Università di Pisa

SCUOLA DI INGEGNERIA

Corso di Laurea in Artificial Intelligence and Data Engineering

Statistica

Candidato
Marco Parola

Relatore
Prof. Romito Marco

Anno Accademico 2019–2020

Indice

1 Introduzione

La compagnia californiana Uber, con sede a San Francisco, fornisce un servizio di trasporto automobilistico privato attraverso un'applicazione mobile, che mette in collegamento diretto passeggeri e autisti.
L'analisi sviluppata nel seguito, deve supportare le decisioni legate all'organizzazione dei turni della compagnia, in particolare non devono essere sprecate forze lavoro quando non è necessario e, allo stesso tempo, non deve essere offerto un servizio scadente, causato dalla mancanza di autisti.

2 Preprocessing dei dati

Il dataset utilizzato in questa analisi è stato costruito partendo da un dataset trovato sulla piattaforma di data science al seguente link https:www.kaggle.comfivethirtyeight/uber-pickups-in-new-york-city#other-American_B01362.csv; esso descrive le corse delle autovetture, che lavorano per l'azienda nei primi anni di servizio nel trimestre luglio-settembre 2014. In particolare ogni osservazione identifica una tratta effettuata nella città di New York, con il relativo timestamp.
Il nuovo dataset è stato generato usando un programma java, in cui ogni osservazione raggruppa le corse di ogni ora ed è descritto da datetime e numero di corse: la classe TestScanner, oltre a fare una conversione di formato del timestamp, legge la prima riga del file iniziale e salva in una variabile l'ora (HH) in cui è stata effettuata la prima corsa, dopo di che scorre riga per riga il resto del file e se l'ora della corsa in esame è uguale all'ora della variabile che contiene l'orario, incrementa un contatore, altrimenti azzera il contatore, aggiorna la variabile con la nuova ora e scrive una riga sul nuovo file, contenente ora e contatore. Dunque ogni giornata è descritta da 24 osservazioni, una per ora.
Numero di osservazioni: 2207 Numero di fattori: 2

- datetime, data e ora del servizio
- num, numero di corse effettuate durante l'ora

3 Valutazione periodicità

In prima battuta è necessario comprendere se i dati possiedono una periodicità ed in caso affermativo stimarla, dunque calcoliamo e grafichiamo la funzione di autocorrelazione.

```
uber = read.csv("tabella.csv");
uber.ts = ts(uber$num, frequency = 168);
plot(uber.ts);
plot(acf(uber$num, lag=200));
```

