Analisi statistica dei tweet francofoni riguardanti la guerra Russo-Ucraina

Corso: Statistica e Analisi Dati

Studenti:
Rosario Pio Gnazzo, Luca Pauzano
Università degli Studi di Salerno

a.s. 2024/2025

February 13, 2025

Contents

1 Introduzione **3**

2 Analisi Dataset Tweet **4**

2.1 Distribuzione dei dati 6

2.1.1 Trasformazione logaritmica 7

2.1.2 Identificazione ed eliminazione degli outliers 8

2.1.3 Verifica della Normalità 9

2.2 Relazioni lineari fra le variabili 11

2.3 Regressioni Lineari 13

2.3.1 Regressione lineare - y: Followers 13

2.3.2 Regressione lineare - y: Score 15

2.3.3 Regressione multinomiale - y: Sentiment 16

2.4 Analisi Temporale 18

2.4.1 Analisi della periodicità 18

2.4.2 Decomposizione di serie storiche 19

3 Time Series Forecast **24**

3.1 Analisi Statistico Temporale dello Score 24

3.1.1 Granularità di 10 minuti 26

3.1.2 Granularità di 30 minuti 27

3.1.3 Granularità di 1 ora 28

3.2 Non-seasonal ARIMA 30

3.2.1 Granularità di 10 minuti 31

3.2.2 Granularità di 30 minuti 32

3.2.3 Granularità di 1 ora 34

3.3 Auto-Regressive Neural Network 35

3.3.1 Granularità di 10 minuti 36

3.3.2 Granularità di 30 minuti 37

3.3.3 Granularità di 1 ora 38

3.4 Valutazione e Confronto dei modelli 39

3.4.1 Evaluation per le serie classiche 40

3.4.2 Evaluation per le serie delle derivate temporali 42

4 Dataset Sintetico **44**

4.1 Struttura del Prompt 44

4.2 Analisi descrittiva 45

4.2.1 Distribuzione dei dati 46

4.2.2 Gestione degli Outlier 47

4.2.3 Analisi della Normalità 49

4.2.4 Osserviamo le relazioni lineari nei dati 49

4.3 Esempio di regressione lineare 50

4.4 Conclusioni finali 51

1 Introduzione

L'obiettivo di questo lavoro è condurre un'**analisi statistica approfondita** sui tweet in lingua francese pubblicati tra il *24 febbraio 2023* e il *4 marzo 2023*, riguardanti la guerra Russo-Ucraina. Lo studio si concentrerà inizialmente sull'esplorazione delle variabili quantitative presenti nel dataset, con particolare attenzione all'individuazione di eventuali relazioni tra di esse. L'obiettivo è sviluppare **modelli statistici** capaci di **descrivere** e **spiegare** le **interazioni tra le variabili**, analizzando come si **influenzano a vicenda**.

Successivamente, l'attenzione si sposterà sull'**analisi temporale dei dati**. Non solo verrà descritto l'andamento dei tweet nel tempo, ma si cercherà anche di costruire modelli statistici in grado di rappresentare le serie storiche, valutando la possibilità dei modelli di effettuare previsioni (**forecasting**) sui dati. I modelli sviluppati quindi saranno testati e confrontati tra loro al fine di individuare il più efficace.

Infine, nell'ultima parte il progetto esplorerà l'utilizzo dei modelli di linguaggio (**LLM**) per la generazione di **dataset sintetici**. L'obiettivo sarà verificare se tali modelli possano produrre dataset *gemelli* che mantengano la **stessa struttura** e **le stesse informazioni** di quelli originali, ma con un numero ridotto di osservazioni. Questo per verificare se il loro utilizzo potrebbe essere utile al fine di trarre conclusioni senza dover elaborare un numero enorme di dati.

2 Analisi Dataset Tweet

Il dataset su cui lavoreremo è così strutturato:

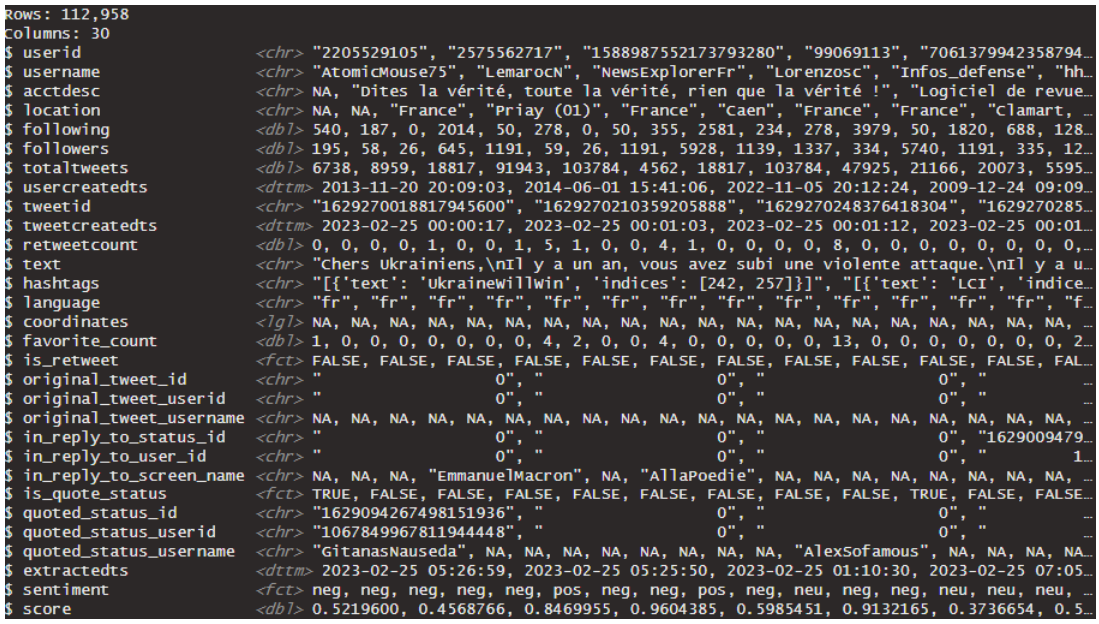


Figure 1: Contenuto del dataset

Contiene **112958 osservazioni** descritte da **30 variabili** rispettivamente divise in

- **17** variabili di classe **chr** (formato stringhe);
- **7** di classe **dbl** (formato numerico);
- **3** di classe **lgl** (formato categoriale);
- **3** di classe **dttm** (formato temporale).

Nello specifico le variabili rappresentano:

1. **userid**: il codice univoco dell'utente che ha postato il tweet;
2. **username**: il nickname dell'utente che ha postato il tweet;
3. **acctdesc**: descrizione del profilo dell'account che ha postato il tweet;
4. **location**: le posizione geografica da dove è stato postato il tweet;
5. **following**: il numero di persone che l'utente segue nel momento di estrazione del dato;
6. **followers**: il numero di persone che seguono l'utente nel momento di estrazione del dato;
7. **totaltweets**: il numero di tweets che l'utente ha postato nel momento di estrazione del dato;
8. **usercreatedts**: quando è stato creato il profilo dell'utente che ha postato il tweet;

9. **tweetid**: il codice univoco del tweet;
10. **tweetcreatedts**: quando è stato postato il tweet;
11. **retweetcount**: il numero di retweets ottenuti dal post nel momento di estrazione del dato;
12. **text**: contenuto del tweet;
13. **hashtags**: insieme degli hashtags contenuti nel tweet;
14. **language**: lingua del tweet;
15. **coordinates**: coordinate geografiche della location;
16. **favorite_counts**: il numero di like ottenuti dal post nel momento di estrazione del dato;
17. **is_retweet**: TRUE se il post è un retweet (FALSE altrimenti);
18. **original_tweet_id**: Id del tweet che è stato retweettato;
19. **original_tweet_userid**: Id dell'utente che è stato retweettato;
20. **original_tweet_username**: username dell'utente che è stato retweettato;
21. **in_reply_to_status_id**: Id del post che è stato risposto;
22. **in_reply_to_userid**: Id dell'utente che è stato risposto;
23. **in_reply_to_screen_name**: username dell'utente che è stato risposto;
24. **is_quote_status**: TRUE se il post contiene una citazione (FALSE altrimenti);
25. **quoted_status_id**: Id del post che è stato citato;
26. **quoted_status_userid**: Id dell'utente che è stato citato;
27. **quoted_status_username**: username dell'utente che è stato citato;
28. **extractedts**: quando è stato estratto il dato;
29. **sentiment**: categoriale che identifica il sentiment del post (neg, pos, neu);
30. **score**: valore quantitativo del sentiment;

Dato che la nostra analisi vertirà sull'utilizzo delle informazioni numeriche e temporali dei dati, ecco che da questo dataset abbiamo eliminato le variabili che non ci interessava tenere, ossia: **accdesc**, **location**, **text**, **hashtags**, **language** e **coordinates**.

> **Nota.** Il valore delle variabili fa riferimento al momento di estrazione del dato. Per questo motivo nel dataset possiamo ci può essere la ripetizione di uno stesso *tweetid* o *userid* ed osservare che i valori delle variabili sono diversi.

Analizziamo il dataset partendo dalle variabili numeriche che lo compongono ossia: *following, followers, totaltweets, retweetcount, favorite_count e score.* Nello specifico estraiamo

	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
following	112958	1616.50	3098.48	686.00	1107.81	853.98	0.00	237024.00	237024.00	16.68	767.71	9.22
followers	112958	14686.02	297700.05	507.00	930.46	687.93	0.00	10593615.00	10593615.00	31.54	1070.53	885.77
totaltweets	112958	70915.77	144023.86	22766.50	39161.61	29928.50	1.00	4196373.00	4196372.00	5.76	57.03	428.52
retweetcount	112958	191.83	390.80	28.00	90.26	41.51	0.00	6698.00	6698.00	3.21	12.26	1.16
favorite_count	112958	1.22	37.53	0.00	0.00	0.00	0.00	11659.00	11659.00	267.80	82445.15	0.11
score	112958	0.73	0.18	0.77	0.74	0.22	0.34	0.97	0.64	-0.41	-1.21	0.00

Figure 2: Tabella con le statistiche quantitative

da esse le informazioni statistiche più importanti che ci possono aiutare a descriverne la struttura:
Cosa possiamo dire osservando questi valori?

- Tutte le variabili, tranne score, sembrano mostrano una distribuzione **fortemente asimmetrica**
  - *La media è molto più alta della mediana* per la maggior parte delle variabili, indicando una lunga coda destra, **asimmetria positiva**.
- Forte presenza di **Outlier estremi** in `followers`, `totaltweets`, `retweetcount` e `favorite_count`
  - L'elevata curtosi infatti ci suggerisce che *poche osservazioni hanno valori molto superiori alla media*

2.1 Distribuzione dei dati

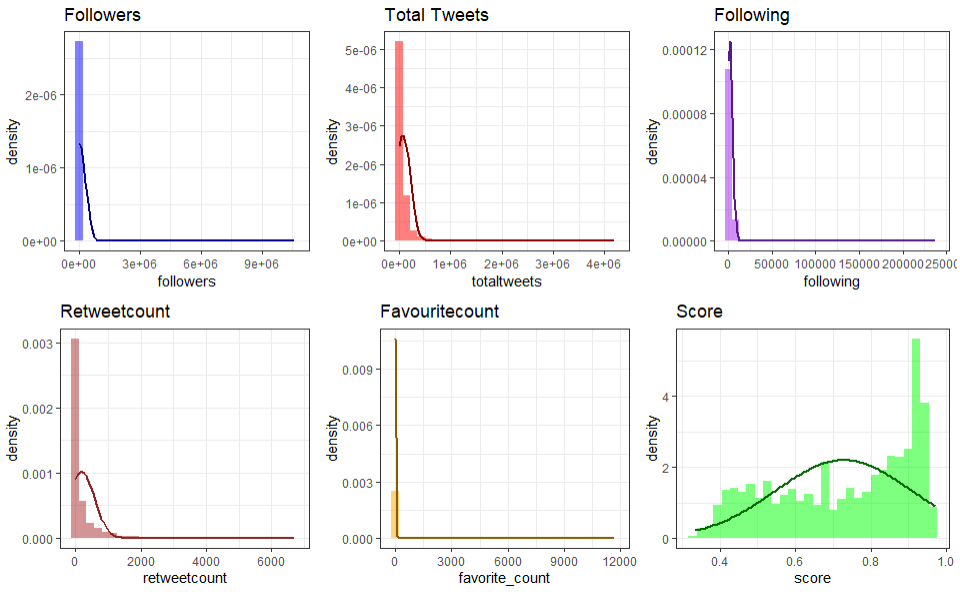


Figure 3: Istogrammi di ogni variabile insieme a Gaussiana teorica sui dati

Come prima cosa visualizziamo la **distribuzione delle variabili** attraverso l'utilizzo dell'**istogramma**, **Figura** 3, per vedere se le supposizioni che abbiamo effettuato sulle distribuzioni sono vere.

Esattamente come si evinceva dalle prime analisi, le distribuzioni delle variabili sono **fortemente asimettriche positive** presentando una coda molto lunga nella parte destra del grafico. Per riuscire a migliorare la leggibilità delle distribuzioni utilizzeremo due strategie, **trasformazione logaritmica** dei dati e l'**eliminazione degi outliers**.

2.1.1 Trasformazione logaritmica

Per osservare come cambia la distribuzione delle variabili se ci spostiamo in scala logaritmica, ma sopratutto anche per avere un'idea della quantità di outliers presenti utilizzeremo il **boxplot**, **Figura** 4. Quello che si nota è che grazie alla trasformazione logaritmica la forza dell'asimettria è calata, seppur rimane ancora presente sopratutto nelle variabili *totaltweets, followers e favorite_count*. Sicuramente la variabile score sta soffrendo dell'effetto di scala, mentre il resto delle variabili no, questo ci indica che il range dei valori delle altre variabili è pressocché simile. Mentre favorite_count mostra un comportamento molto atipico con la maggior parte dei dati sono compressi sullo 0. Questo è dovuto al fatto che nel dataset i tweet che ricevono almeno un like sono molti di meno rispetto a chi non ne riceve nessuno.

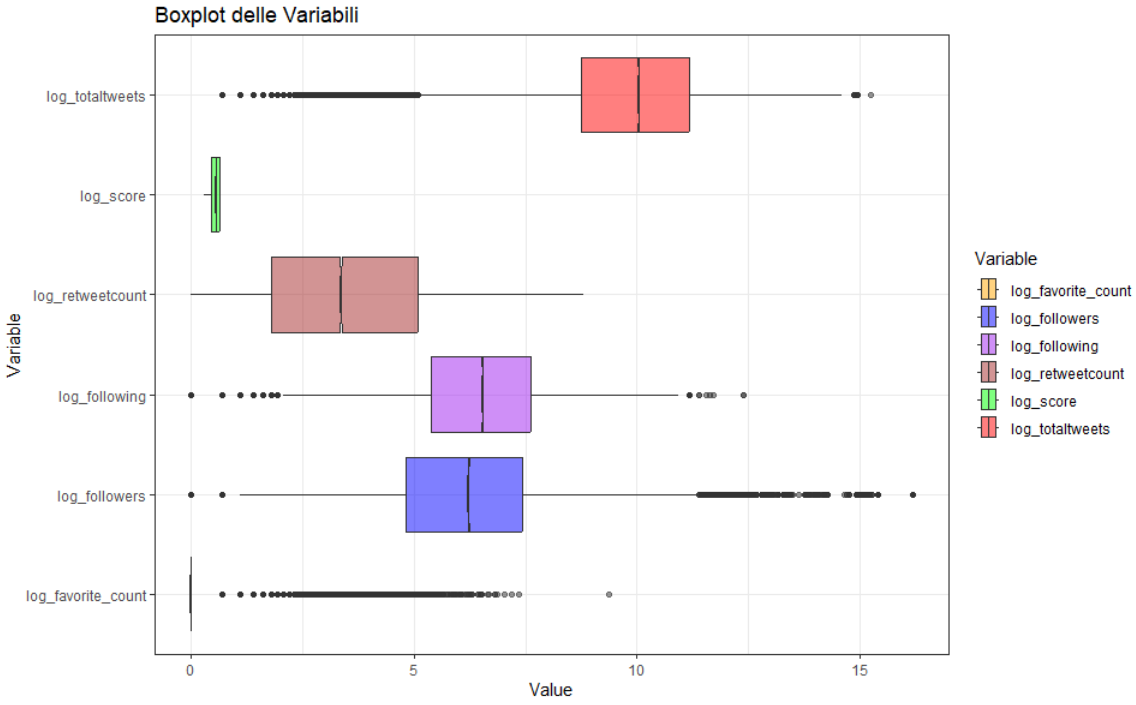


Figure 4: Boxplot di ogni variabile in scala logaritmica

2.1.2 Identificazione ed eliminazione degli outliers

Per cercare di migliorare ancora le distribuzione delle variabili, opereremo ora un'eliminazione degli outliers. Siccome la loro presenza in alcune variabili sembra essere molto forte, implementeremo più metodi di identificazione per poi utilizzare la tecnica che ne cattura il numero maggiore. I metodi in questione sono:

- **Regola di Tukey** o **IQR Method**, valori oltre $\pm 1.5 * IQR$ dai quartili sono considerati outlier;

```r
IQR_method <- function(x) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  return(x < lower_bound | x > upper_bound)
    }
```

Listing 1: Funzione IQR Method

- **Metodo dello Z-score** (Standard Score), valori oltre ± 3 dello z-score possono essere considerati outlier;

```r
ZScores_method <- function(x) {
  z_scores <- (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
  return(abs(z_scores) > 3)
}
```

Listing 2: Funzione Z-Scores Method

- **Metodo di Hampel**, basato sulla mediana e la deviazione mediana assoluta (MAD)

```r
ZScores_method <- function(x) {
  med <- median(x, na.rm = TRUE)
  mad_val <- mad(x, constant = 1.4826, na.rm = TRUE) #MAD + costante per
    normalizzazione
  return(abs(x - med) / mad_val > 3)
}
```

Listing 3: Funzione Z-Scores Method

Applicando tutti e tre i metodi alle variabili possiamo osservare il numero di outliers identificati da ciascun metodo per ciascuna variabile sia in scala normale che logaritmica, **Figura** 5.

