ARIMA

Contents

1 Time series

Le serie temporali vengono create con la funzione `ts(data, start, end, frequency)`, dove:

- `data` è un vettore di dati equispaziati nel tempo
- `start` è la data della prima osservazione
- `end` è la data dell'ultima osservazione
- `frequency` è il numero di osservaizoni per unità temporale

Il significato dell'unità tempo base è arbitrario: se ad esempio indichiamo `start=2019` e `frequency=12` significa che i dati partono dal 2019 e hanno cadenza mensile. È possibile indicare `start=c(2019,6)` per stabilire che il primo dato è di Giugno 2019. **NOTA**: `start` deve essere o uno scalare o un vettore di due elementi, nel cui caso il secondo elemento è l'indice (base 1) del sottoperiodo quando `frequency` è maggiore di 1.

Le opzioni `end` o `deltat` possono essere indicate quando si vuole troncare il vettore di ingresso.

Come dati di esempio, carichiamo i dati della pandemia COVID-19 da Our World in Data:

```
datafile <- "owid-covid-data.csv"
url <- "https://covid.ourworldindata.org/data/owid-covid-data.csv"
if (!file.exists(datafile)) {
  print("Downloading data from the Internet")
  download.file(url, datafile)
}
covid <- read.csv(datafile)
```

Dell'intero set di dati filtriamo e selezioniamo solo i nuovi casi per milione in Italia, cotruendo poi un oggetto time series. Usiamo la libreria `lubridate` per semplificare la gestione delle date:

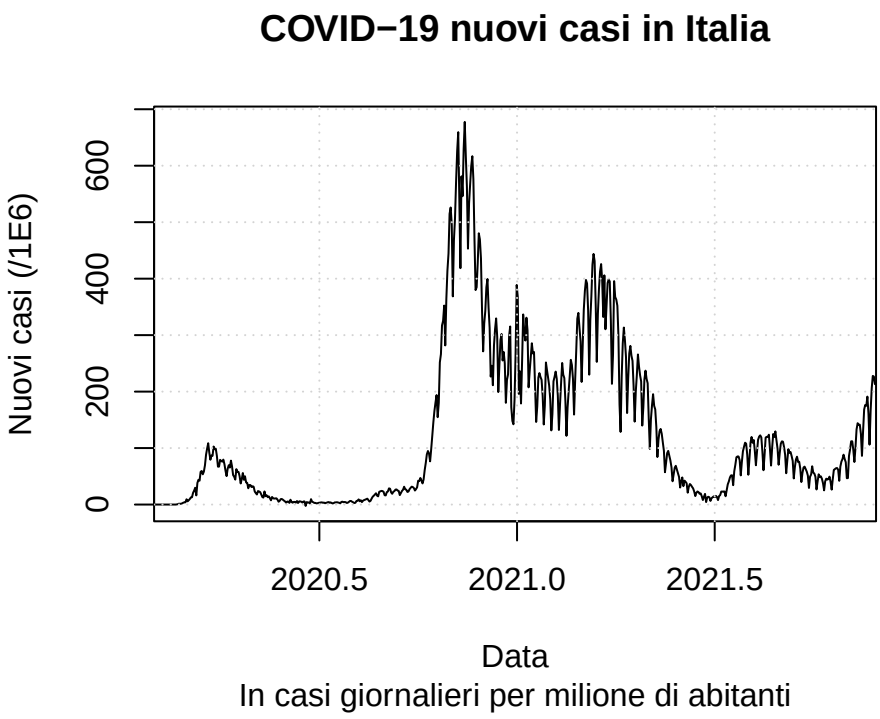
```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
st <- decimal_date(ymd(covid[covid$location=="Italy",]$date[1]))
cpm <- ts(
  covid[covid$location=="Italy",]$new_cases_per_million,
  start=st,
  frequency=365.25
```