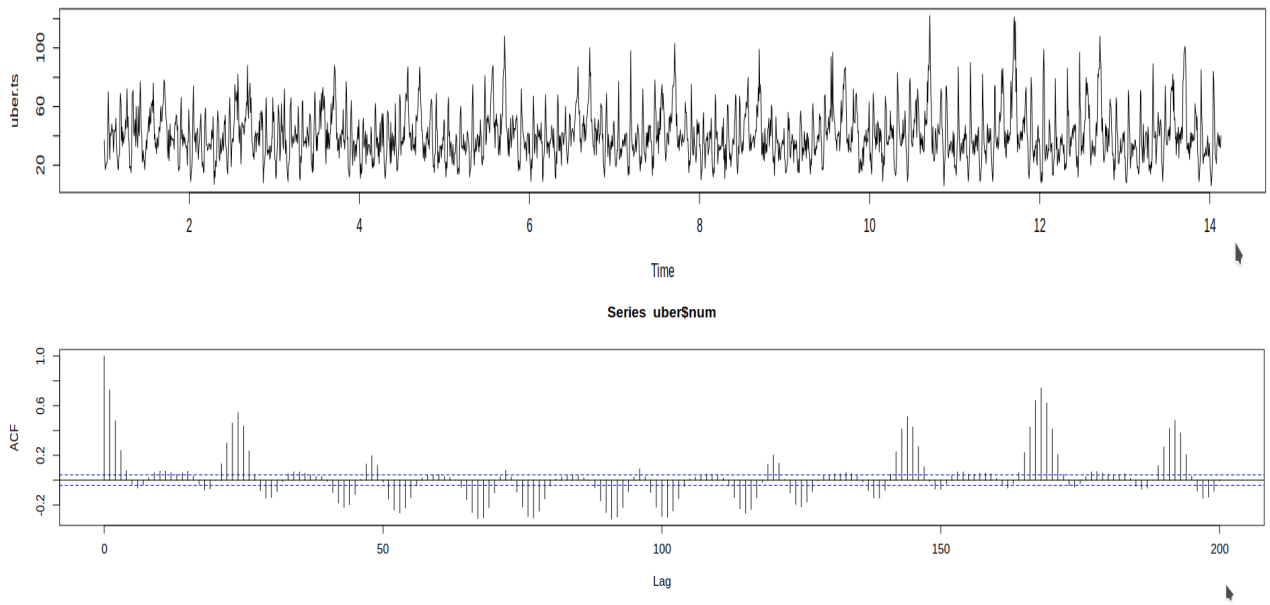

Figura 1: Andamento della serie (sopra) e funzione di autocorrelazione(sotto)

Visualizzando il grafico dell'andamento della serie in figura 1 si notano alcuni picchi con una certa periodicità, associabili all'ora con più corse durante la giornata. Inoltre si nota che ogni 7 picchi, due hanno una prominenza maggiore, dunque possiamo associarli al sabato e alla domenica.
Dal grafico dell'autocorrelazione si nota che i picchi predominanti si trovano in corrispondenza del valore 168, che non è un valore casuale, ma è ottenuto dalla moltiplicazione di 24 (osservazioni giornaliere) e 7 (giorni alla settimana), dunque la periodicità che andremo a considerare è relativa ad una settimana.
Altri valori con un alto valore della funzione di autocorrelazione si trovano in corrispondenza dell'indice 24 (periodo giornaliero). Infine gli altri due picchi evidenti si trovano in corrispondenza di 144 e 192, quando il sabato viene confrontato con la domenica e viceversa; in generale sono casi non interessanti.

4 Modellazione fenomeno

Lo step da eseguire, dopo aver fissato la frequenza, ovvero il numero di osservazioni presenti in una finestra temporale, è il calcolo di un modello che descriva il fenomeno.

4.1 Modello addittivo

Il primo modello applicato è il modello additivo.

```
uber.ts = ts(uber$num, frequency = 168);
uber.ts.da = decompose(uber.ts, type="additive");
plot(uber.ts.da);
```