- **Variabili non trasformate**
  Le variabili *following, followers, totaltweets, retweetcount, favorite_count* hanno un numero elevato di outlier secondo tutti i metodi. *Score* non presenta outlier in nessun metodo, suggerendo una distribuzione più regolare.

- **Variabili logaritmiche**
  Dopo la trasformazione logaritmica, il numero di outlier si **riduce drasticamente** per tutte le variabili, specialmente per *log_followers, log_following, log_totaltweets*. Per *log_retweetcount* si hanno zero outlier con ogni metodo, suggerendo che la trasformazione ha eliminato gran parte dell'estrema variabilità nei dati originali.

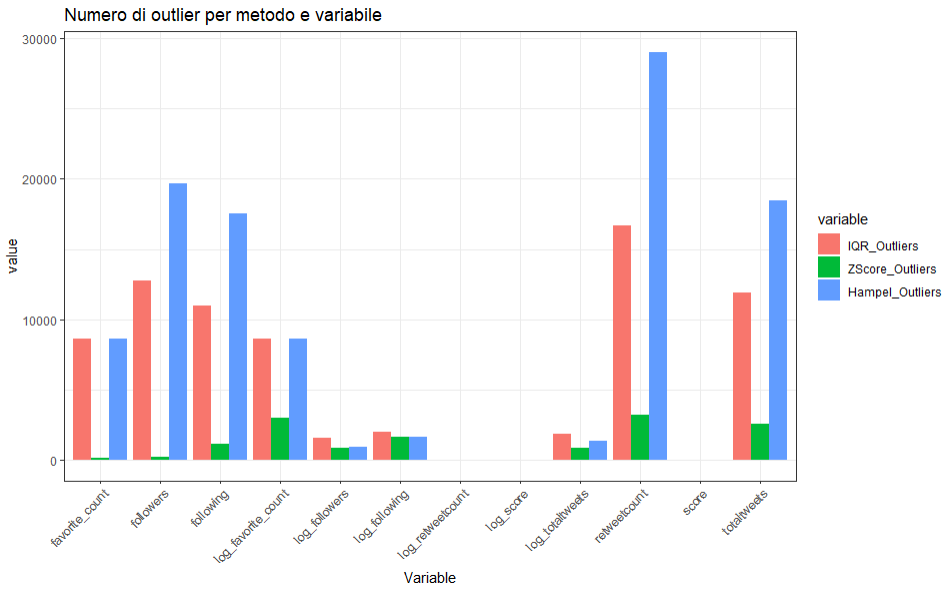


Figure 5: `Barplot` del numero di outliers identificanti per metodo

Si nota facilmente come il metodo che individua più outliers sia il metodo di Hampel, per questo motivo useremo quest'ultimo per estrapolarli ed eliminarli. Una volta effettuata la pulizia, possiamo di nuovo osservare come sono cambiate le distribuzioni una volta eliminati gli outliers, **Figura** 6. Si nota come l'eliminazione dei valori estremi abbiamo portato ad un **accentramento della distribuzione** soprattutto per le variabili *log_followers, log_totaltweets e log_following*, mentre *log_retweetcount e log_score* sono rimaste quasi invariate mostrando che gli outliers eliminati non influivano di molto nella visualizzazione. Dal density plot si evince anche come soprattutto *log_followers e log_following* presentino una distribuzione **multimodale**, mentre *log_totaltweets* sembrerebbe essere tendente alla normalità; cosa che invece nessun altra distribuzione sembra avere.
Per verificare queste ipotesi effettuiamo un test di Normalità delle distribuzioni.

2.1.3 Verifica della Normalità

Per testare quanto le distribuzioni si avvicinino alla Normalità (distribuzione Gaussiana) effettueremo un'analisi del **QQ-Plot** (grafico utilizzato per confrontare la distribuzione dei quantili empirici dei dati con una distribuzione dei quantili teorici della normale) e due test delle ipotesi:

1. **Test del Chi-Quadro**
   Il test χ^2 per la normalità è un test statistico che verifica se un insieme di dati segue una distribuzione normale. Si divide il dominio della variabile in k classi e si confrontano le frequenze osservate O_i con quelle attese E_i sotto l'ipotesi di normalità. La statistica test è:
   $$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$
   che segue approssimativamente una distribuzione χ^2 con $k - p - 1$ gradi di libertà,

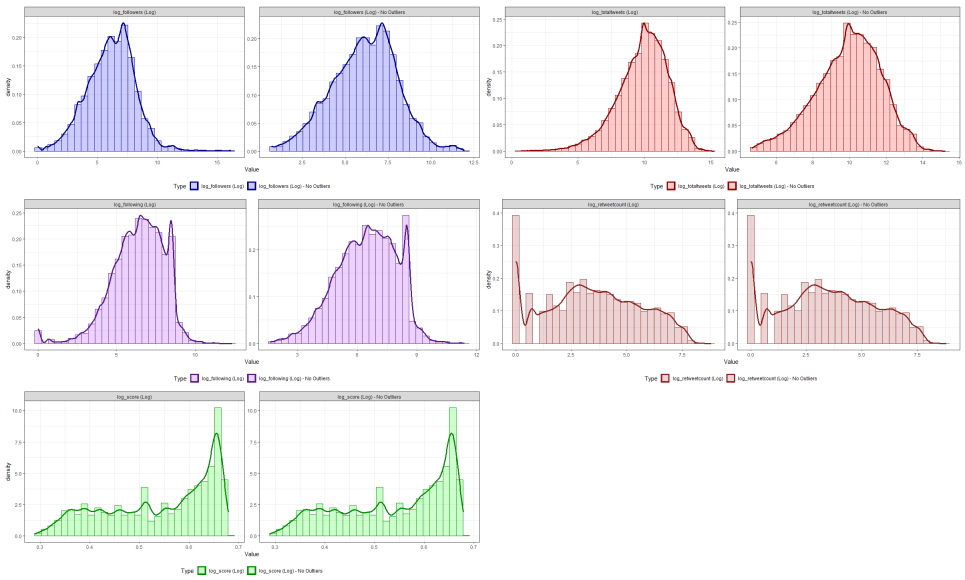


Figure 6: Confronto istogrammi-density plot pre e post eliminazione outliers

dove p è il numero di parametri stimati.

2. **Test Kolmogorov-Smirnov** (più robusto per grandi campioni di dati)
   Il test di Kolmogorov-Smirnov (K-S) confronta la funzione di distribuzione empirica $F_n(x)$ di un campione con la funzione di distribuzione cumulativa teorica $F(x)$ della distribuzione normale. La statistica test è:

$$D_n = \sup_x |F_n(x) - F(x)|$$

   dove D_n rappresenta la massima differenza assoluta tra le due funzioni di distribuzione. Se D_n è sufficientemente grande, si rifiuta l'ipotesi di normalità.

Quindi una volta effettuata l'analisi con il grafico QQ-Plot, **Figura** 7, e testate la normalità delle distribuzioni con i due metodi possiamo dire che **nessuna distribuzione segue perfettamente la Gaussianità** sebbene i grafici mostrino che i quantili empirici seguano per lunghi tratti la distribuzione dei quantili teorici; ma la struttura delle code li distanzia da essa. Infatti, per entrambi i test si rifiuta l'ipotesi di normalità in quando $pvalue \leq 0.05$.

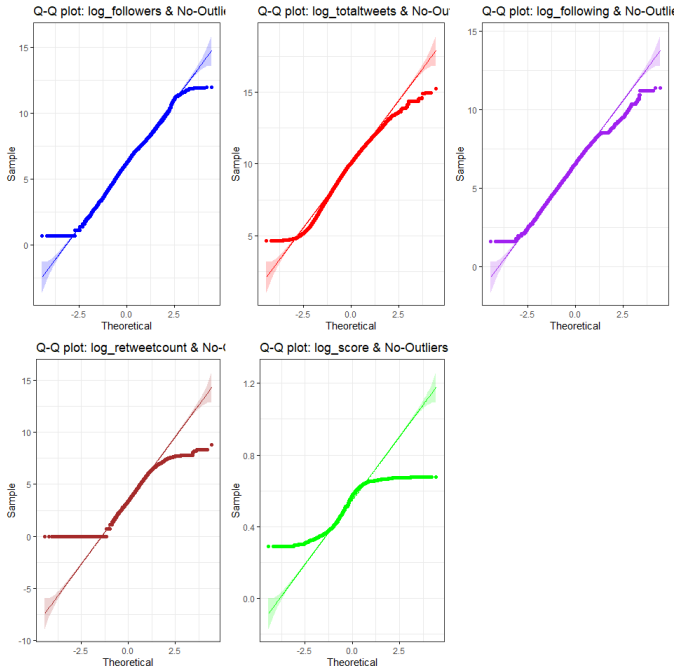


Figure 7: QQ-Plot post eliminazione

2.2 Relazioni lineari fra le variabili

È importante, in un'analisi statistica completa, valutiamo se sono presenti relazioni lineari fra le variabili numeriche del dataset, in quanto ci aiutano descrivere meglio la struttura d'insieme del dataset. Per osservare tali relazioni utilizzeremo come grafico lo **scatterplot** a coppie fra tutte le variabili, invece come statistica calcoleremo la **matrice delle correlazioni di Pearson** che indica la **forza** e la **direzione** di una relazione lineare tra due variabili quantitative (varia tra -1 e 1). Osservando la **Figura** 8 ci rendiamo subito conto che esistono relazioni lineari anche piuttosto forti tra alcune variabili (es. fra *log_following* e *log_followers*), mentre fra altre questa condizione non persiste.
Osservando meglio la matrice delle correlazioni visualizzata tramite `corrplot`, possiamo trarre alcune considerazioni:

- **Correlazioni forti**
  - log_followers e log_following (0.69): **utenti con più follower tendono a seguire più persone;**
  - log_followers e log_totaltweets (0.65): **chi ha più follower tende ad aver pubblicato più tweet.**
- **Correlazioni moderate**
  - log_totaltweets e log_following (0.47): **gli utenti che seguono molte persone tendono ad aver scritto più tweet;**
  - log_followers e log_favorite_count 0.27): **chi ha più follower potrebbe ricevere più like sui post.**

- **Correlazioni deboli o inverse**
  - log_retweetcount e log_followers (-0.16): **non necessariamente più follower significano più retweet;**
  - log_retweetcount e log_favorite_count (-0.20): **potrebbe indicare che alcuni tweet ricevono più retweet a discapito dei like**;
  - score mostra correlazioni molto basse con tutte le altre variabili, suggerendo che **potrebbe non dipendere linearmente** dalle metriche considerate.

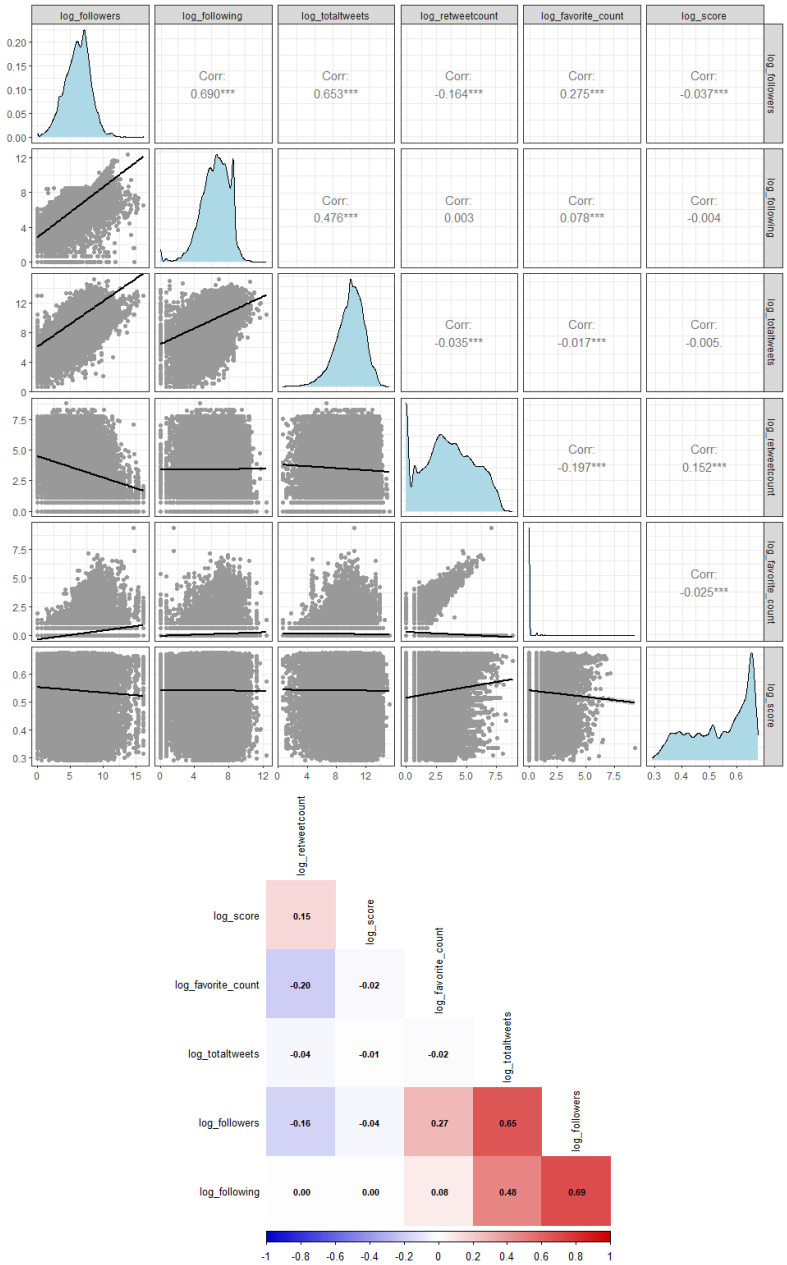


Figure 8: Output funzione **GGally::ggpairs** e Matrice delle Correlazioni

2.3 Regressioni Lineari

Dall'analisi delle relazioni lineari abbiamo visto che alcune variabili possono essere interpretate linearmente in parte da un'altra variabile, ciò ci fa presupporre che la loro distribuzione possa essere modellata come una **combinazione lineare** delle altre variabili.

2.3.1 Regressione lineare - y: Followers

Prendiamo ad esempio la variabile **log_followers** essa si lega linearmente, con forza (**0.69**), alla variabile **log_following**, ma è legata linearmente anche ad altre variabili come log_totaltweets oppure log_favorite_count seppur in maniera più debole. Allora proviamo a modellare tale variabile come una combinazione lineare di tutte le altre variabili quantitative e vediamo quanto della sua distribuzione riusciamo a spiegare.

Research Question. (1) È possibile costruire un modello di Regressione Lineare Multipla capace di spiegare come e chi influenza l'andamento del numero di Followers?

Il primo modello di regressione lineare che costruiamo è il seguente:

$$\begin{aligned} log\_followers = \beta_0 + \beta_1 \cdot log\_following + \beta_2 \cdot log\_totaltweets + \\ \beta_3 \cdot log\_retweetcount + \beta_4 \cdot log\_favorite\_count + \beta_5 \cdot log\_score + \epsilon \end{aligned} \tag{1}$$

Il **summary(model)** che otteniamo è il seguente:

```
Call:
lm(formula = log_followers ~ log_following + log_totaltweets +
    log_retweetcount + log_favorite_count + log_score, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9431 -0.6501 -0.1042  0.4881  9.8428

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       -1.618812   0.025417 -63.689  < 2e-16 ***
log_following      0.562692   0.002299 244.789  < 2e-16 ***
log_totaltweets    0.456795   0.002031 224.954  < 2e-16 ***
log_retweetcount  -0.096915   0.001618 -59.892  < 2e-16 ***
log_favorite_count 0.805317   0.006136 131.242  < 2e-16 ***
log_score         -0.213529   0.031532  -6.772 1.28e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.133 on 112952 degrees of freedom
Multiple R-squared:  0.6834,    Adjusted R-squared:  0.6834
F-statistic: 4.876e+04 on 5 and 112952 DF,  p-value: < 2.2e-16
```

Figure 9: summary() del modello di regressione lineare

Le principali conclusioni che possiamo trarre da questo output sono che:

1. Il modello è **altamente significativo** (p-value $< 2.2e-16$).
2. L'$\mathbf{R^2} = \mathbf{0.6834}$, indicando che circa il 68.34% della variabilità nei follower è spiegata dalle variabili indipendenti.
3. Tutti i coefficienti sono **statisticamente significativi** ($p < 0.001$).

Risposta. (1) Per spiegare quali fattori influenzano la variabile Followers bisogna osservare i valori dei **coefficienti di regressione stimati**

1. log_following (+0.563): un aumento del 1% del following comporta un aumento del valore atteso pari al 0.56% circa nel numero di followers;
2. log_totaltweets (+0.457): **più tweet implica** in percentuale **anche più followers**, circa del 0.45% per ogni aumento del 1%;
3. log_favorite_count (+0.805): i **like sono fortemente correlati con un maggior numero di follower**;
4. log_score (−0.214): un maggiore punteggio (qualunque esso sia) impatta negativamente sul numero di follower diminuendo del 0.2% circa ogni 1% in più.

