edge_link(aes(alpha=sharedUserbase), color="darkgrey") +
  geom_node_point(aes(size = centrality, color=log(totalGameTime))) +
  geom_node_text( aes( filter = centrality > 0.97, label = name), color = "black", size = 3, repel=TRUE
  labs(title = "Rete giochi/giochi per utenza condivisa")
```

Rete giochi/giochi per utenza condivisa