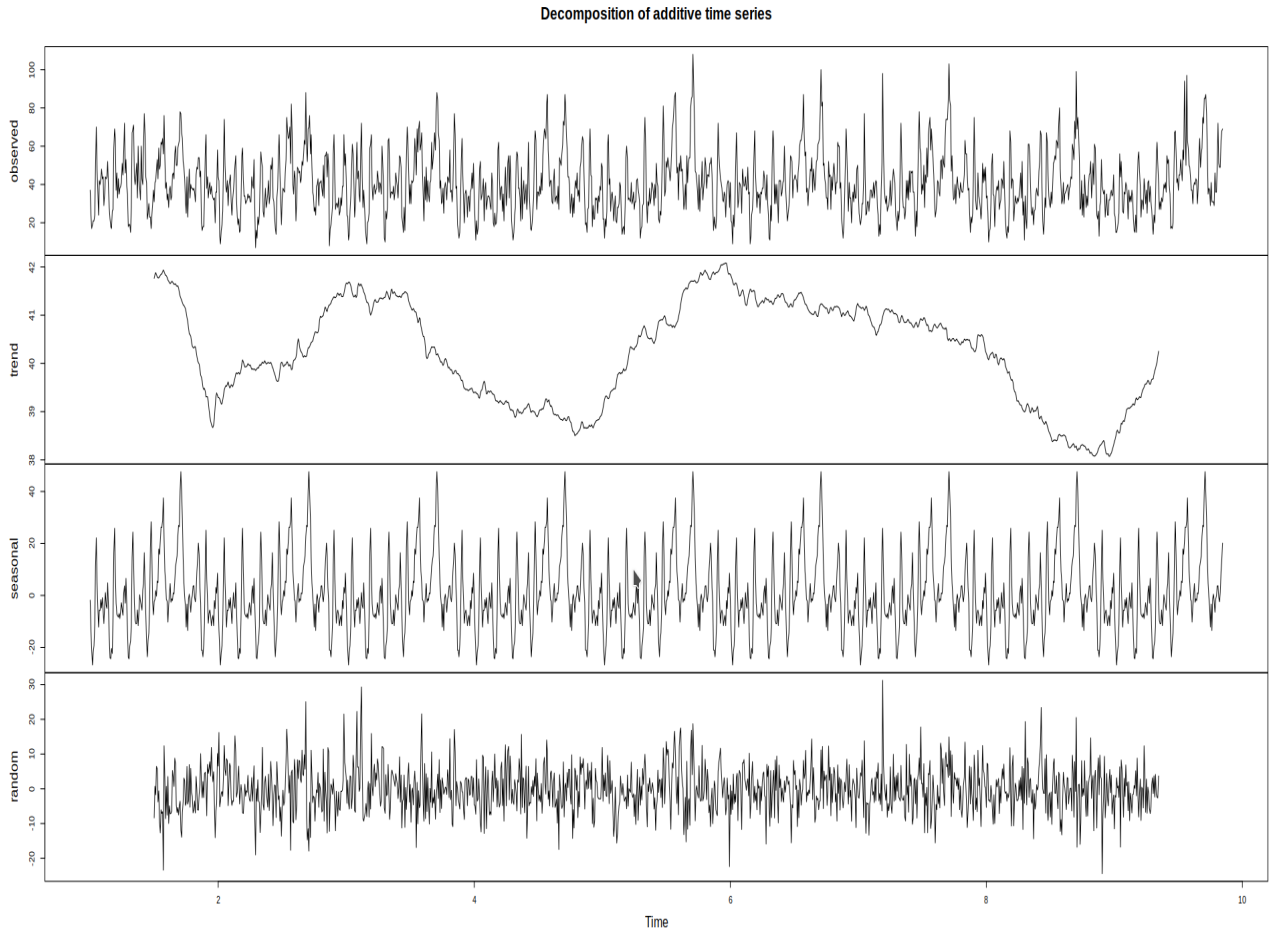

Figura 2: Serie decomposta

La presenza di una periodicità è confermata, come mostra la figura 2, dal confronto del grafico della stagionalità con quello del rumore ottenuto dalla decomposizione; infatti l'ordine di grandezza del primo è maggiore del secondo.

```
plot(uber.ts.dc$seasonal);
lines(uber.ts.dc$random, col="red");
```

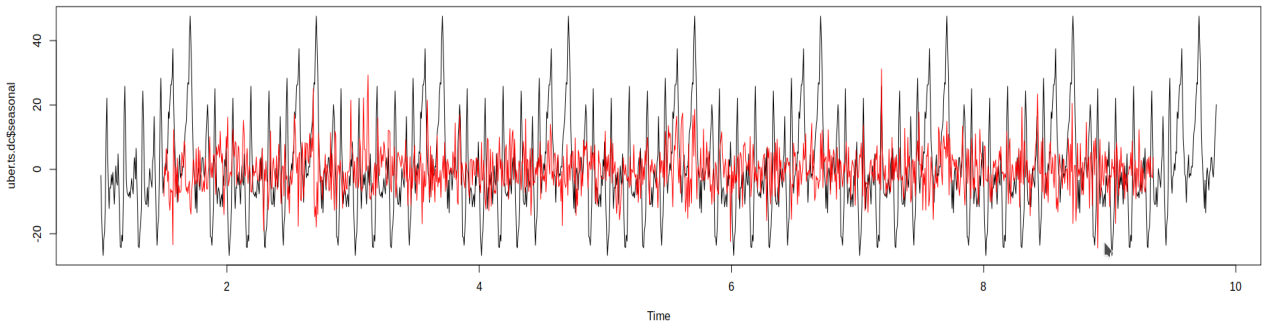

Figura 3: Confronto stagionalità rumore

Inoltre si nota come il modello ottenuto sia buono a prima vista, perchè l'ordine di grandezza del grafico della stagionalità è paragonabile a quello osservato.