Per verificare la bontà di adattamento del modello in maniera ancora più parsimoniosa, dobbiamo effettuare la **Diagnosi dei Residui** la quale ci dice che un modello riesce a catturare al 100% la variabilità dei dati se e solo se sono valide le seguenti condizioni:

1. Distribuzione normale dei residui (Test di Shapiro-Wilk)
2. Autocorrelazione dei residui

Par verificare le ocndizioni usiamo la funzione:

```
forecast::checkresiduals(model)
```

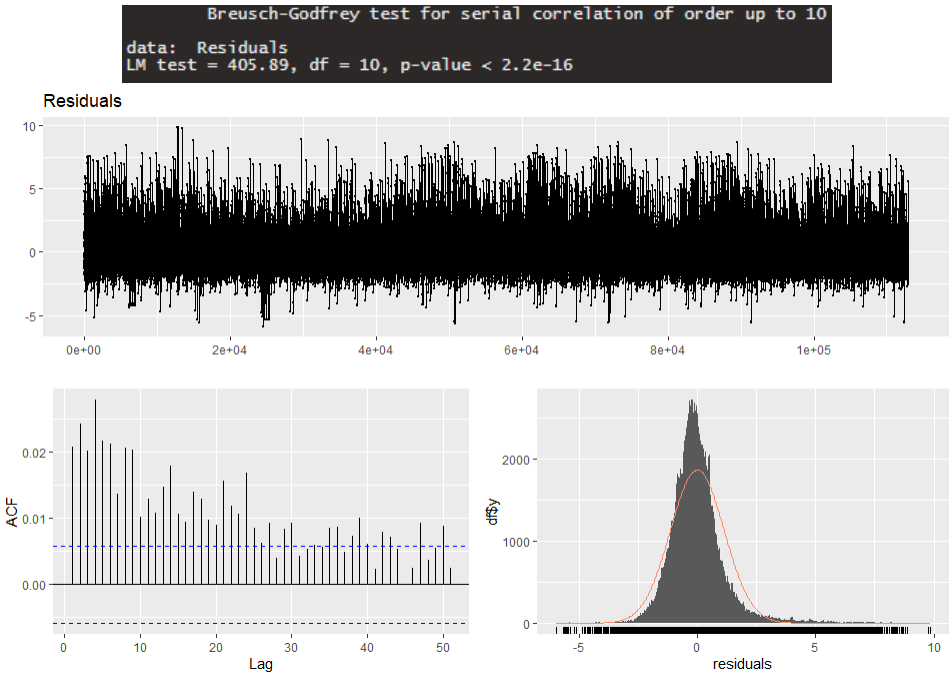


Figure 10: Output Test di Shapiro-Wilk e checkresiduals

Sia dal test di Shapiro-Wilk che dal grafico si evince che i residui del modello **non** seguano la

distribuzione normale e presentano autocorrelazione residua. Questo indica che **non** tutta la variabilità dei dati è catturata dal modello.

2.3.2 Regressione lineare - y: Score

Research Question. (2) Abbiamo visto dall'analisi delle relazioni lineari che la variabile score non è legata linearmente a nessun altra variabile quantitativa, ma se considerassimo una combinazione lineare delle variabili con l'aggiunta anche delle **variabili categoriali**, la capacità esplicativa aumenterebbe?

Per rispondere a questa domanda costruiamo il modello lineare

$$\begin{aligned} log\_score = \beta_0 + \beta_1 \cdot log\_following + \beta_2 \cdot log\_totaltweets + \\ \beta_3 \cdot log\_retweetcount + \beta_4 \cdot log\_favorite\_count + \beta_5 \cdot log\_followers + \\ \alpha_1 \cdot is\_retweetTRUE + \alpha_2 \cdot is\_quote\_statusTRUE + \\ \alpha_3 \cdot sentimentNEU + \alpha_4 \cdot sentimentPOS + \epsilon \end{aligned} \tag{2}$$

Il **summary(model)** che otteniamo è il seguente:

```
Call:
lm(formula = score ~ ., data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.45724 -0.11055  0.04229  0.12259  0.36567

Coefficients:
                    Estimate Std. Error  t value Pr(>|t|)
(Intercept)        0.7790956  0.0028926  269.342  < 2e-16 ***
log_followers     -0.0009601  0.0004145   -2.316  0.02056 *
log_following      0.0010834  0.0004013    2.700  0.00694 **
log_totaltweets   -0.0004537  0.0003412   -1.330  0.18369
log_retweetcount   0.0142177  0.0002891   49.182  < 2e-16 ***
log_favorite_count -0.0101183  0.0010502   -9.635  < 2e-16 ***
is_retweetTRUE    -0.0482994  0.0019139  -25.236  < 2e-16 ***
is_quote_statusTRUE  0.0203032  0.0016245   12.498  < 2e-16 ***
sentimentneu      -0.2140961  0.0013206 -162.118  < 2e-16 ***
sentimentpos      -0.1426388  0.0012799 -111.444  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1571 on 112948 degrees of freedom
Multiple R-squared:  0.2499,    Adjusted R-squared:  0.2499
F-statistic:  4182 on 9 and 112948 DF,  p-value: < 2.2e-16
```

Figure 11: summary() del modello di regressione lineare

L'**interpretazione dei coefficienti** ci porta a dire che:

- (Intercept) = 0.779: Il valore atteso in % dello score quando tutte le variabili indipendenti sono zero, ossia quando non è un retweet, nemmeno una citazione e il sentiment è negativo;
- log_retweetcount (+0.0142): i retweet aumentano in percentuale lo score

- log_favorite_count (-0.0101): contrariamente ai retweet, i like sono associati a una riduzione del score, seppur in maniera molto debole;
- is_retweetTRUE (-0.0483): se un post è un retweet, lo score diminuisce in media di 0.0483%, suggerendo che i tweet classici tendono ad avere un punteggio più alto;
- is_quote_statusTRUE ($+0.0203$): essere un tweet con una citazione porta ad avere mediamente un score leggermente più alto, ma di pochissimo;
- sentimentneu (-0.2141) e sentimentpos (-0.1426): indica che sia i tweet neutri che quelli positivi mediamente sono associati a punteggi più bassi rispetto a quelli negativi;

Risposta. (2) Il modello è significativo e ci permette di descrivere in maniera più precisa le relazioni fra le variabili, ma la sua capacità di catturare la variabilità dei dati è bassa. Infatti un $R^2 = 0.25$ indica che il 75% della variabilità **non** è spiegata, il che ci indica che molto probabilmente la struttura che regola la variabile score è più complessa.

2.3.3 Regressione multinomiale - y: Sentiment

Research Question. (3) È possibile costruire un modello che ci aiuti a spiegare cosa impatta nell'etichettatura di un tweet come **Neutrale, Positivo o Negativo**?

Stiamo cercando dunque di *predire il sentiment* (che è una variabile categorica con tre livelli: *negativo*, *neutro* e *positivo*) in base alle variabili esplicative viste in precedenza. Per farlo, la scelta più logica è quello di usare un modello *multinomiale* in quanto permette di stimare la probabilità di ciascun livello della variabile categorica in relazione alle variabili indipendenti. Il modello multinomiale è così costruito:

$$\begin{aligned}\text{logit}(P(\text{sentiment} = k)) =& \beta_0^{(k)} + \beta_1^{(k)} \cdot \text{log\_following} + \beta_2^{(k)} \cdot \text{log\_followers} + \beta_3^{(k)} \cdot \text{log\_totaltweets} + \\ & \beta_4^{(k)} \cdot \text{log\_retweetcount} + \beta_5^{(k)} \cdot \text{log\_favorite\_count} + \beta_6^{(k)} \cdot \text{log\_score} \\ & \beta_7^{(k)} \cdot \text{is\_retweetTRUE} + \beta_8^{(k)} \cdot \text{is\_quote\_statusTRUE}\end{aligned} \tag{3}$$

Dove $\text{logit}(P(\text{sentiment} = k))$ è la funzione logistica che descrive la probabilità che il sentiment faccia parte della categoria k-esima, ossia *neg, neu o pos*.

```
Call:
multinom(formula = sentiment ~ ., data = df)

Coefficients:
    (Intercept) log_following log_followers log_totaltweets log_retweetcount log_favorite_count is_retweetTRUE is_quote_statusTRUE  log_score
neu    5.750340   -0.04789599    0.06520649     -0.07422326      -0.01652485       -0.09110507    -0.34144111          -0.2774030 -12.141198
pos    3.280338    0.01101317    0.06981298     -0.09004268      -0.00872174       -0.01043088     0.04314175           0.2451836  -8.079562

Std. Errors:
    (Intercept) log_following log_followers log_totaltweets log_retweetcount log_favorite_count is_retweetTRUE is_quote_statusTRUE  log_score
neu  0.07162513   0.007435656   0.007770523     0.006561595      0.006134194        0.02009507     0.03671124           0.03523007 0.09442649
pos  0.06709327   0.007381749   0.007381132     0.006113659      0.005424289        0.01867025     0.03497750           0.02808377 0.08263845
```

Figure 12: summary() del modello di regressione multinomiale

Coefficienti per k = Neutro rispetto a k = Negativo

- *log_following* (-0.0479): Il coefficiente negativo suggerisce che più sono i *following*, meno probabile è che il sentiment sia neutro.
- *log_followers* ($+0.0652$): Maggiore è il numero di *followers*, maggiore è la probabilità di un sentiment neutro.
- *log_totaltweets* (-0.0742): Più tweet un utente ha postato, più è probabile che il suo sentiment sia negativo.
- *log_retweetcount* (-0.0165): I *tweet* con più retweet sono probabilmente associati a sentiment più forti, come quello negativo.
- *log_favorite_count* (-0.0911): Più *like* suggeriscono un maggiore coinvolgimento e, quindi, un sentiment potenzialmente più positivo o negativo.
- *is_retweetTRUE* (-0.3414): I *retweet* sono tipicamente meno neutri, tendendo verso il positivo o negativo.
- *is_quote_statusTRUE* (-0.2774): I *tweet* citati tendono a evocare sentimenti più forti.
- *log_score* (-12.1412): Lo *score* più alto è fortemente associato a un cambiamento nel sentiment verso il negativo.

Coefficienti per k = Positivo rispetto a k = Negativo

- *log_following* ($+0.0110$): l'effetto è piuttosto piccolo, suggerendo che i *following* non influiscono molto sul sentiment positivo.
- *log_followers* ($+0.0698$): Maggiore è il numero di *followers*, più probabile che il sentiment sia positivo.
- *log_totaltweets* (-0.0900): Più tweet un utente pubblica, meno probabile che il suo sentiment sia positivo.
- *log_retweetcount* (-0.0087): Il numero di *retweet* ha un effetto negativo molto più ridotto sul sentiment positivo rispetto a quello neutro, suggerendo che i *retweet* sono meno influenti nel determinare un sentiment positivo.
- *log_favorite_count* (-0.0104): Un aumento dei *like* ricevuti riduce leggermente la probabilità di un sentiment positivo rispetto al sentiment negativo, ma l'effetto è modesto.
- *is_retweetTRUE* ($+0.0431$): Se un *tweet* è un *retweet*, aumenta la probabilità che il sentiment sia positivo rispetto a negativo.
- *is_quote_statusTRUE* ($+0.2452$): I *tweet* con citazioni hanno una probabilità maggiore di essere positivi rispetto che negativi.
- *log_score* (-8.0796): Un aumento del logaritmo dello *score* riduce la probabilità di avere un sentiment positivo rispetto al sentiment negativo.

Risposta. (3) Dall'analisi dei coefficienti, sono emerse alcune caratteristiche chiave sul sentiment dei tweet che possiamo rassumere come segue

1. Il sentiment **negativo** è spesso associato a utenti **più attivi**, con un **elevato numero di tweet**, **retweet** e **interazioni**.
2. I tweet associati ad utenti con più follower invece tendono ad avere sentiment più **neutri** o **positivi**.
3. Un score più alto è **fortemente correlato** al sentiment **negativo**.

2.4 Analisi Temporale

Un'analisi molto interessante da poter fare sui dati è quella di osservare la distribuzione nel tempo dei twitter; contando il numero di occorrenze dei post per sentiment in un dato sotto-intervallo di tempo.

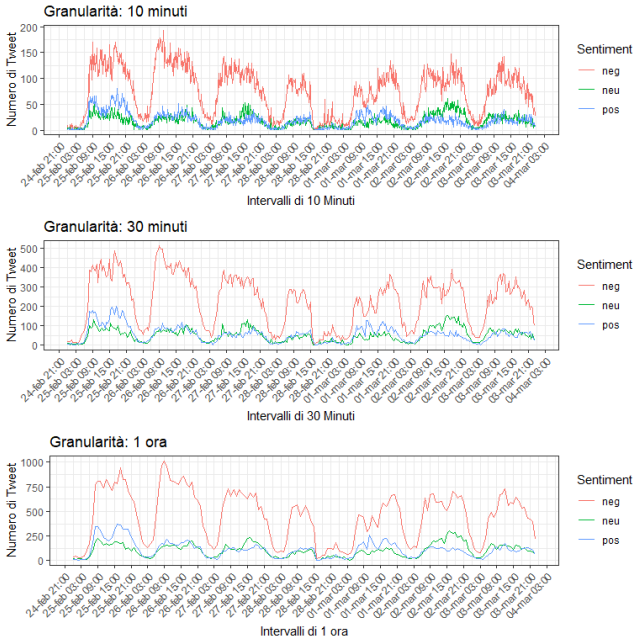


Figure 13: Time series delle occorrenze dei tweet calcolate per diversi sotto-intervalli di tempo

2.4.1 Analisi della periodicità

In **Figura** 13 si osserva chiaramente che nelle serie storiche è presente una **struttura di periodicità** temporale nei dati che si conserva a tutti i livelli di analisi temporale.
Per cercare di analizzare in maniera più robusta l'esistenza della periodicità nei dati, effettuaiamo un'analisi delle caratteristiche. Utilizzeremo `ggseasonplot()` per osservare se c'è un andamento comune nella serie e `ggACF()` per osservare se nei ritardi dei dati è presente autocorrelazione. Se in questi due grafici si osservano andamenti ripetitivi allora siamo in

presenza di serie con caratteristiche temporali forti.
Come si osserva in **Figura** 14, è chiaramente visibile la tendenza di decrescita iniziale seguita da una crescita molto forte nelle ore centrali della giornata che rimane stabile, per poi decrescere sul finale. Questo andamento rimane immutato per tutti i giorni di analisi e per tutte le tipologie di sentiment (per questo motivo abbiamo lasciato solo i grafici per il sentiment negativo, figura 14).

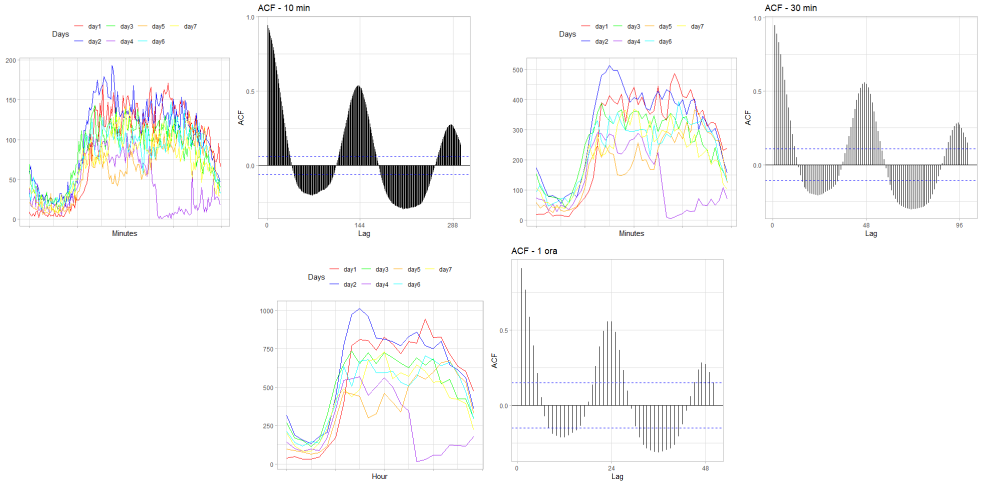


Figure 14: `seasonalplot` e `ACF` per le serie con sentiment negativo

2.4.2 Decomposizione di serie storiche

La decomposizione di una serie temporale consiste nel separare la serie nei suoi componenti fondamentali per analizzarne meglio la struttura e i pattern sottostanti. In generale, una serie temporale Y_t può essere scomposta in:

- **Trend** (T_t): il componente a lungo termine che rappresenta la direzione generale della serie.
- **Stagionalità** (S_t): il componente periodico che ripete a intervalli regolari.
- **Residuo** (R_t): il componente irregolare o casuale che rappresenta la variazione non spiegata dagli altri due.

A seconda della relazione tra questi componenti, si distinguono diversi tipi di decomposizione, ma in questa analisi ne testeremo 3 classiche e 1 costruita da noi.

Siccome la struttura delle serie è molto simile fra loro, d'ora in avanti l'analisi temporale che mostreremo sarà incentrata sulle serie negative.

1: Decomposizione Additiva
Se i componenti della serie temporale si combinano linearmente, si utilizza il modello additivo:

$$Y_t = T_t + S_t + R_t \tag{4}$$

Questo modello è adatto quando l'ampiezza della stagionalità e del rumore è costante nel tempo.

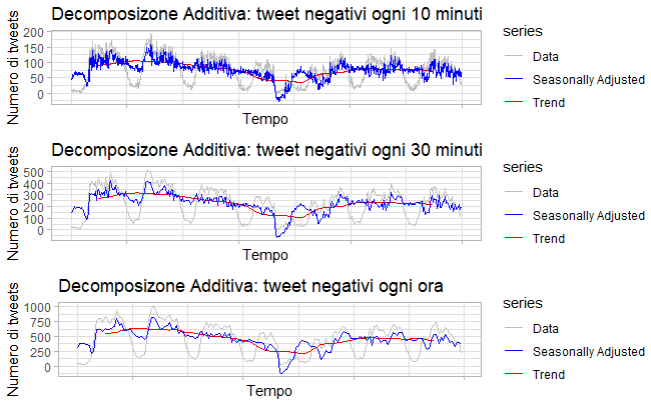


Figure 15: Grafico mostra come si adatta il modello di decomposizione additiva sulle time series

2: Decomposizione Moltiplicativa

Se l'ampiezza della stagionalità e del rumore varia in proporzione al livello della serie, si usa il modello moltiplicativo:

$$Y_t = T_t \cdot S_t \cdot R_t \tag{5}$$

Questo modello è utile per serie in cui la variabilità aumenta con il valore della serie temporale.