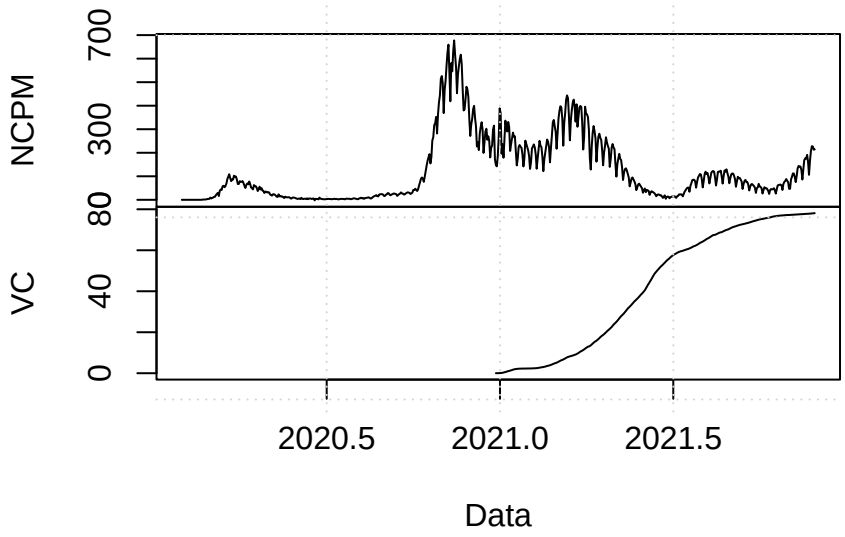
```
)
plot(cpm,
     main="COVID-19 nuovi casi in Italia",
     sub="In casi giornalieri per milione di abitanti",
     xlab="Data",
     ylab="Nuovi casi (/1E6)",
     xaxs="i"
     )
grid()
```



Si noti che l'espressione `decimal_date(ymd(covid$date[1]))` converte la data 2020-02-24 (una stringa) in un oggetto tempo 2020-02-24 e infine in un valore decimale a base annuale: 2020.147541 (*data astrale*).

È possibile creare oggetti timeserie multivariati, passando all'argomento `data` una matrice con più colonne:

```
cpmv <- ts(
  data=cbind(
    covid[covid$location=="Italy",]$new_cases_per_million,
    covid[covid$location=="Italy",]$people_vaccinated_per_hundred
  ),
  names=c("NCPM", "VC"),
  start=st,
  frequency=365.25
)
plot(cpmv,
     main="COVID-19 nuovi casi in Italia",
     sub="In casi giornalieri per milione di abitanti",
     xlab="Data",
     ylab="Nuovi casi (/1E6)",
     )
grid()
```



Funzioni utili per manipolare le serie temporali sono `window()` e `time()`: la prima consente di estrarre una finestra temporale tra due date, la seconda consente di estrarre il vettore dei tempi. Inoltre, sono utili le funzioni di smoothing fornite dalla libreria `zoo`

```
c(start(cpm), end(cpm))
```

```
## [1] 2020.082 2021.908
```

```
date_decimal(c(start(cpm), end(cpm)))
```