4.2 Modello moltiplicativo

Il secondo modello applicato è il modello moltiplicativo.

```
uber.ts.dm = decompose(uber.ts, type="multiplicative");
plot(uber.ts.dm);
```

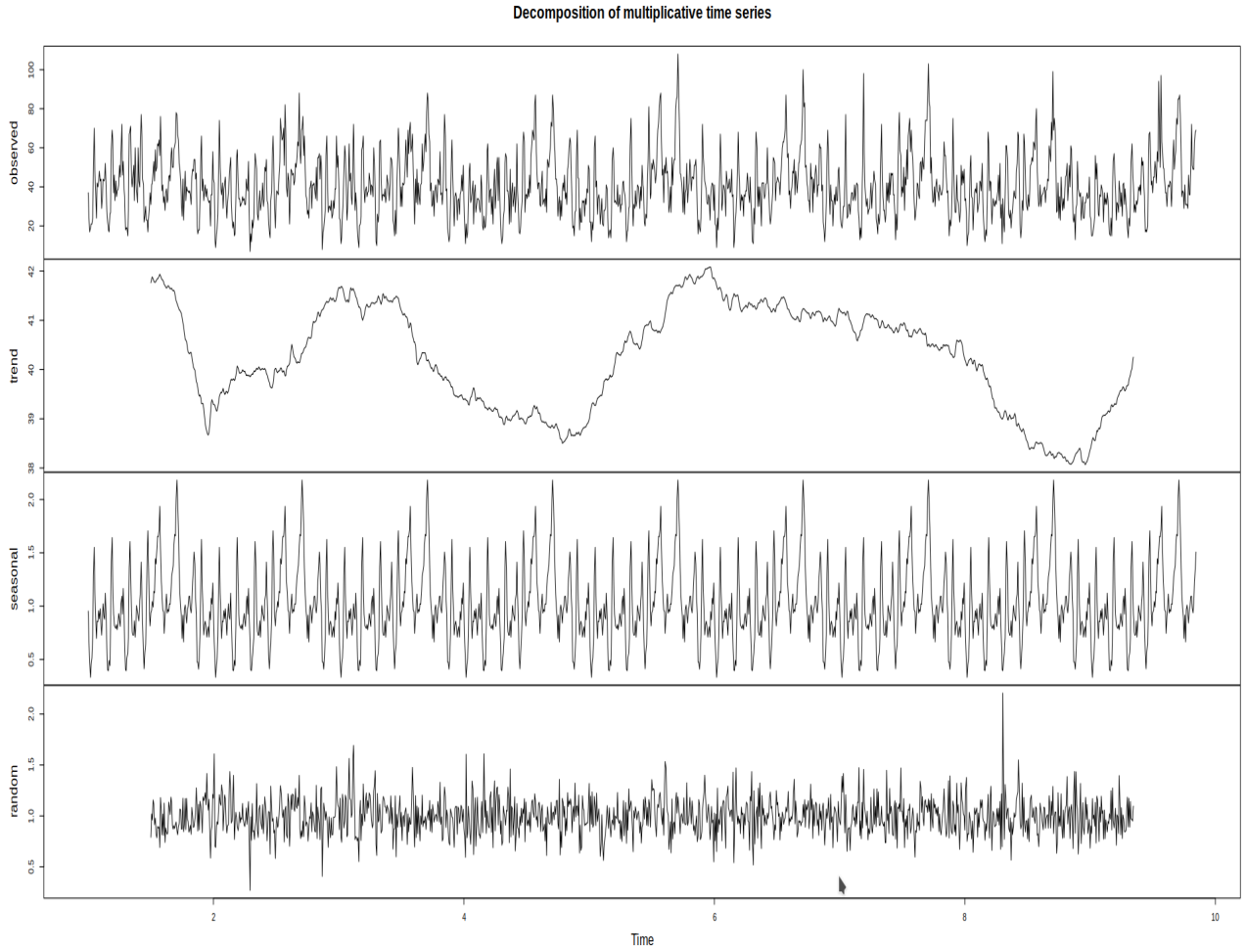

Figura 4: Serie decomposta

Dal grafico in figura 4 si nota che il modello moltiplicativo è pessimo per descrivere il fenomeno, infatti l'ordine di grandezza della stagionalità è il medesimo del rumore e nettamente inferiore a quelli del trend e del grafico osservato.
Non vale la pena di eseguire ulteriori test, per valutare la bontà di questo modello. Dunque nel seguito verrà considerato solamente il modello additivo.

5 Valutazione modello additivo

I test effettuati nella sezione seguente hanno la finalità di testare l'andamento casuale dei residui, riportati in figura 5, e dunque una loro distribuzione gaussiana.