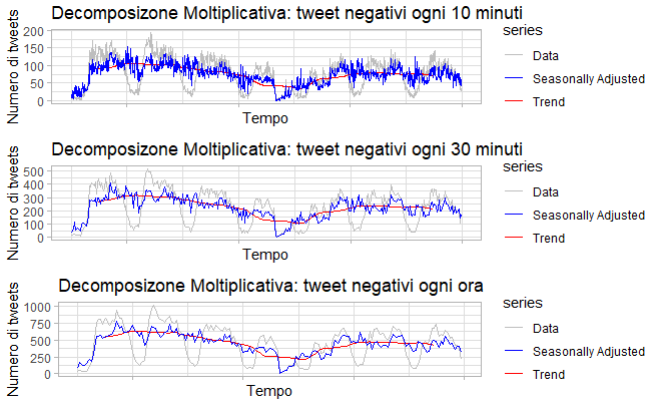


Figure 16: Grafico mostra come si adatta il modello di decomposizione moltiplicativa sulle time series

3: Decomposizione STL (Seasonal-Trend Decomposition using LOESS)

La decomposizione STL è una tecnica più avanzata che utilizza la regressione LOESS per estrarre il trend e la stagionalità in modo più flessibile. Il modello generale segue la struttura:

$$Y_t = T_t + S_t + R_t \tag{6}$$

ma con una metodologia iterativa che permette di **adattare il trend e la stagionalità nel tempo**. STL infatti è particolarmente utile quando i pattern stagionali cambiano nel tempo o quando il trend non è lineare.

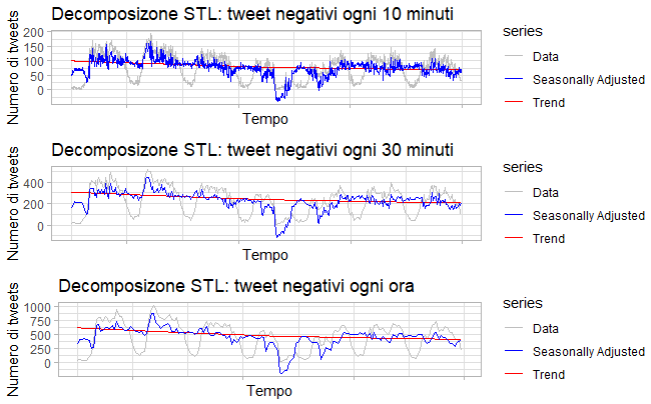


Figure 17: Grafico mostra come si adatta il modello di decomposizione STL sulle time series

4: Decomposizione con Termini di Fourier e Media Mobile

Questo modello di decomposizione è stato costruito da noi combinando una *serie di Fourier* per stimare la stagionalità e una *media mobile* per stimare il trend. La scomposizione quindi segue la forma:

$$Y_t = T_t + S_t + R_t \tag{7}$$

Dove:

- Il trend T_t viene stimato utilizzando una media mobile:

$$T_t = \frac{1}{N} \sum_{i=-k}^{k} Y_{t+i} \tag{8}$$

  N che rappresenta la finestra della media mobile pari alla frequenza di osservazione giornaliera dei dati (144 per 10 min, 48 per 30 min e 24 per 1 ora)

- La stagionalità S_t viene modellata con una serie di Fourier:

$$S_t = \sum_{n=1}^{N} \left[a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right] \tag{9}$$

  dove P è il periodo della serie e i coefficienti a_n, b_n sono stimati dai dati.
  Con i termini di Fourier, si ha bisogno di *meno* predittori specialmente quando la frequenza è alta (come nel nostro caso). Per questo motivo la scelta del numero ottimale K di termini di Fourier deve essere compreso in un intervallo di valori non troppo grandi (es. fra 1 e 10), in modo tale da catturare le principali componenti armoniche senza incappare nella modellazione del *rumore*.

```
K_values <- 1:10 # Testiamo K da 1 a 10
# funzione per valutare i K
K_fourier <- function(serie, K_values){
  aic_values <- numeric(length(K_values)) # lista per risultati
  for (i in seq_along(K_values)) {
    K <- K_values[i]
    # Genera i termini di Fourier
    fourier_terms <- fourier(serie, K = K)
```

```
    # Modello lineare con i termini di Fourier
    fit <- tslm(serie ~ fourier_terms)
    # Salvare l'AIC del modello
    aic_values[i] <- AIC(fit)
  }
  return(aic_values)
}
# chiamate a funzione per ottenere i K
aic_values10 <- K_fourier(neg_ts10, K_values)
aic_values30 <- K_fourier(neg_ts30, K_values)
aic_values1h <- K_fourier(neg_ts1h, K_values)
# Trovare il K ottimale, ossia quello con AIC piu piccolo
best_K10 <- K_values[which.min(aic_values10)]
best_K30 <- K_values[which.min(aic_values30)]
best_K1h <- K_values[which.min(aic_values1h)]
```

Abbiamo ottenuto che i migliori valori di K per le frequenze di 10 min, 30 min e 1 ora sono 5, 2 e 2.

- Il residuo R_t è la componente rimanente dopo la rimozione del trend e della stagionalità.

La **Figura** 18 mostra separatamente le stime delle caratteristiche delle serie e il modello di decomposizione completo come si adatta ai dati.

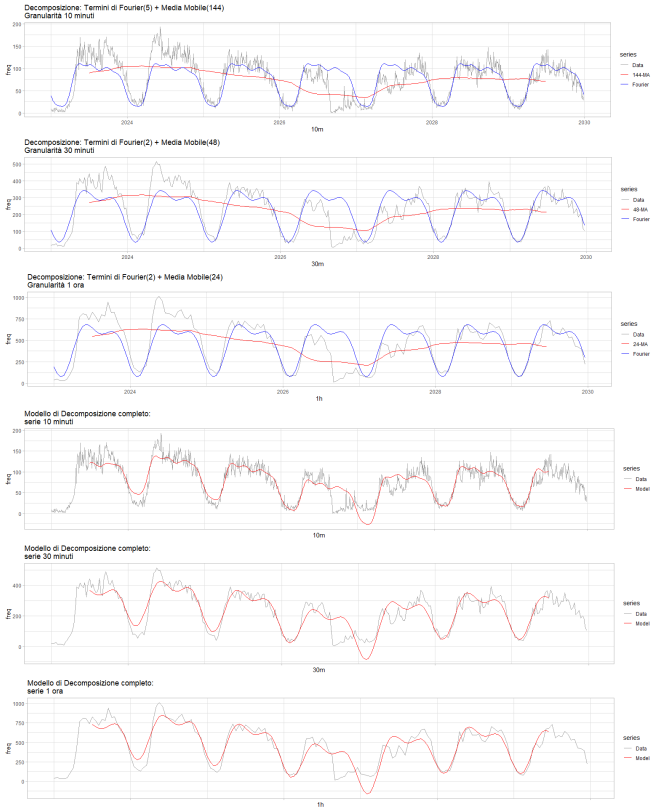


Figure 18: Grafico sopra mostra la serie si Fourier e la media mobile costruite sui dati, mentre il grafico sotto mostra i modelli di decomposizione completi

Research Question. (4) È possibile stimare l'andamento del numero di Tweet postati nel tempo? Se sì, sarebbe possibile effettuare previsioni?

Riflessioni generali:

- Notiamo che la decompozione **Additiva** riesce a modellare bene i picchi delle curve, mentre si perde l'informazione degli avvallamenti. Identifica un leggero trend di decrescita del numero di tweet effettuati all'incirca fino a metà serie, per poi riprendersi.
- La decompozione **Moltiplicativa**, a differenza di quella Additiva, risulta più centrale nella sua modellazione. In tal modo non si distanzia molto né dagli avvallamenti né dai picchi, ma così facendo l'informazione catturata potrebbe non essere ottimale. Nonostante ciò, riesce a rappresentare in maniera ottimale il punto di flessione minimo della serie, cosa che invece l'Additiva non riusciva a fare.
- Per quanto riguarda la decomposizione **STL**, mostra un comportamento simile a quello dell'Additiva nella prima parte, con la differenza che sembrerebbe cogliere un trend più debole nei dati. A differenza invece di quella Moltiplicativa, sovrastima il punto di flessione minimo della serie e questo la porta successivamente a non seguire bene l'andamento.
- Infine, la **nostra** decomposizione si concentra in una stima più smooth della serie riuscendo sia nella parte iniziale che finale a rappresentare discretamente bene la serie. Tuttavia la parte centrale della flessione non riesce a modellarla bene, indicando che è poco robusto a variazioni improvvise.

Risposta. (4) Tutti i modelli di Decomposizione ci permettono di stimare in maniera piuttosto precisa l'**andamento generale dei dati** nel tempo. Ci mostrano come le caratteristiche delle serie mutano nel tempo, permettendoci di descrivere in maniera più robusta le time series. Tuttavia parlare di **previsioni è azzardato** in quanto, se si intende prevedere *puntualmente* il numero di occorrenze future ciò è molto difficile (con questi metodi) poiché:

1. Ogni metodo è sensibile a caratteristiche diverse che lo portano a distanziarsi puntualmente dai valori originali della serie;
2. Il numero di occorrenze è sensibile a sua volta ad eventi esterni, dunque ha forte oscillazioni puntuali.

Se invece si intende effettuare previsioni sull'andamendo futuro *generale* della serie, allora i modelli possono aiutare in quanto permettono di stimare in maniera robusta il trend e la periodicità dei dati anche per intervalli di tempo futuri.

3 Time Series Forecast

Dal dataset non solo possiamo effettuare analisi di natura statistica sulle relazioni tra le variabili come abbiamo fatto fino ad ora, ma grazie alla presenza delle variabili temporali (*tweetcreatedtd* e *extractedts*) possiamo concentrarci nell'analizzare le relazioni delle variabili rispetto a loro stesse nel tempo. In questo studio ci focalizzeremo su due variabili, lo **Score** e il numero di **Tweet effettuati**, per vedere se è possibile costruire modelli statistici capaci di catturare l'andamento temporale dei dati.

3.1 Analisi Statistico Temporale dello Score

Innanzitutto, osserviamo come la variabile dello Score cambia nel tempo a seconda delle tre classi del Sentiment, **Figura** 19. Dall'immagine si nota molto bene come ci troviamo dinanzi a **serie storiche ad alta frequenza**, questo rende non solo la visualizzazione poco leggibile ma debiliterebbe anche la costruzione di modelli statisici sui dati. Per questo motivo dobbiamo effettuare un'operazione di diminuzione della frequenza, ma prima di ciò bisogna **analizzare statisticamente le time series**.

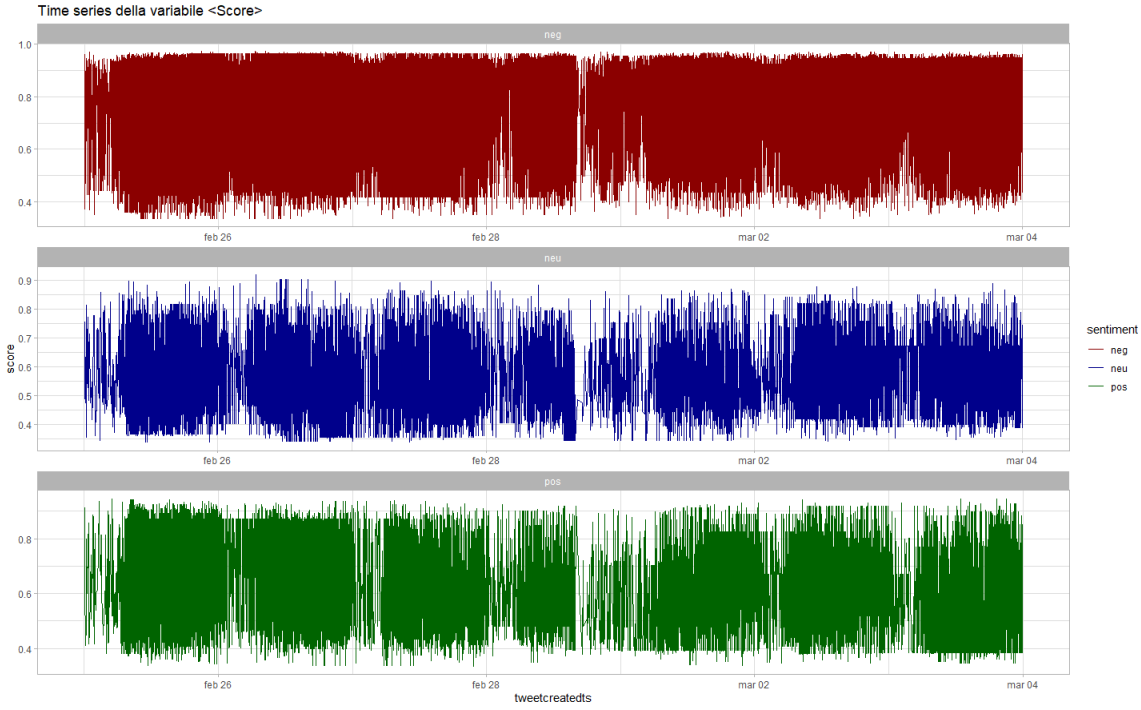


Figure 19: Time series della variabile Score suddivise per categoria di Sentiment

Analisi della Stazionarietà e Autocorrelazione

Sicuramente, per loro natura, le serie non presentano forti *trend*, né tantomeno forte *periodicità* o *ciclicità* dei dati. Siccome si distribuiscono come serie ad alta frequenza, molto probabilmente seguono una distribuzione **White Noise** e di **stazionarietà**, rimanendo stabili intorno alla propria media e con la stessa variabilità nel tempo.
Per verificare queste supposizioni effettueremo l'analisi utilizzando due test delle ipotesi:

1. **Test di Ljung-Box**, verifica l'**autocorrelazione residua** in una serie temporale.
    - *Ipotesi nulla* (H_0), non vi è autocorrelazione fino a un certo ritardo temporale: **distribuzione white noise**
    - *Ipotest alternativa* H_1, vi è autocorrelazione fino a un certo lag.
    - Per un *p-value piccolo* (< 0.05) si rifiuta H_0, indicando quindi che c'è **autocorrelazione significativa** nei dati.
2. **Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test**, verifica se una serie temporale è **stazionaria**.
    - *Ipotesi nulla* (H_0), la serie è **stazionaria**.
    - *Ipotesi alternativa* (H_1), la serie **non** è **stazionaria**.
    - Per un *p-value piccolo* (< 0.05) si rifiuta H_0, indicando quindi che la serie **non è stazionaria**.

Ricordiamo che autocorrelazione **non implica** non stazionarietà (e viceversa) in quanto la stazionarietà riguarda proprietà generali della serie nel tempo, mentre l'autocorrelazione si concentra su pattern di dipendenza temporale osservando sotto-intervalli di lag.
Effettuiamo i Test ed osserviamo i risultati.

```
# test per la serie dello score - Neg
Box.test(negTS$score, lag = 10, type = "Ljung-Box")
negTS |> select(score) |> as.ts() |> ur.kpss() |> summary()
# test per la serie dello score - Neu
Box.test(neuTS$score, lag = 10, type = "Ljung-Box")
neuTS |> select(score) |> as.ts() |> ur.kpss() |> summary()
# test per la serie dello score - Pos
Box.test(posTS$score, lag = 10, type = "Ljung-Box")
posTS |> select(score) |> as.ts() |> ur.kpss() |> summary()
```

Score	KPSS Test (Test-Statistic)	Ljung-Box Test (p-value)
Negativo	73.8069 Non stazionario	$< 2.2e-16$ autocorrelazione presente
Neutrale	17.6576 Non stazionario	$< 2.2e-16$ autocorrelazione presente
Positivo	34.6240 Non stazionario	$< 2.2e-16$ autocorrelazione presente

Table 1: Risultati dei test KPSS e Ljung-Box per le serie temporali dei punteggi di sentiment

Dai risultati dei test si evince che le serie sono tutte **non stazionarie** e non seguono una distribuzione white noise quindi **presentano autocorrelazione** nei ritardi definiti. Dunque questo va contro l'ipotesi iniziale che c'eravamo fatto osservandole soltando, dunque esiste una **struttura sottostante alle serie** che ne descrive l'andamento nel tempo.

Questo risulta però essere un **problema** in quanto molti dei modelli statistici costruiti per lavorare su dati temporali **presuppongono la condizione di stazionalità** degli stessi. Per questo motivo dobbiamo cercare il modo per riporatare i dati verso una stazionarietà, per farlo **diminueremo la frequenza dei dati** (aumentando la granularità a *10 minuti*, *30 minuti* e *1 ora*) e considereremo non solo le serie stesse ma anche le **serie delle loro derivate (prime) temporali**. Queste misurano **la velocità di cambiamento** dei dati,

ma soprattutto rendono le serie più trattabili statisticamente, (eliminando trend e stagionalità, evidenziando cambiamenti significativi) migliorando così la modellizzazione predittiva.

Diminuiamo la frequenza di osservazione dei dati aggregandoli, per farlo utilizzermo la funzione `aggregatets(ts, FUN=\previoustick", on=\minutes", k=1)` del pacchetto *highfrequency* che consente di ottenere serie storiche omogenee, equispaziati nel tempo ed aggregate a frequenza più bassa ripetto ai dati di partenza sfruttando diversi metodi di interpolazione. Dopo aver aggregato creiamo anche le serie delle derivate temporali e visualizziamo i grafici.