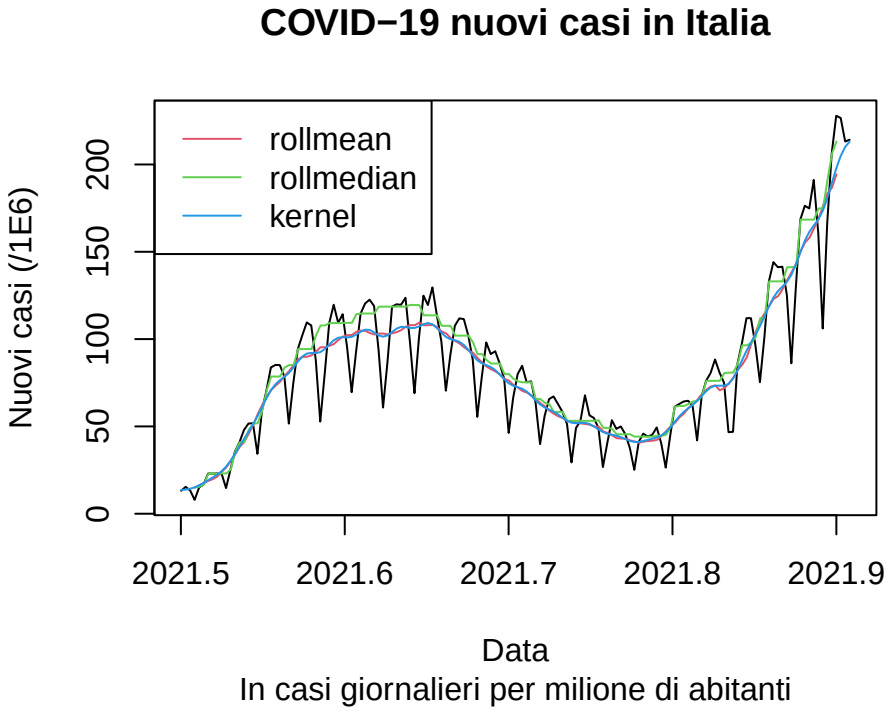
```
## [1] "2020-01-30 23:59:59 UTC" "2021-11-28 11:04:33 UTC"
```

```
win <- window(cpm, start=2021.5, end=end(cpm))
plot(win,
     main="COVID-19 nuovi casi in Italia",
     sub="In casi giornalieri per milione di abitanti",
     xlab="Data",
     ylab="Nuovi casi (/1E6)",
     xaxs="r"
     )
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

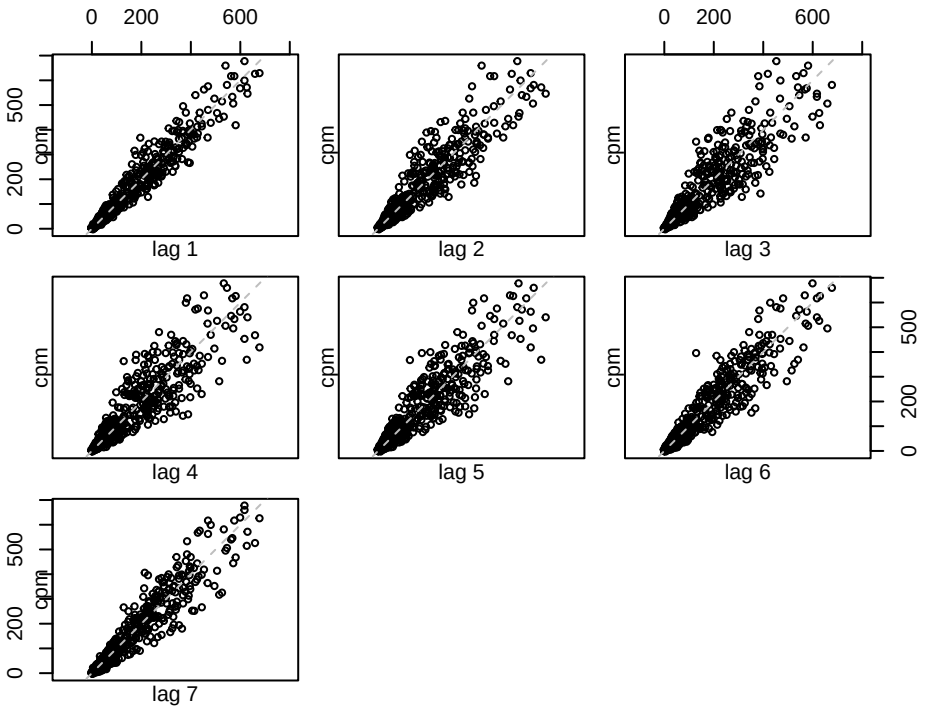
```
lines(rollmean(win, 7), typ="l", col=2)
lines(rollmedian(win, 7), typ="l", col=3)
lines(ksmooth(time(win), win, "normal", bandwidth=1/(365.25 / 7)), col=4)
legend("topleft", lty=1, col=2:4, legend=c("rollmean", "rollmedian", "kernel"))
```



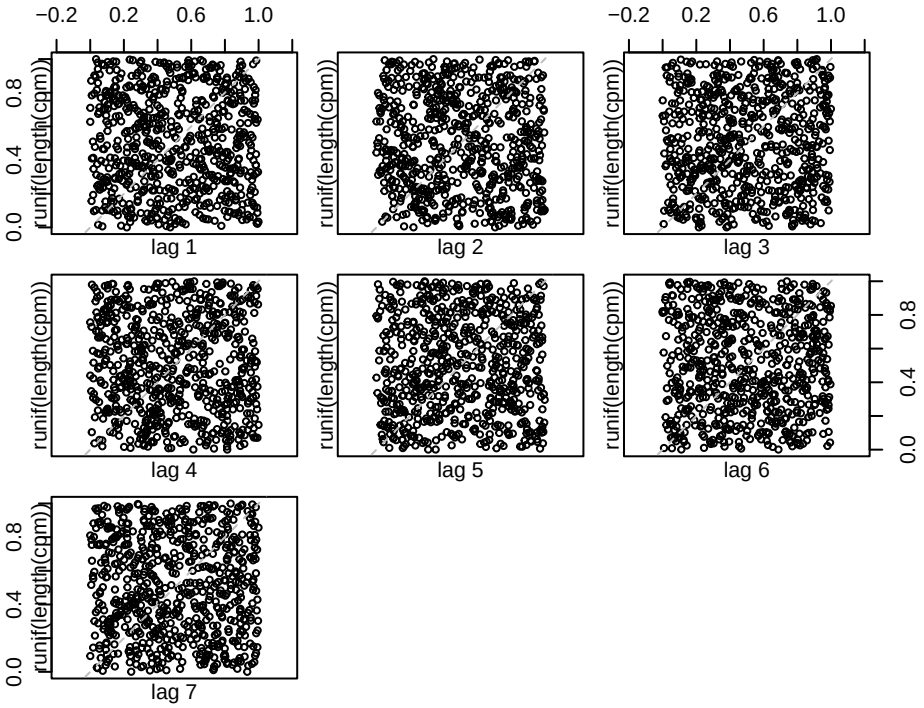
2 Regressione e Predizione