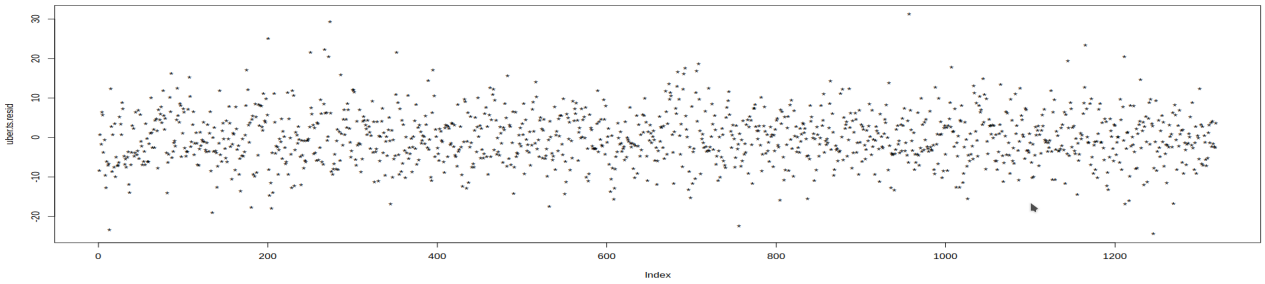

Figura 5: Residui

Inoltre possiamo visualizzare l'autocorrelazione tra i residui in figura 13.

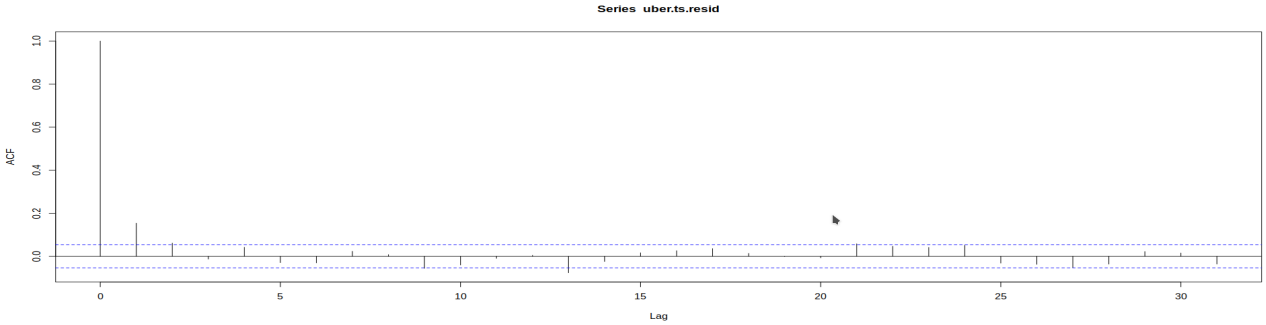

Figura 6: Funzione di autocorrelazione dei residui

Si deduce dal grafico che il modello additivo calcolato è un buon modello, perchè i valori sono tutti contenuti nell'intervallo di confidenza delimitato dalle rette tratteggiate blu, ma questa deduzione deve essere confermata effettuando alcuni test, riportati nella seguente sezione.

5.1 Test QQ_plot e visualizzazione distribuzione

Nel test QQ-Plot vengono confrontati i residui con una distribuzione gaussiana, i punti dovranno addensarsi lungo una retta, nel caso in cui siano privi di una struttura e duqnue siano dispoasti in maniera casuale.
Nel secondo test invece vengono confrontate la distribuzione dei residui, con una distribuzione normale teorica avente media e deviazione standard calcolate dai residui stessi, con la speranza che i due andamenti siano il più affini possibili.

```
layout(t(1:2));
uber.ts.resid = uber.ts.da$random;
qqnorm(uber.ts.resid);
qqline(uber.ts.resid, col="red");
hist(uber.ts.resid, 10, freq=F);
lines(density(uber.ts.resid), col="red");
lines(sort(uber.ts.resid), dnorm(sort(uber.ts.resid), mean(uber.ts.resid), sd(uber.ts.resid)));
```