3.1.1 Granularità di 10 minuti

```r
# aggreghiamo i dati ad una frequenza di osservazione del dato di 10 minuti
score10neg <- aggregateTS(negTS, alignBy = "minutes", alignPeriod = 10)
score10neu <- aggregateTS(neuTS, alignBy = "minutes", alignPeriod = 10)
score10pos <- aggregateTS(posTS, alignBy = "minutes", alignPeriod = 10)
# Calcolo delle serie delle derivate prime
deriv_neg10 <- diff(score10neg) / diff(index(score10neg))
deriv_neu10 <- diff(score10neu) / diff(index(score10neu))
deriv_pos10 <- diff(score10pos) / diff(index(score10pos))
```

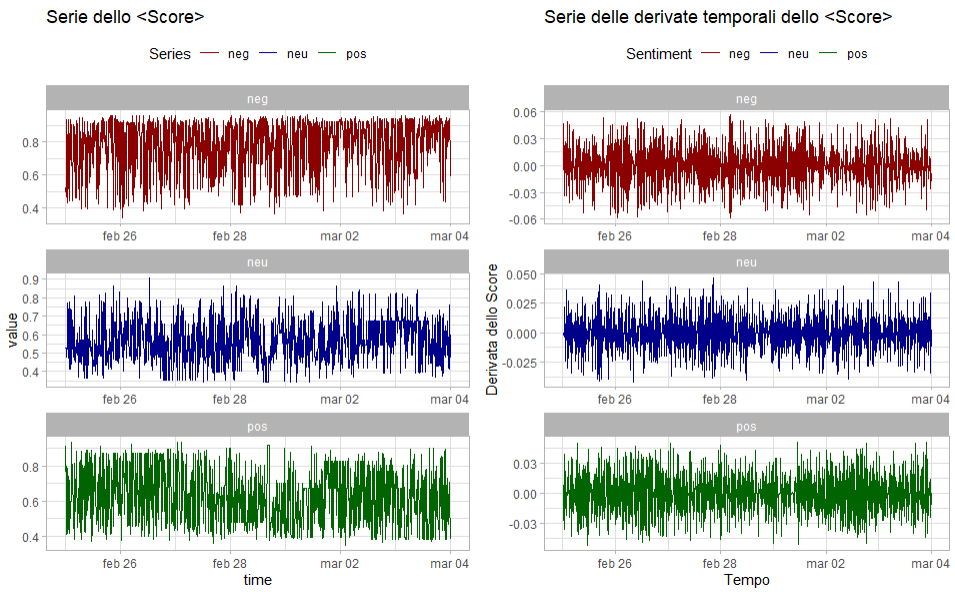


Figure 20: Confronto delle Time series della variabile Score con le derivate prime a granularità di 10 minuti

Osservando il plot, **Figura** 20, si nota come per le serie classiche il *noise* sia calato e sembra far emergere una simil **struttura ondulatoria** dei dati soprattutto per lo score neutrale, confermando la presenza di un sottostante. Mentre per le serie delle derivate prime sembrerebbe che la loro distribuzione si avvicini ad un processo white noise, ma per dirlo effettuiamo (come prima) i test.

Analisi della Stazionarietà e Autocorrelazione

Serie classica	KPSS Test (Test-Statistic)	Ljung-Box Test (p-value)
Negativo	1.6076 Non stazionario	$= 0.5361$ Autocorrelazione assente
Neutrale	1.0155 Non stazionario	$= 3.389e-06$ Autocorrelazione presente
Positivo	1.5981 Non stazionario	$= 0.01512$ Autocorrelazione presente

Table 2: Risultati dei test KPSS e Ljung-Box per lo score classico a granularità di 10 minuti

Serie derivate	KPSS Test (Test-Statistic)	Ljung-Box Test (p-value)
Negativo	0.0111 Stazionario	$< 2.2e-16$ Autocorrelazione presente
Neutrale	0.006 Stazionario	$< 2.2e-16$ Autocorrelazione presente
Positivo	0.0068 Stazionario	$< 2.2e-16$ Autocorrelazione presente

Table 3: Risultati dei test KPSS e Ljung-Box per le derivate prime a granularità di 10 minuti

Notiamo che a questa frequenza la statistica test per l'ipotesi di stazionarietà per le serie classiche è diminuita molto ma rimane ancora più grande della soglia critica del 1%. Mentre a questo livello solo la serie dello score negativa perde autocorrelazione nei ritardi. Per quanto riguarda invece la serie delle derivate temporali, queste presentano stazionarietà.

3.1.2 Granularità di 30 minuti

```
# aggreghiamo i dati ad una frequenza di osservazione del dato di 30 minuti
score30neg <- aggregateTS(negTS, alignBy = "minutes", alignPeriod = 30)
score30neu <- aggregateTS(neuTS, alignBy = "minutes", alignPeriod = 30)
score30pos <- aggregateTS(posTS, alignBy = "minutes", alignPeriod = 30)
# Calcolo delle serie delle derivate prime
deriv_neg30 <- diff(score30neg) / diff(index(score30neg))
deriv_neu30 <- diff(score30neu) / diff(index(score30neu))
deriv_pos30 <- diff(score30pos) / diff(index(score30pos))
```

Osservando il plot, **Figura** 21, i grafici delle serie classiche mostrano sempre più la struttura ondulatoria dei dati. Mentre per le serie delle derivate prime, anche graficamente somigliano sempre più ad un processo aleatorio casuale data la stazionarietà.

Analisi della Stazionarietà e Autocorrelazione

Serie classica	KPSS Test (Test-Statistic)	Ljung-Box Test (p-value)
Negativo	0.7737 Non stazionario	$= 0.07077$ Autocorrelazione assente
Neutrale	0.2137 Non stazionario	$= 0.0007944$ Autocorrelazione presente
Positivo	0.2168 Non stazionario	$= 0.357$ Autocorrelazione assente

Table 4: Risultati dei test KPSS e Ljung-Box per lo score classico a granularità di 30 minuti

Notiamo che a questa frequenza la statistica test per l'ipotesi di stazionarietà per le serie classiche continua a diminuire molto ma rimane ancora più grande della soglia critica del 1%. Mentre la serie delle derivate temporali preservano la stessa struttura dei dati mostrata nel livello di granularità di prima.

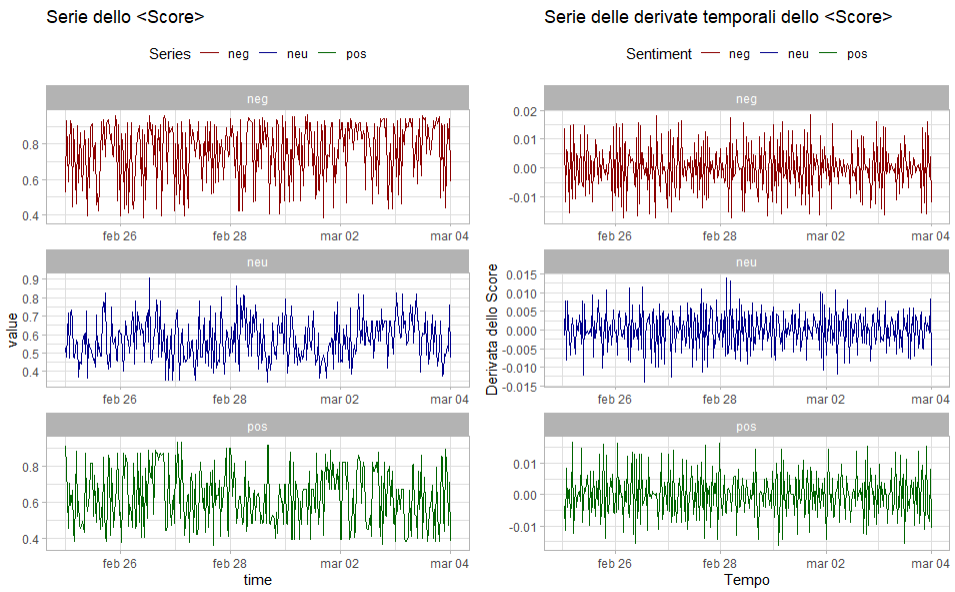


Figure 21: Confronto delle Time series della variabile Score con le derivate prime a granularità di 30 minuti

Serie derivate	KPSS Test (Test-Statistic)	Ljung-Box Test (p-value)
Negativo	0.0217 Stazionario	$< 2.2e-16$ Autocorrelazione presente
Neutrale	0.0151 Stazionario	$= 3.563e-13$ Autocorrelazione presente
Positivo	0.0145 Stazionario	$= 1.984e-12$ Autocorrelazione presente

Table 5: Risultati dei test KPSS e Ljung-Box per le derivate prime a granularità di 30 minuti

3.1.3 Granularità di 1 ora

```r
# aggreghiamo i dati ad una frequenza di osservazione del dato di 1 ora
score1hneg <- aggregateTS(negTS, alignBy = "hours", alignPeriod = 1)
score1hneu <- aggregateTS(neuTS, alignBy = "hours", alignPeriod = 1)
score1hpos <- aggregateTS(posTS, alignBy = "hours", alignPeriod = 1)
# Calcolo delle serie delle derivate prime
deriv_neg1h <- diff(score1hneg) / diff(index(score1hneg))
deriv_neu1h <- diff(score1hneu) / diff(index(score1hneu))
deriv_pos1h <- diff(score1hpos) / diff(index(score1hpos))
```

Osservando il plot, **Figura** 22, i grafici delle serie classiche hanno perso la struttura ondulatoria dei dati, denotando che abbiamo probabilmente abbassato troppo la frequenza dei dati **perdendo informazioni**.

Analisi della Stazionarietà e Autocorrelazione

Serie classica	KPSS Test (Test-Statistic)	Ljung-Box Test (p-value)
Negativo	0.3793 stazionario	$= 0.1761$ Autocorrelazione assente
Neutrale	0.1504 stazionario	$= 0.09925$ Autocorrelazione assente
Positivo	0.1444 stazionario	$= 0.4608$ Autocorrelazione assente

Table 6: Risultati dei test KPSS e Ljung-Box per lo score classico a granularità di 1 ora

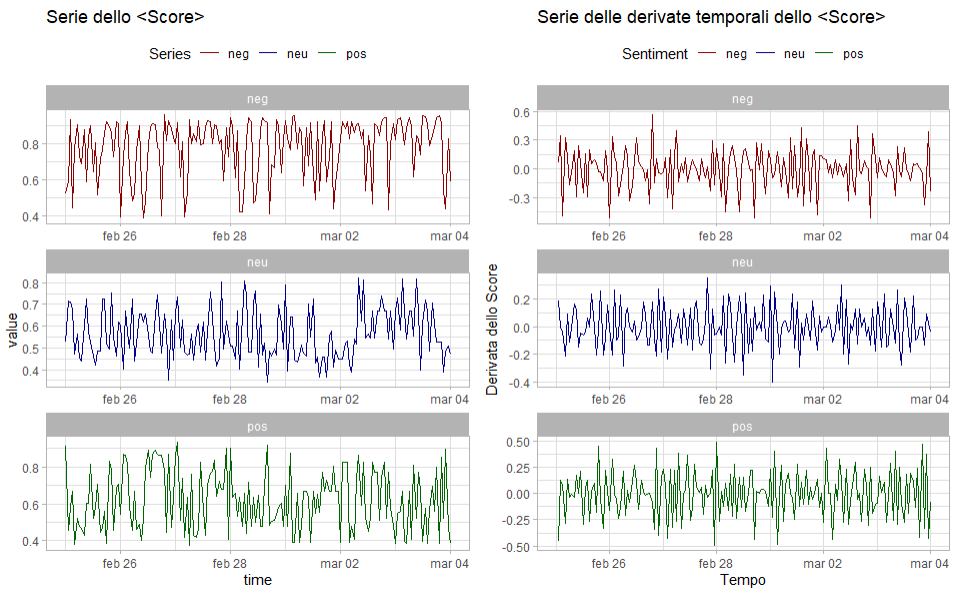


Figure 22: Confronto delle Time series della variabile Score con le derivate prime a granularità di 1 ora

Serie derivate	**KPSS Test (Test-Statistic)**	**Ljung-Box Test (p-value)**
Negativo	0.046 Stazionario	$< 2.32e-05$ Autocorrelazione presente
Neutrale	0.0198 Stazionario	$= 4.77e-06$ Autocorrelazione presente
Positivo	0.0255 Stazionario	$= 1.427e-11$ Autocorrelazione presente

Table 7: Risultati dei test KPSS e Ljung-Box per le derivate prime a granularità di 1 ora

A questa granularità tutte le serie classiche sono stazionarie e prive di autocorrelazione, questo indica che sono descrivibili come un processo **white noise**, tuttavia molto probabilmente ciò ha portato alla *perdita di informazioni* che potevano essere utili per la modellazione statistica futura. Mentre anche a questo livello, la struttura statistica delle serie delle derivate rimane invariata.

Nella sezione successiva costruiremo proprio dei modelli statistici per cercare di descrivere e fare previsioni sia delle serie classiche che delle derivate prime; ci aspettiamo che i risultati migliori si ottengano su quest'ultime in quanto hanno sempre preservato la struttura delle loro informazioni.

3.2 Non-seasonal ARIMA

ARIMA è l'acronimo di AutoRegressive Integrated Moving Avarage, mette insieme il concetto della differenziazione con i modelli autoregressivi e la media mobile.

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_p \varepsilon_{t-p} \tag{10}$$

In *backshift-notation* diventa

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(1 - B)^d y_t = c + (1 + \theta_1 B + \cdots + \theta_q B^q)\varepsilon_t \tag{11}$$

Dove y'_t è la serie differenziata (anche più di una volta), mentre i predittori sono i lag della serie e i lag-errors sempre della serie. Questa struttura prende il nome di **ARIMA(p, d, q) Model** dove

- p = ordine della parte autoregressiva
- d = gradi di differenziazione
- q = ordine della media mobile

Molti modelli fra i più conosciuti sono solo un caso specifico del modello ARIMA(p, d, q) infatti

- White Noise = ARIMA(0, 0 , 0)
- Random Walk = ARIMA(0, 1, 0)
- Autoregressione = ARIMA(p, 0, 0)
- Media Mobile = ARIMA(0, 0 , q)

Quello che faremo è implementare un modello ARIMA(p, d, q) per le singole serie dello score e delle derivate temporali dello score, dividendo come prima l'analisi in varie granularità.

3.2.1 Granularità di 10 minuti

Osserviamo come i modelli ARIMA(p, d, q) fittano sui dati.

Dati delle serie score classiche

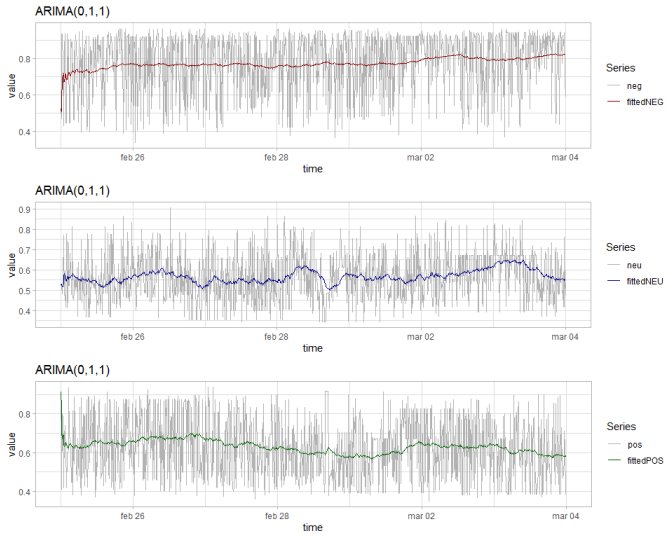


Figure 23: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali dello score (grigio)

Dati delle serie delle derivate prime dello score

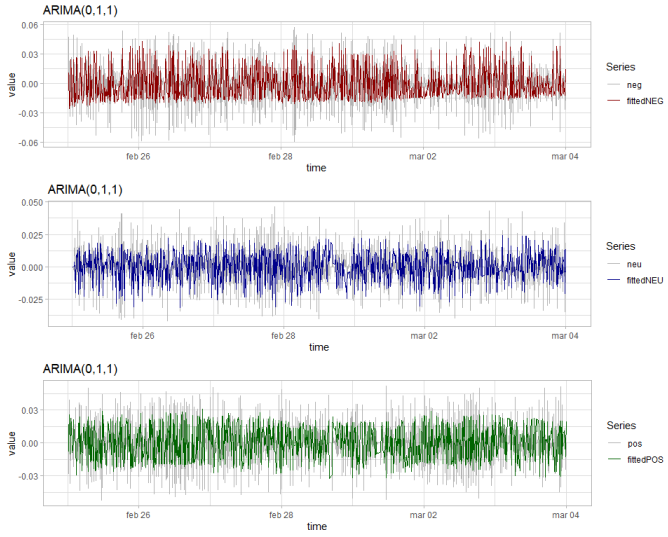


Figure 24: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali delle derivate prime (grigio)

Valutazione della bontà

Serie	ARIMA	AIC	AICc	BIC
10min_NEG	(0,1,1)	-736.2132	-736.2012	-726.3817
10min_NEU	(0,1,1)	-1430.1671	-1430.1552	-1420.3357
10min_POS	(0,1,1)	-828.6892	-828.6772	-818.8577
10min_NEG_Deriv	(0,0,1)	-5378.2247	-5378.2128	-5368.3933
10min_NEU_Deriv	(0,0,1)	-6072.1787	-6072.1667	-6062.3472
10min_POS_Deriv	(0,0,1)	-5470.7007	-5470.6888	-5460.8693

Table 8: Risultati dei modelli ARIMA con criteri di selezione AIC, AICc e BIC per granularità 10 minuti. In verde è colorata la cella con il valore migliore della metrica indicata, in rosso quella invece peggiore.

3.2.2 Granularità di 30 minuti

Osserviamo come i modelli ARIMA(p, d, q) fittano sui dati

Dati delle serie score classiche

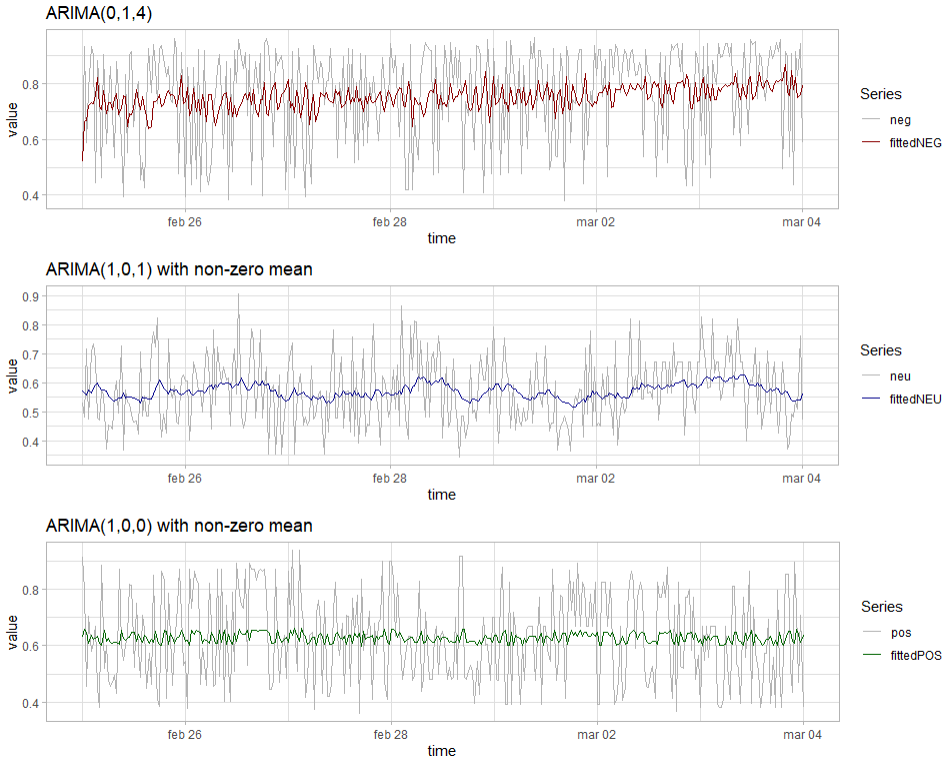


Figure 25: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali dello score (grigio)

Dati delle serie delle derivate prime dello score

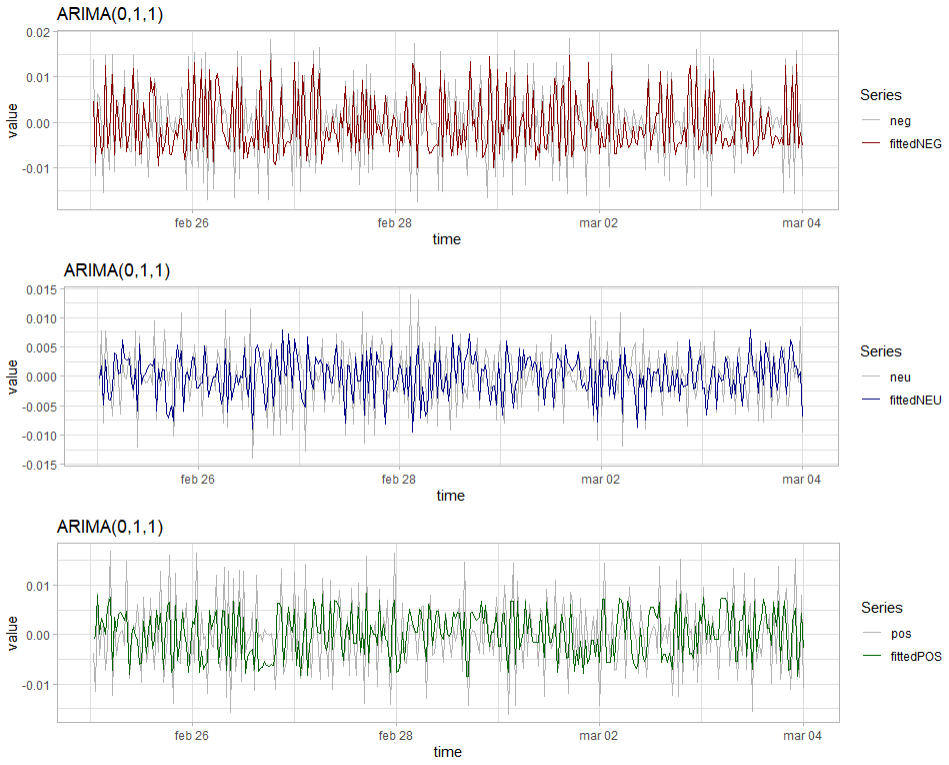


Figure 26: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali delle derivate prime (grigio)

Valutazione della bontà

Serie	ARIMA	AIC	AICc	BIC
30min_NEG	(0,1,4)	-232.2346	-232.0528	-213.1491
30min_NEU	(1,0,1)	-502.3702	-502.2498	-487.0899
30min_POS	(1,0,0)	-278.7450	-278.6729	-267.2847
30min_NEG_Deriv	(0,0,4)	-2517.8393	-2517.6574	-2498.7537
30min_NEU_Deriv	(0,0,1)	-2779.7993	-2779.7633	-2772.1651
30min_POS_Deriv	(1,0,1)	-2557.1577	-2557.0854	-2545.7064

Table 9: Risultati dei modelli ARIMA con criteri di selezione AIC, AICc e BIC per granularità 30 minuti. In verde è colorata la cella con il valore migliore della metrica indicata, in rosso quella invece peggiore.

3.2.3 Granularità di 1 ora

Osserviamo come i modelli ARIMA(p, d, q) fittano sui dati

Dati delle serie score classiche

Figure 27: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali dello score (grigio)

Dati delle serie delle derivate prime dello score

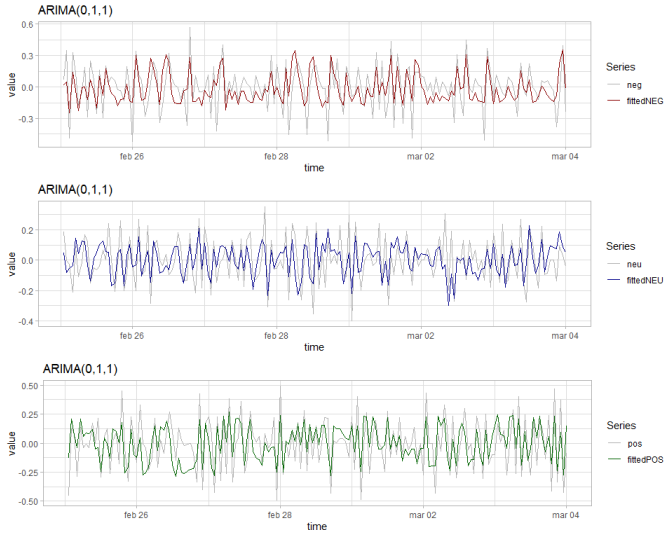


Figure 28: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali delle derivate prime (grigio)

Valutazione della bontà

Serie	ARIMA	AIC	AICc	BIC
1h_NEG	(1,0,1)	-132.1259	-131.8820	-119.6063
1h_NEU	(1,0,1)	-251.5103	-251.2664	-238.9908
1h_POS	(2,0,0)	-137.2945	-137.0506	-124.7749
1h_NEG_Deriv	(1,0,2)	-126.1111	-125.8657	-113.6153
1h_NEU_Deriv	(0,0,1)	-245.8598	-245.7871	-239.6119
1h_POS_Deriv	(0,0,1)	-128.1096	-128.0369	-121.8617

Table 10: Risultati dei modelli ARIMA con criteri di selezione AIC, AICc e BIC per granularità 1 ora. In verde è colorata la cella con il valore migliore della metrica indicata, in rosso quella invece peggiore.

Riflessioni generali:
Osservando i risultati ottenuti dalle metriche di bontà di adattamento dei modelli, possiamo affermare che a tutti i livelli di granularità:

- I dati su cui i modelli performano meglio sono le serie delle derivate temporali;
- Di tutte le serie considerate, i modelli risultano **più** efficaci quando sono costruiti sulle serie **Neutrali**;
- Di tutte le serie considerate, i modelli risultano **meno** efficaci efficaci quando sono costruiti sulle serie **Negative**;

3.3 Auto-Regressive Neural Network

Il modello **Auto-Regressive Neural Network** (ARNN) è una combinazione tra i modelli autoregressivi tradizionali e le reti neurali artificiali. La funzione che lo implementa in R è `nnetar()`, per la precisione è un modello per la previsione di serie temporali che utilizza una rete neurale feedforward con un livello nascosto.
Il modello ARNN dunque estende il classico modello autoregressivo **AR(p)** introducendo una rete neurale per apprendere le dipendenze non lineari presenti nei dati.

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}) + \epsilon_t \tag{12}$$

dove:

- y_t è il valore della serie temporale al tempo t.
- p è l'ordine del modello autoregressivo.
- $f(\cdot)$ è una funzione non lineare appresa dalla rete neurale.
- ϵ_t è il termine di errore.

Il vantaggio di usare un modello del genere rispetto ai modelli ARIMA è quello che riesce adattarsi a dinamiche più complesse rispetto agli ARIMA e supporta *bootstrap aggregation* per migliorare la robustezza delle previsioni.

Come per i modelli ARIMA anche in questo caso costruiremo un modello `nnetar()` per tutte le serie e per ogni granularità.

3.3.1 Granularità di 10 minuti

Osserviamo come i modelli `nnetar()` fittano sui dati

Dati delle serie score classiche

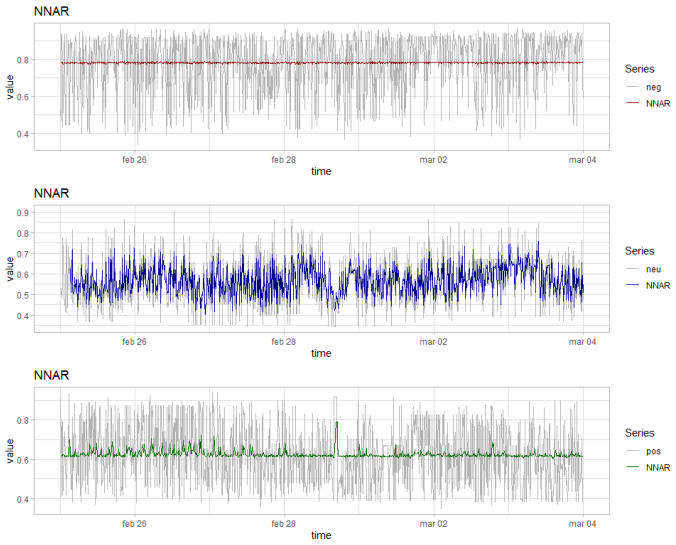


Figure 29: Grafico dei fitted values del modello `nnetar()` (colorato) con i valori reali dello score (grigio)

Dati delle serie delle derivate prime dello score

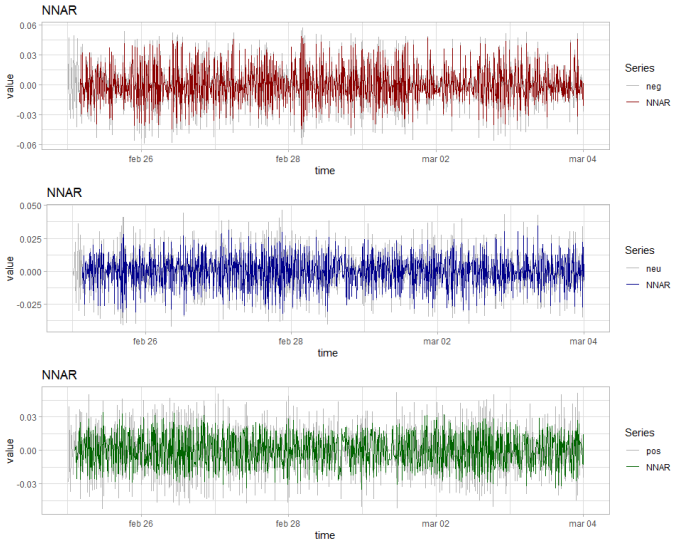


Figure 30: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali delle derivate prime (grigio)

3.3.2 Granularità di 30 minuti

Osserviamo come i modelli `nnetar()` fittano sui dati

Dati delle serie score classiche

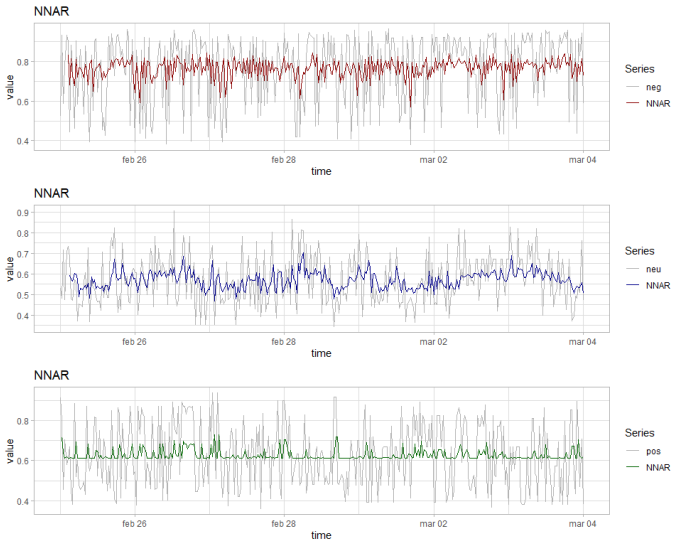


Figure 31: Grafico dei fitted values del modello `nnetar()` (colorato) con i valori reali dello score (grigio)

Dati delle serie delle derivate prime dello score

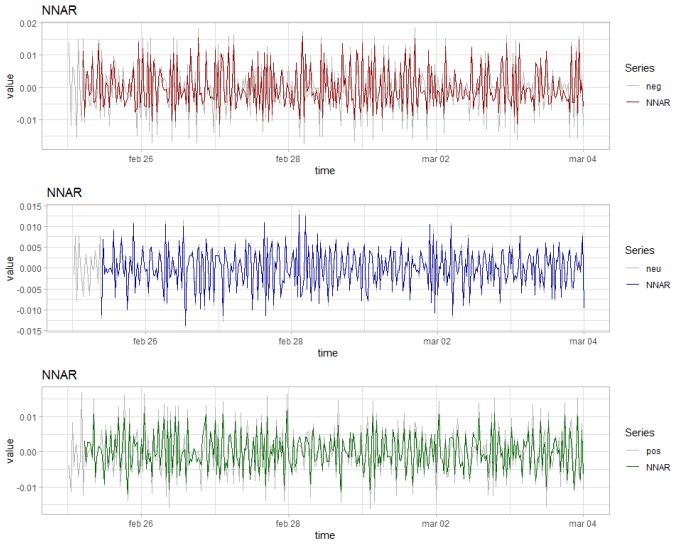


Figure 32: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali delle derivate prime (grigio)

3.3.3 Granularità di 1 ora

Osserviamo come i modelli `nnetar()` fittano sui dati

Dati delle serie score classiche

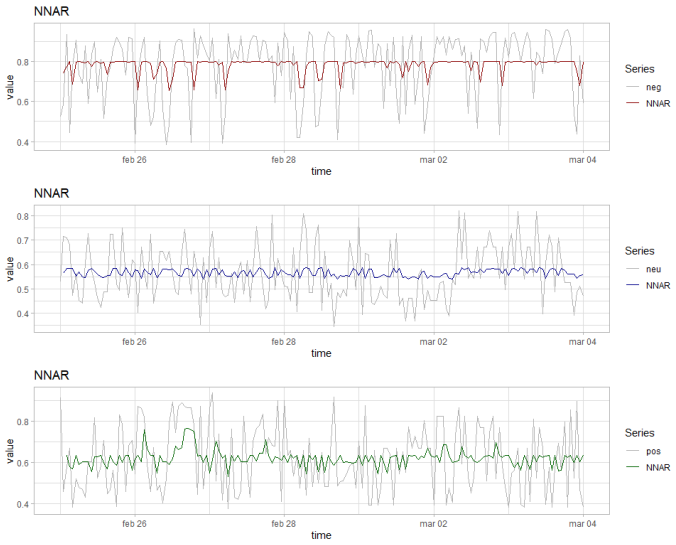


Figure 33: Grafico dei fitted values del modello `nnetar()` (colorato) con i valori reali dello score (grigio)

Dati delle serie delle derivate prime dello score

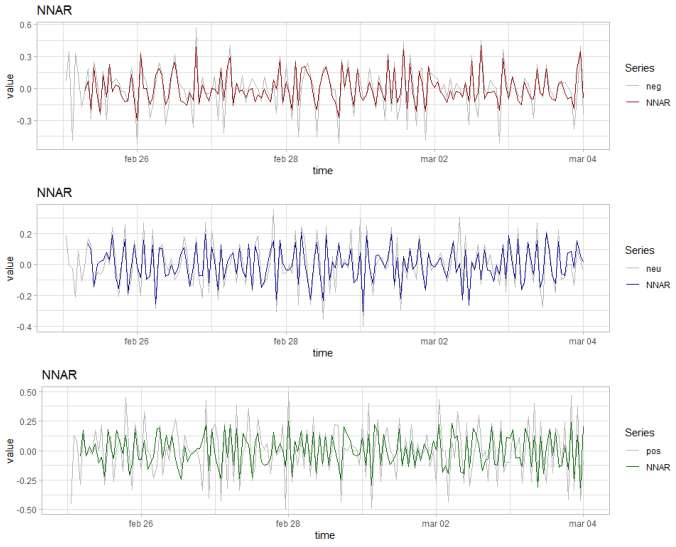


Figure 34: Grafico dei fitted values del modello ARIMA (colorato) con i valori reali delle derivate prime (grigio)

3.4 Valutazione e Confronto dei modelli

Una volta costruiti e implementati i modelli sulle serie storiche, quello che ci chiediamo è:

Research Question. (5) È possibile effettuare **previsione** della variabile Score?
Quali sono i dati migliori su cui addestrare e testare i modelli?
Quali sono i modelli che hanno i migliori risultati nel task?