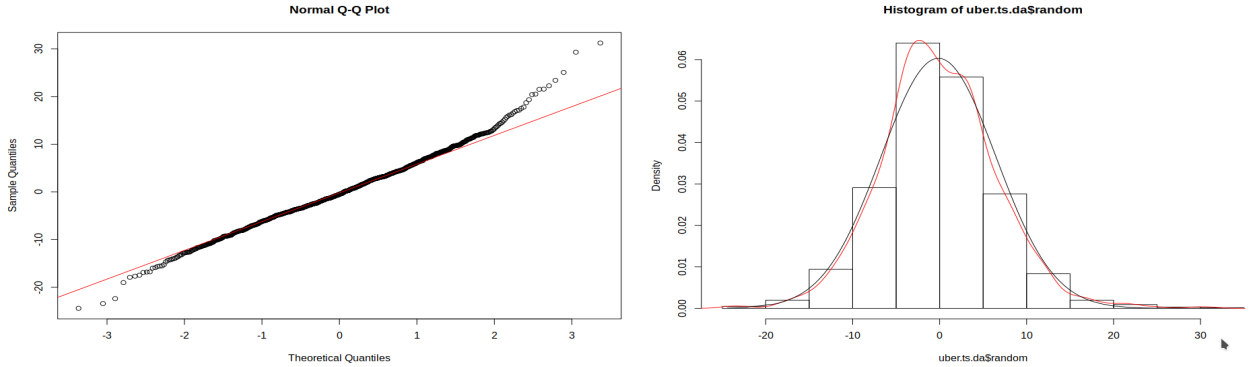

Figura 7: QQ-plot (sinistra) e distribuzione dei residui (destra)

Da entrambi i test otteniamo ottimi risultati, per cui concludiamo che questo è un buon modello e partendo da questo possiamo effettuare alcune predizioni.

6 Predizione

Lo step finale dell'analisi è calcolare un modello per eseguire una previsione, che sia il più accurata possibile. Il metodo utilizzato è un metodo di smorzamento esponenziale: HoltWinters.

```
uber.hw = HoltWinters(uber.ts);
plot(uber.hw);
```

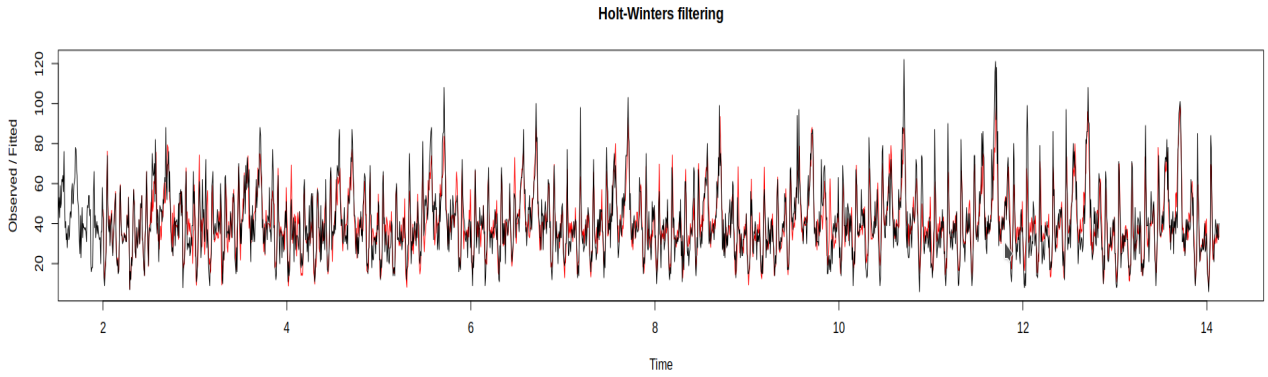

Figura 8: Metodo Holt-Winters

Da una prima valutazione grafica si può dedurre che il modello è abbastanza buono, in quanto sia presente una sovrapposizione abbastanza netta dei due andamenti.
Previsioni di una e due settimane:

```
layout(1:2)
plot(uber.hw, predict(uber.hw, 168))
plot(uber.hw, predict(uber.hw, 336))
```