Per valutare i modelli sulle time series useremo l'approccio **Train-Test Split** per ottenere una valutazione abbastanza robusta delle prestazioni dei modelli. Infatti, dividendo la serie temporale in un *training set* e un *test set* ci permette di valutare la capacità del modello di **generalizzare** su dati mai visti.

```r
# Funzione di Train-Test Split
train_test_split <- function(series, test_size) {
  n <- length(series)
  train_end <- n - test_size
  train <- series[1:train_end]
  test <- series[(train_end + 1):n]
  return(list(train = train, test = test))
}

# Funzione per calcolare le metriche dei modelli
calculate_metrics <- function(errors) {
  rmse <- sqrt(mean(errors^2, na.rm = TRUE))
  mae <- mean(abs(errors), na.rm = TRUE)
  return(tibble(RMSE = rmse, MAE = mae))
}

# Funzione per addestrare sul train set e prevedere sul test set
train_and_forecast <- function(train, test, model_type) {
  if (model_type == "ARIMA") {
    fit <- auto.arima(train, seasonal = FALSE,
                      stepwise = FALSE,
                      approximation = FALSE)
    forecasted <- forecast(fit, h = length(test))
  } else if (model_type == "NNETAR") {
    fit <- nnetar(train)
    forecasted <- forecast(fit, h = length(test))
  }
  errors <- test - forecasted$mean
  return(calculate_metrics(errors))
}
```

Listing 4: Insieme delle funzioni costruite per l'addestramento-test dei modelli e la loro valutazione sul task della previsione

```r
results_score <- list()
test_size <- 10  # Numero di istanti temporali per cui fare previsione

for (series_name in names(score_list)) {
  series <- score_list[[series_name]]

  # Fase 1: Train-test split
  split <- train_test_split(series, test_size)
  train <- split$train
  test <- split$test

  # Fase 2:  ARIMA
  arima_test_metrics <- train_and_forecast(train, test, "ARIMA")
  arima_test_metrics <- mutate(arima_test_metrics,
        Serie = series_name, Model = "ARIMA", Stage = "Test Set")


  # Fase 3: NNETAR
  nnetar_test_metrics <- train_and_forecast(train, test, "NNETAR")
  nnetar_test_metrics <- mutate(nnetar_test_metrics, Serie = series_name,
    Model = "NNETAR", Stage = "Test Set")

  # Fase 4: Risulati
  results_score[[series_name]] <- bind_rows(arima_test_metrics,
    nnetar_test_metrics)
}

# Combinazione dei risultati
final_results_score <- bind_rows(results_score)

# Ordinazione dei risultati
final_results_score <- final_results_score %>%
  arrange(Serie, Model, Stage)
```

Listing 5: Codice R che ci permette di addestrare-testare e valutare il modello ARIMA() e nnetar() per ogni serie

3.4.1 Evaluation per le serie classiche

Analizziamo i risultati dai modelli sulle serie storiche nella **Tabella** 11.

- Serie con granularità **10 minuti**
  - **score10neg**: L'`ARIMA` ha prestazioni migliori rispetto a `NNETAR`
  - **score10neu**: `NNETAR` ha un MAE inferiore, ma `ARIMA` ha un RMSE minore
  - **score10pos**: `ARIMA` performa meglio rispetto a `NNETAR`
- Serie con granularità **30 minuti**
  - **score30neg**: RMSE simile, ma MAE suggerisce una leggera superiorità di `ARIMA`
  - **score30neu**: Prestazioni molto simili tra i due modelli
  - **score30pos**: `ARIMA` si comporta leggermente meglio
- Serie con granularità **1 ora**
  - **score1hneg**: Prestazioni simili, ma `NNETAR` ha un MAE inferiore
  - **score1hneu**: `NNETAR` ha prestazioni migliori in entrambe le metriche
  - **score1hpos**: `ARIMA` ha prestazioni leggermente migliori

RMSE	MAE	Serie	Modello	Stage
0.1181	0.1037	score10neg	ARIMA	Test Set
0.1294	0.1180	score10neg	NNETAR	Test Set
0.1071	0.0925	score10neu	ARIMA	Test Set
0.1091	0.0869	score10neu	NNETAR	Test Set
0.1241	0.1097	score10pos	ARIMA	Test Set
0.1398	0.1226	score10pos	NNETAR	Test Set
0.2025	0.1711	score30neg	ARIMA	Test Set
0.2000	0.1816	score30neg	NNETAR	Test Set
0.1197	0.1043	score30neu	ARIMA	Test Set
0.1197	0.1084	score30neu	NNETAR	Test Set
0.1804	0.1631	score30pos	ARIMA	Test Set
0.1824	0.1654	score30pos	NNETAR	Test Set
0.1761	0.1483	score1hneg	ARIMA	Test Set
0.1772	0.1442	score1hneg	NNETAR	Test Set
0.1076	0.0965	score1hneu	ARIMA	Test Set
0.0976	0.0871	score1hneu	NNETAR	Test Set
0.1806	0.1610	score1hpos	ARIMA	Test Set
0.1832	0.1653	score1hpos	NNETAR	Test Set

Table 11: Risultati dei modelli ARIMA e NNETAR sulle diverse serie temporali.

I valori predetti dai migliori modelli per le serie storiche dello score nelle varie granularità temporali, sono visibili nella **Figura** 35.

Risposta. (5) Considerazioni finali
Per la maggior parte delle serie temporali, `ARIMA` sembra offrire prestazioni migliori in termini di *RMSE* e *MAE*. Tuttavia, `NNETAR` si comporta meglio con le serie score del sentiment neutrale, in particolare per *score1hneu*. Probabilmente perché, come avevamo visto, le serie neutrali presentano caratteristiche non lineari più marcate che la rete neurale riesce a catturare in maniera più accurata.
Comunque, *la differenza tra i due modelli non è particolarmente grande*, suggerendoci che entrambi hanno una capacità predittiva simile; dunque il miglior modello deve essere scelto valutando il trade-off fra:
- Natura della time series
- Costo computazionale del modello
Osservando la Figura 11 si nota come sia i modelli `ARIMA` che `NNETAR` più che effettuare una previsione vera e propria quello che fanno è *stimare il trend della serie* (dovuto probabilmente all'elevata variabilità dei valori delle time series). Possiamo concludere quindi che è **possibile stimare l'andamento futuro delle serie, ma una previsione puntuale dei valori risulta poco affidabile.**

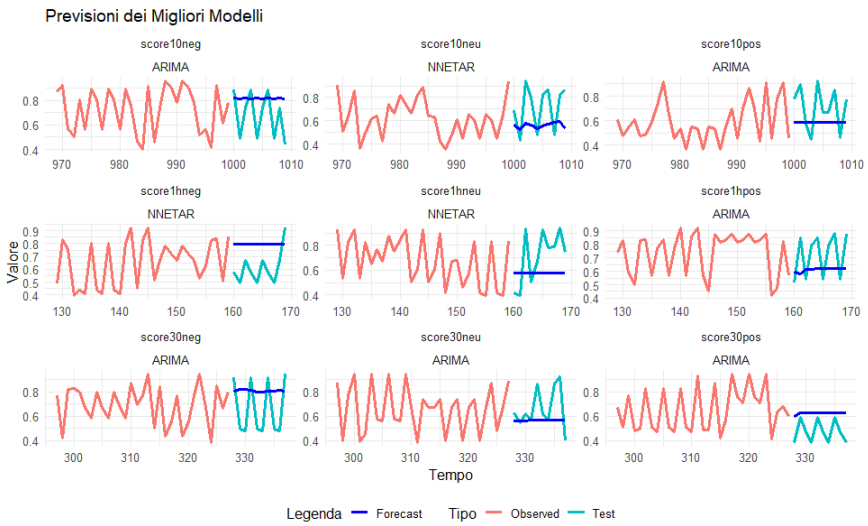


Figure 35: Grafico dei predict values dei migliori modelli di `ARIMA` e `NNETAR` per le singole serie storiche suddivisi per la frequenza di osservazione del dato

3.4.2 Evaluation per le serie delle derivate temporali

Analizziamo, anche per quanto riguarda le serie delle derivate temporali, i risultati ottenutio dai modelli.

- Serie con granularità **10 minuti**
  - **Serie deriv10neg**: ARIMA è leggermente migliore rispetto a NNETAR
  - **Serie deriv10neu**: NNETAR supera ARIMA, specialmente in termini di MAE
  - **Serie deriv10pos**: Prestazioni molto simili tra i due modelli

- Serie con granularità **30 minuti**
  - **Serie deriv30neg**: Prestazioni molto simili, con un leggero vantaggio per NNETAR
  - **Serie deriv30neu**: NNETAR ha una lieve superiorità su entrambi gli errori
  - **Serie deriv30pos**: NNETAR mostra un miglior adattamento rispetto ad ARIMA

- Serie con granularità **1 ora**
  - **Serie deriv1hneg**: ARIMA mostra prestazioni migliori
  - **Serie deriv1hneu**: ARIMA supera NNETAR nelle metriche
  - **Serie deriv1hpos**: ARIMA è leggermente migliore

Nota. I risultati della tabella 12 così come quelli della tabella 11, non rappresentano una valutazione robusta al 100% della capacità predittiva dei modelli in quanto potrebbe essere più efficiente valutarli su diversi range temporali e con tecniche più forti come Cross-Validation.

RMSE	MAE	Serie	Model	Stage
0.0115	0.0090	deriv10neg	ARIMA	Test Set
0.0136	0.0108	deriv10neg	NNETAR	Test Set
0.0186	0.0145	deriv10neu	ARIMA	Test Set
0.0161	0.0118	deriv10neu	NNETAR	Test Set
0.0127	0.0097	deriv10pos	ARIMA	Test Set
0.0128	0.0102	deriv10pos	NNETAR	Test Set
0.0112	0.0099	deriv30neg	ARIMA	Test Set
0.0109	0.0094	deriv30neg	NNETAR	Test Set
0.0049	0.0038	deriv30neu	ARIMA	Test Set
0.0047	0.0034	deriv30neu	NNETAR	Test Set
0.0085	0.0072	deriv30pos	ARIMA	Test Set
0.0075	0.0061	deriv30pos	NNETAR	Test Set
0.1938	0.1350	deriv1hneg	ARIMA	Test Set
0.2266	0.1798	deriv1hneg	NNETAR	Test Set
0.1078	0.0845	deriv1hneu	ARIMA	Test Set
0.1151	0.0987	deriv1hneu	NNETAR	Test Set
0.3033	0.2596	deriv1hpos	ARIMA	Test Set
0.3490	0.3060	deriv1hpos	NNETAR	Test Set

Table 12: Risultati dei modelli ARIMA e NNETAR sulle serie delle derivate temporali

Risposta. (5) Conclusioni:
Osservando la tabella 12 si nota come, a differenza dei risultati di prima, fra le diverse granularità esista una tipologia di modello che performa meglio e dunque che è da preferire. Infatti **i modelli `ARIMA` risultano avere prestazioni migliori** quando la frequenza di osservazione del dato è stata aggregata **ad 1 ora**.
Mentre, ad una frequenza più alta, **alla granularità di 30 minuti sono i modelli `NNETAR` a mostrare prestazioni migliori**. Per quanto riguarda le serie osservate a 10 minuti, i due modelli hanno prestazioni molto simili (con piccole differenze in alcune serie), dunque i risultati non ci indicano un modello chiaramente superiore ad un altro.

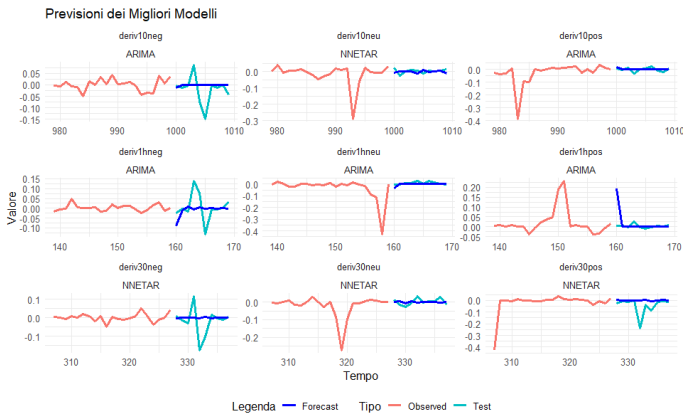


Figure 36: Grafico dei predict values dei migliori modelli di `ARIMA` e `NNETAR` per le singole serie storiche suddivisi per la frequenza di osservazione del dato

4 Dataset Sintetico

In questa sezione esploreremo la possibilità di utilizzare modelli linguistici di grandi dimensioni (LLM) per la **generazione di dati sintetici**. L'obiettivo è valutare se questi modelli riescano a creare dataset che riflettano la struttura di dati reali, a partire da **prompt** ben formulati che ne descrivano le caratteristiche più importanti.
Per verificare questa ipotesi, abbiamo inizialmente analizzato il dataset reale come descritto nella **Sezione 2**, ricavandone quante più informazioni possibili. Successivamente, abbiamo elaborato un prompt dettagliato, che avesse come target quello di guidare l'LLM nella generazione di un dataset sintetico che conservi la struttura del dataset originale, ma con un numero **ridotto** di osservazioni.
L'idea alla base di questo approccio è che gli LLM potrebbero rappresentare uno strumento efficace per la creazione di dataset *gemelli*, mantenendo la coerenza con i dati originali, ma riducendo al contempo i costi computazionali del dataset originale.

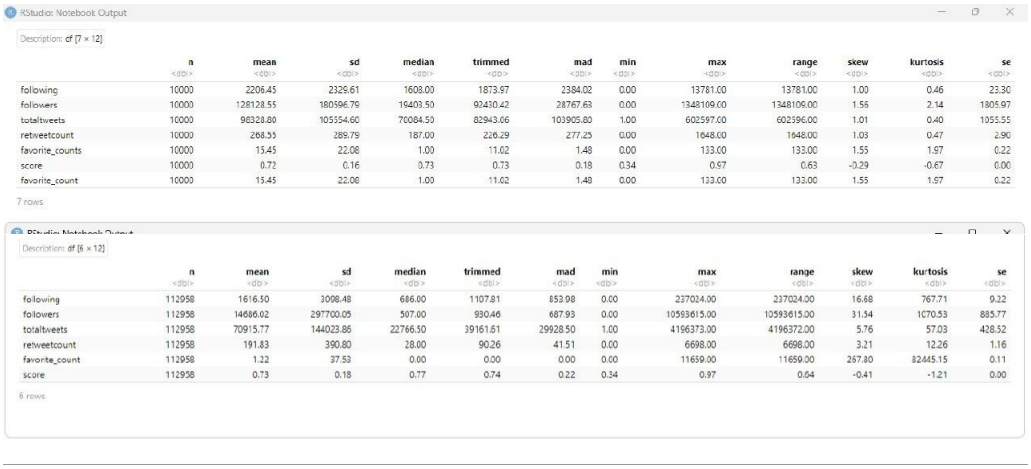
4.1 Struttura del Prompt

Il prompt è stato strutturato in modo chiaro e dettagliato per guidare il modello LLM nella generazione di un dataset sintetico coerente con quello reale. Le sezioni principali includono:

- **Obiettivo**: Una spiegazione chiara del compito richiesto al modello, specificando l'importanza di mantenere la struttura del dataset originale e le sue proprietà statistiche.
- **Struttura del dataset originale**: Descrizione dettagliata delle variabili presenti, con distinzione tra tipi di dati (caratteri, numerici, booleani, categoriali e date).
- **Analisi statistica**: Riassunto delle principali caratteristiche delle variabili numeriche (media, deviazione standard, mediana, skewness, kurtosis, ecc.), delle frequenze delle variabili categoriche e booleane, e delle correlazioni tra le variabili.
- **Vincoli sulla generazione**: Specifiche per garantire che i dati generati rispettino le distribuzioni e le relazioni presenti nel dataset originale.

Nota. Per osservare il prompt effettivo che abbiamo passato all'LLM, è presente un file `.txt` nella cartella che lo contiene interamente.

4.2 Analisi descrittiva



RStudio: Notebook Output

Description: df [7 × 12]

	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
following	10000	2206.45	2325.61	1608.00	1873.97	2384.02	0.00	13781.00	13781.00	1.00	0.46	23.30
followers	10000	128128.55	180596.79	16403.50	92430.42	28767.63	0.00	1348109.00	1348109.00	1.55	2.14	1805.97
totaltweets	10000	98328.80	105554.60	70084.50	82943.06	103905.80	1.00	602597.00	602596.00	1.01	0.40	1055.55
retweetcount	10000	268.55	289.79	187.00	226.29	277.25	0.00	1648.00	1648.00	1.03	0.47	2.90
favorite_counts	10000	15.45	22.08	1.00	11.02	1.48	0.00	133.00	133.00	1.55	1.97	0.22
score	10000	0.72	0.16	0.73	0.73	0.18	0.34	0.97	0.63	-0.29	-0.67	0.00
favorite_count	10000	15.45	22.08	1.00	11.02	1.48	0.00	133.00	133.00	1.55	1.97	0.22

7 rows

Description: df [6 × 12]

	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
following	112958	1616.50	3098.48	686.00	1107.81	853.98	0.00	237024.00	237024.00	16.68	767.71	9.22
followers	112958	14686.02	297700.05	507.00	930.46	687.93	0.00	10593615.00	10593615.00	31.54	1070.53	885.77
totaltweets	112958	70915.77	144023.86	22766.50	39161.61	29928.50	1.00	4196373.00	4196372.00	5.76	57.03	428.52
retweetcount	112958	191.83	390.80	28.00	90.26	41.51	0.00	6698.00	6698.00	3.21	12.26	1.16
favorite_count	112958	1.22	37.53	0.00	0.00	0.00	0.00	11659.00	11659.00	267.80	82445.15	0.11
score	112958	0.73	0.18	0.77	0.74	0.22	0.34	0.97	0.64	-0.41	-1.21	0.00

6 rows

Figure 37: Risulatati delle metriche estratte dal dataset sintetico

L'analisi descrittiva del dataset generato dall'LLM evidenzia che il modello sembra produrre dati **più regolari** e meno variabili, con una *minore dispersione*. Variabili come `followers, retweet count e favorite count` risultano più uniformi, suggerendoci che l'LLM abbia appreso sì le tendenze **generali** ma senza riprodurre pienamente la **complessità** e l'**eterogeneità** dei dati reali.