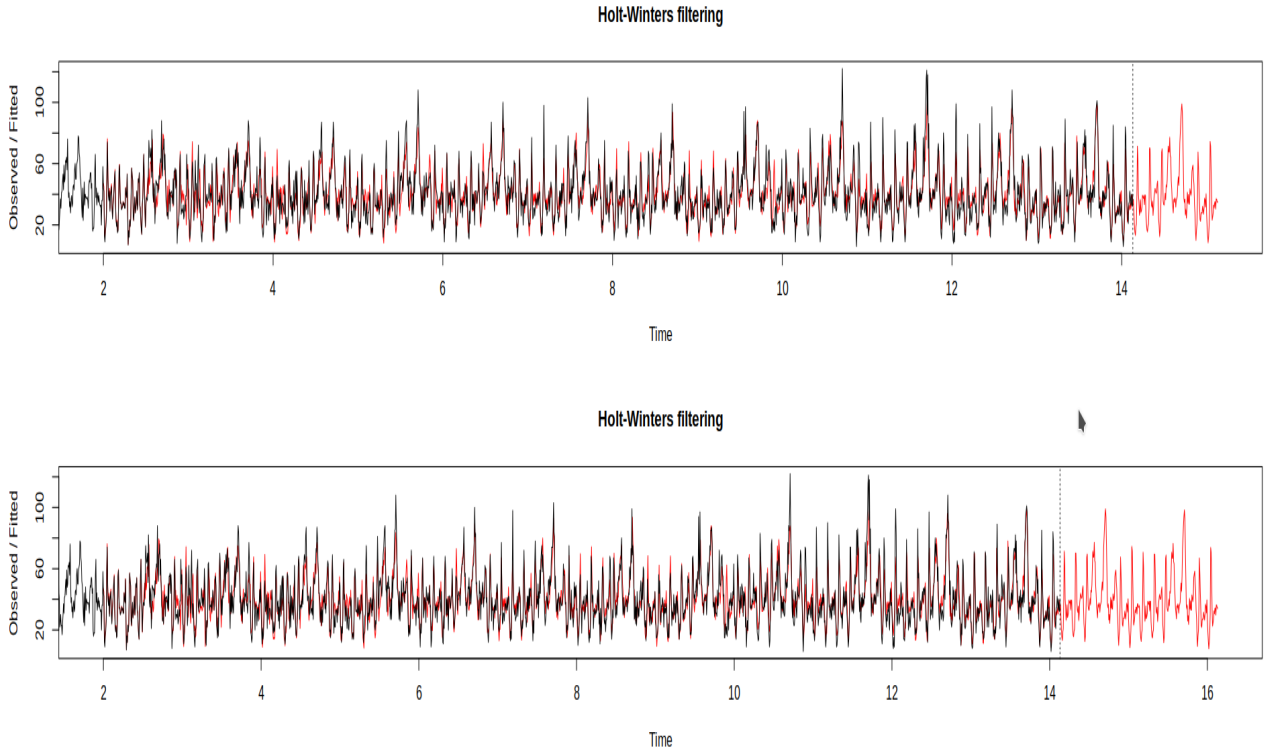

Figura 9: Predizione di 1 settimana (sopra) predizione di 2 settimane (sotto)

Dal grafico si può fare una valutazione euristica positiva, in entrambe le predizioni sono presenti sia i picchi giornalieri che quelli con una prominenza più ampia, relativi al fine settimana.

6.1 Analisi residui e varianza spiegata

Anche in questa sezione vogliamo verificare un andamento random dei residui, con le stesse modalità della sezione 5.