Un'altro aspetto importante è la tendenza dell'LLM di generare valori **medi**, attenuando la presenza di distribuzioni fortemente sbilanciate. Questo si traduce in una **minore asimmetria**e in una varianza ridotta rispetto al dataset originale.

	Variable	Mean	SD
	<chr>	<dbl>	<dbl>
following	following	2.206448e+03	2.329609e+03
followers	followers	1.281286e+05	1.805968e+05
totaltweets	totaltweets	9.832880e+04	1.055546e+05
retweetcount	retweetcount	2.685537e+02	2.897879e+02
favorite_counts	favorite_counts	1.545150e+01	2.207848e+01
score	score	7.237027e-01	1.645885e-01
favorite_count	favorite_count	1.545150e+01	2.207848e+01

Volevamo verificare analiticamente a quanto corrispondesse questa differenza fra le variabili, quindi calcolato di quanto si distanziasse in media i valori sintetici da quelli reali. Si osserva, **Figura** 4.2, che la deviazione standard delle variabili `followers` e `total tweets` nel dataset sintetico sono abbastanza alte, ciò rafforza l'idea di una distribuzione più ampia e **meno concentrata**. D'altro canto però, per variabili come `retweetcount` e `favorite count`, i valori medi sono più simili tra i due dataset, questo potrebbe indicare che si è avvicinato alla tendenza originale, ma con una lieve sovrastima.

Queste differenze ci suggeriscono che, sebbene il dataset sintetico possa essere utile per analisi esplorative preliminari, presenta limiti nella sua capacità di replicare fedelmente le proprietà statistiche del dataset reale.

4.2.1 Distribuzione dei dati

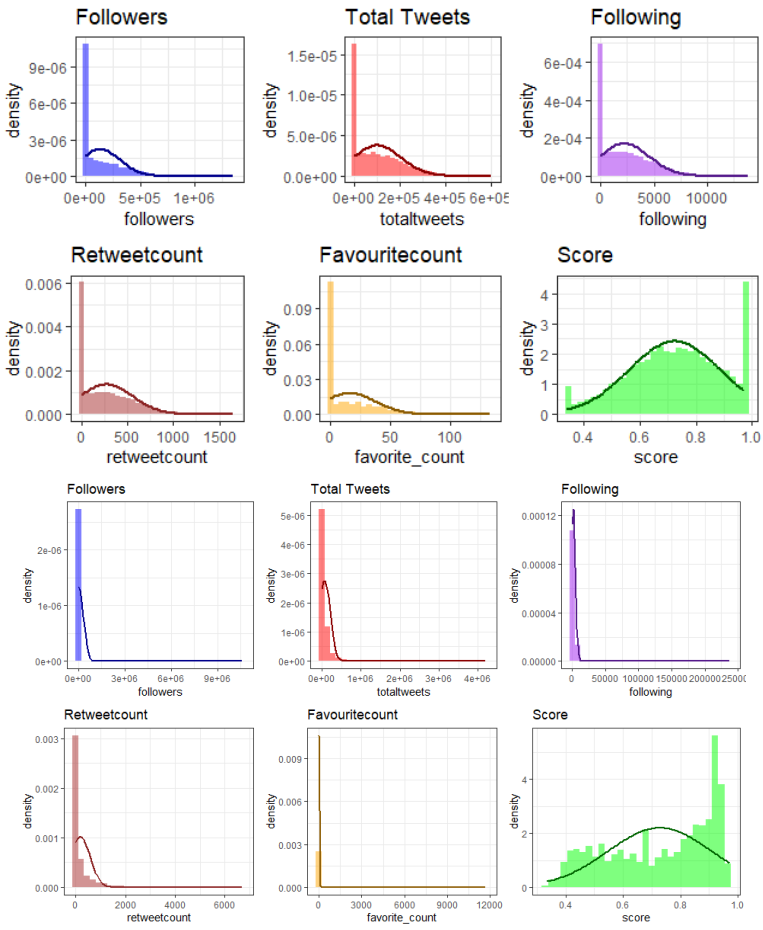


Figure 38: Confroto fra gli istogrammi ottenuti dal dataset sintetico (sopra) con quelli originali (in basso)

Osserviamo, **Figura** 38, graficamente quindi i dati sintetici che ci ha generato l'LLM confrontandoli con quelli ottenuti nella Sezione 2. Un aspetto evidente che notiamo dai grafici è proprio la tendenza del modello a produrre **distribuzioni più regolari** e prive di code lunghe. In particolare, per variabili come `followers` e `retweetcount`, le distribuzioni risultano più compatte rispetto ai dati reali. Un'altra osservazione riguarda i `favorite count` e lo `score`, infatti mentre nei dati reali queste variabili mostrano distribuzioni con picchi accentuati, nei dati sintetici appaiono più appiattite. Questo indica che l'LLM ha teso a **centralizzare i valori** attorno alle medie dei dati.

Nel complesso, possiamo affermare che il modello riesce a catturare **la struttura generale delle variabili** ma con una **semplificazione** che riduce la rappresentazione di eventi anomali e della naturale variabilità dei dati reali.

4.2.2 Gestione degli Outlier

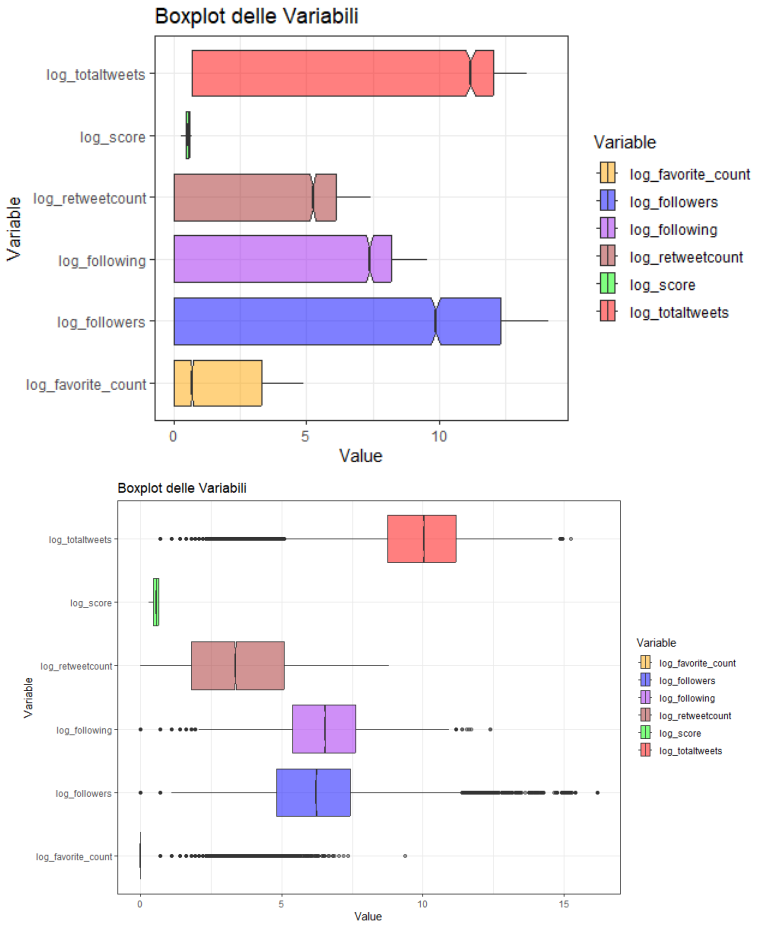


Figure 39: Confronto boxplot ottenuti dal dataset sintetico (sopra) rispetto a quelli originali (sotto)

Come abbiamo visto LLM sembrerebbe aver generato meno valori estremi, quindi proprio come fatto con il dataset originale, osserviamo i dati usando `Boxplot` per avere una visualizzazione anche degli outliers, **Figura** 39. I grafici mostrano ciò che ci aspettavamo, ossia un range più ristretto e una minore variabilità. Infatti, la riduzione della varianza implica meno valori con caratteristiche estreme.
Per il dataset sintetico inoltre, possiamo notare che i `boxplot` presentano delle pieghe evidenti nei box, cosa che invece nei grafici del dataset originale non sono presenti. Questa differenza è dovuta all'uso dell'opzione notch = TRUE nel codice, la quale aggiunge una tacca (per l'appunto il notch) ai boxplot per rappresentare l'**intervallo di confidenza della mediana**. Dunque, *le tacche saranno tanto più visibili quanto la distribuzione dei dati è regolare*; i dati sintetici mostrano un notch più evidente ma non così grande.
Abbiamo visto graficamente che nei dati sintetici ci siano pochi outliers, identifichiamoli analiticamente e contiamoli.

	Variable	IQR_Outliers	ZScore_Outliers	Hampel_Outliers
	<chr>	<int>	<int>	<int>
following	following	74	77	109
followers	followers	364	144	3809
totaltweets	totaltweets	71	73	129
retweetcount	retweetcount	75	79	178
favorite_counts	favorite_counts	429	143	4523
score	score	0	0	0
favorite_count	favorite_count	429	143	4523
log_followers	log_followers	0	0	0
log_following	log_following	0	0	3027
log_totaltweets	log_totaltweets	0	0	3171
log_retweetcount	log_retweetcount	0	0	0
log_favorite_count	log_favorite_count	0	0	1342
log_score	log_score	0	0	0

13 rows

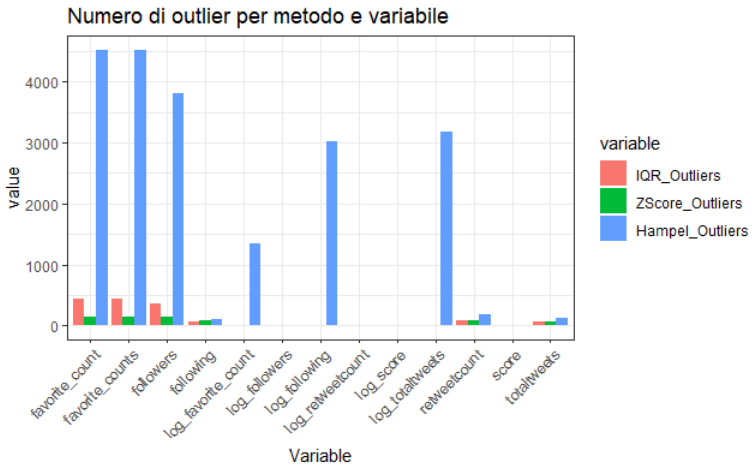


Figure 40: Numero di outlier estrapolati dal dataset sintetico

L'analisi con i tre metodi di estrapolazione (IQR, ZScore e Hampel Filter) conferma che il numero di outliers è molto minore rispetto a quello che avevamo calcolato per il dataset originale, tebella 5. Questo comportamento può rappresentare un limite quando si cerca di modellare fenomeni in cui gli outlier giocano un ruolo **chiave**.

4.2.3 Analisi della Normalità

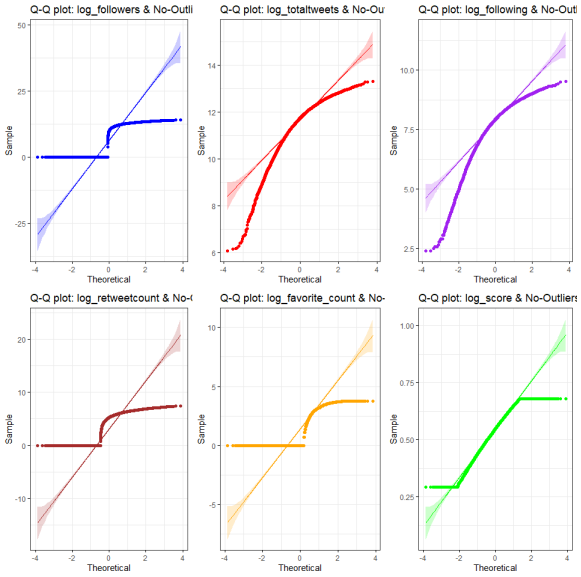


Figure 41: QQ-Plot delle variabili del dataset sintetico post eliminazione outliers

Già dalla visualizzazione degli istogrammi si nota che i dati generati non seguano la distribuzione Normale, il che è un bene in quanto nemmeno i dati originali lo fanno. Tuttavia, usiamo i `QQ-plot` dei dati logaritmici non solo per confrontarli sempre con quelli originali, **Figura** 7, ma anche per osservare la distribuzione dei quantili alle code. Infatti un'aspetto che si nota nella **Figura** 41, è la presenza di segmenti piatti nei quantili più estremi, segno ulteriore che l'LLM tende a limitare la concentrazione dei valori nelle code.

4.2.4 Osserviamo le relazioni lineari nei dati

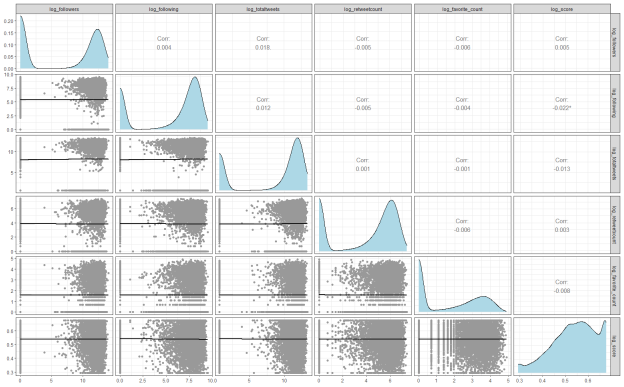


Figure 42: Scatterplot estrapolato dalle variabili del dataset sintetico

Il risultato della **Figura** 42, mette in luce una **problematica** sostanziale. Se nel dataset originale, si osservano chiaramente relazioni lineari tra le variabili, nel dataset generato dall'LLM, nonostate nel prompt fossero indicate le correlazioni originali, queste relazioni

risultano quasi completamente **attenuate**. Sebbene le distribuzioni marginali appaiano simili, la dispersione dei punti nei grafici di correlazione suggerisce una perdita di struttura nelle dipendenze tra le variabili. Questo implica che durante la fase di generazione dei dati, LLM non riesce a rappresentare i legami lineari fra le variabili e quindi un'ipotetica analisi più complessa usando modelli statistici verrebbe resa vana.

4.3 Esempio di regressione lineare

```
Call:
lm(formula = score ~ ., data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.39694 -0.11604  0.00783  0.13042  0.25764

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          0.7339159  0.0063686 115.240   <2e-16 ***
log_followers        0.0001406  0.0002724   0.516   0.6058
log_following       -0.0010198  0.0004521  -2.256   0.0241 *
log_totaltweets     -0.0004382  0.0003223  -1.360   0.1740
log_retweetcount     0.0002051  0.0005979   0.343   0.7316
log_favorite_count  -0.0007105  0.0009739  -0.730   0.4656
is_retweetTRUE      -0.0026219  0.0043634  -0.601   0.5479
is_quote_statusTRUE -0.0022038  0.0039905  -0.552   0.5808
sentimentneu         0.0040969  0.0046587   0.879   0.3792
sentimentpos         0.0018061  0.0044919   0.402   0.6876
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1646 on 9990 degrees of freedom
Multiple R-squared:  0.0009394, Adjusted R-squared:  3.93e-05
F-statistic: 1.044 on 9 and 9990 DF,  p-value: 0.4019
```

Figure 43:

Infatti, come ci aspettavamo, dall'analisi della regressione lineare sui dati generati dall'LLM si evidenzia un'importante limitazione: la regressione non riesce a catturare nessun legame. Questo si traduce in una **perdita di struttura nei rapporti tra le variabili**, con correlazioni più deboli e distorte rispetto alla realtà.
Di conseguenza, la mancanza di coerenza nelle relazioni lineari compromette sia la capacità dei modelli di regressione di spiegare le variabili ma anche di individuare eventuali pattern significativi.

4.4 Conclusioni finali

Research Question. (6) Il dataset sintetico riesce a replicare le proprietà statistiche e strutturali di quello reale? Può essere usato come alternativa valida per analisi esplorative o modellizzazione?

Risposta. (6) L'analisi del dataset *sintetico* generato dall'LLM ha evidenziato sia punti di forza che limitazioni. Da un lato, LLM è riuscito a riprodurre in modo coerente la **struttura generale** dei dati originali, generando valori con **distribuzioni plausibili** ; d'altra parte l'LLM tende a semplificare le distribuzioni. Questa **tendenza alla semplificazione** può avere implicazioni importanti, infatti la perdita di variabilità potrebbe **limitare la sua applicabilità** in scenari in cui la presenza di outlier e la **complessità** della distribuzione sono elementi chiave.
Di conseguenza, possiamo dire che l'utilizzo dei dataset sintetici generati dai Large Language Model, dipende dallo scopo dell'analisi che si vuole fare.
Se l'obiettivo è condurre un'*analisi esplorativa preliminare* o creare un **dataset di supporto** per test e simulazioni, allora il dataset sintetico può essere una risorsa utile, grazie soprattutto alla **riduzione del rumore** nei dati. Mentre, se si ha intenzione di usarlo per studi che richiedono un'elevata fedeltà statistica, quindi dove si necessità di catturare fenomeni complessi, allora il dataset sintetico non può essere sufficientemente rappresentativo.