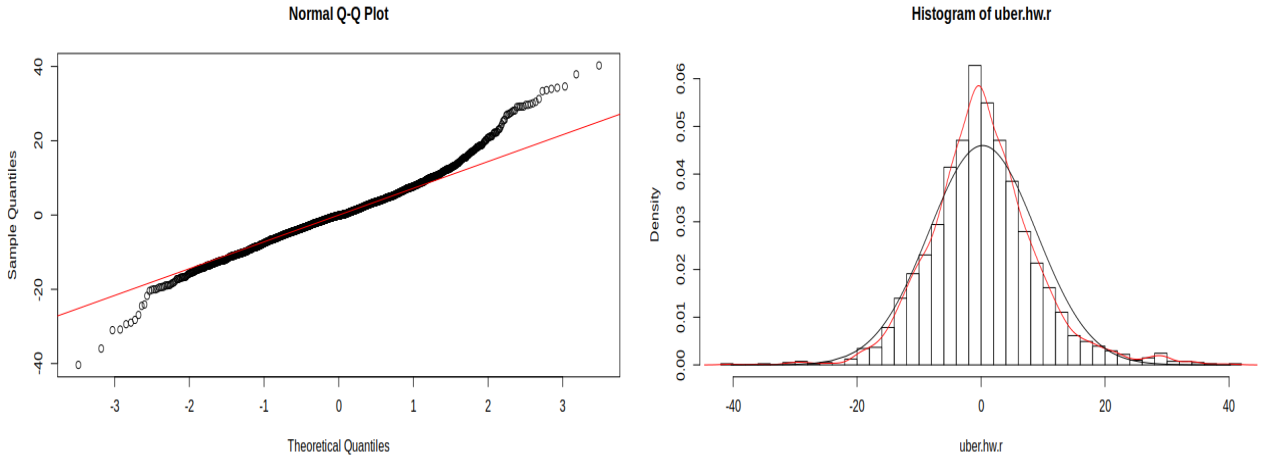

Figura 10: QQ-plot (sinistra) e distribuzione dei residui (destra)

I due test ci restituiscono risultati soddisfacenti, dunque possiamo considerare i residui come rumore aleatorio. Calcolando il valore della varianza spiegata si trova un buon valore: 79%, ad indicare che viene descritto piuttosto bene il fenomeno.

```
> 1 - var(uber.hw.r)/var(window(uber.ts, start= c(13,1), end=c(13,168)))
[1] 0.7901728
```

Figura 11: Formula e valore della varianza spiegata

6.2 Autovalutazione

E' opportuno eseguire una valutazione numerica della predizione, dunque effettuando un'autovalutazione del modello, si può calcolare l'errore relativo della predizione.
Per fare questo, splittiamo i dati in training e test set, si calcola il modello predittivo usando il training set, effettuiamo una previsione e si confrontano i valori predetti (grafico rosso) con quelli effettivi (grafico nero).

```
train = window(uber.ts, end=c(12,168));
test = window(uber.ts, start=c(13,1), end=c(13,168));
train.hw = HoltWinters(train);
test.pred = predict(train.hw, 168);
ts.plot(test, test.pred, col=c("black", "red"));
```

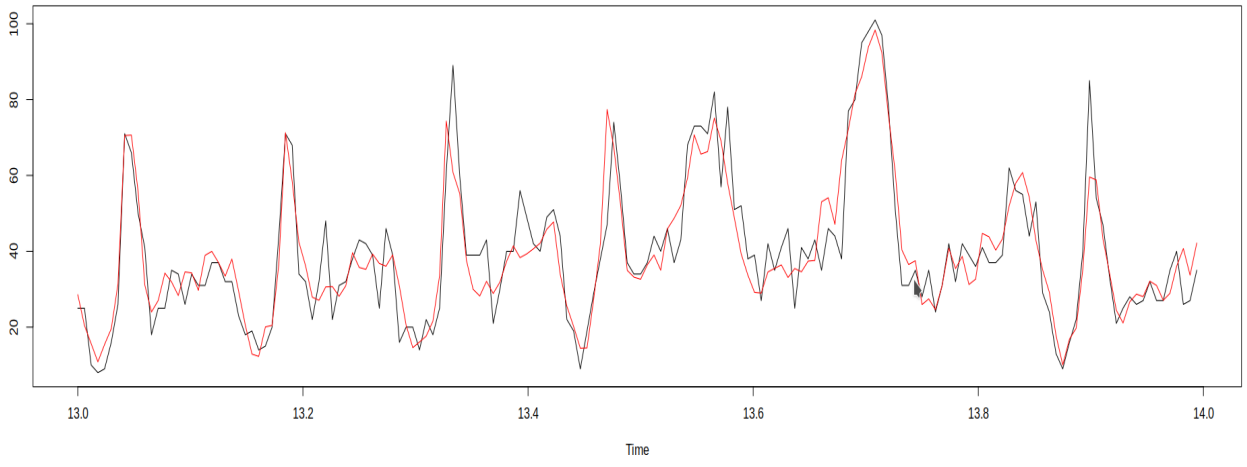

Figura 12: Andamento effettivo (nero) grafico predetto (rosso)

Infine si calcola l'errore relativo nella predizione.

```
> mean((test - test.pred)^2)/var(test)
[1] 0.1685695
```

Figura 13: Errore relativo

Il risultato ottenuto è molto buono, infatti abbiamo una previsione errata al 16% nel test appena svolto. Questo risultato è molto buono perchè è alto, ma non eccessivo, altrimenti ci troveremmo in un caso di over fitting, in cui il modello calcolato predice così bene, da essere in grado di riprodurre anche il rumore, che invece è una componente random.