Prima di qualsiasi analisi su una serie temporale è utile visualizzare il cosiddetto **lag plot**, che è un particolare grafico a dispersione in cui si confrontano i dati di una serie con gli stessi dati con un certo ritardo: se il segnale è puramente casuale, il risultato sarà una nuvola dispersa; viceversa, ogni pattern significa che i dati sono affeti da un andamento regolare:

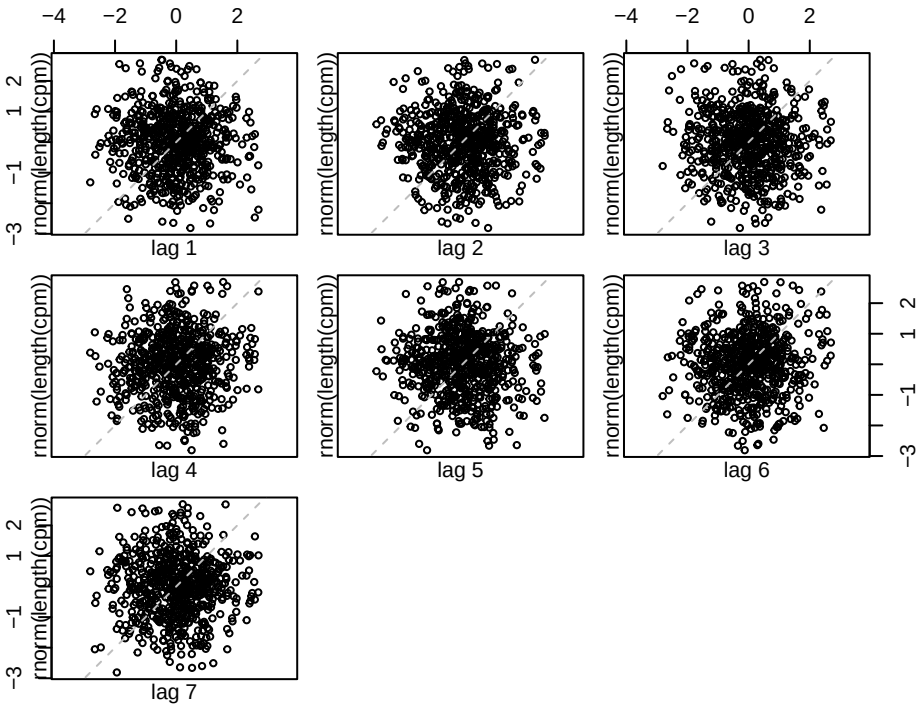
```
lag.plot(cpm, lags=7)
```



```
lag.plot(runif(length(cpm)), lags=7)
```




```
lag.plot(rnorm(length(cpm)), lags=7)
```



2.1 Auto-ARIMA

La libreria `forecast` mette a disposizione il metodo più semplice per effettuare la regressione di una serie temporale mediante ARIMA (*Auto-Regressive Integrative Moving Average*). Mettiamolo alla prova sulla serie temporale COVID-19, addestrando il modello fino alla data `2021.7=``rdate_decimal(2021.7)‘, utilizzando il

modello per predire i successivi 30 giorni, e poi confrontandolo con i dati reali.

```
library(forecast)
```

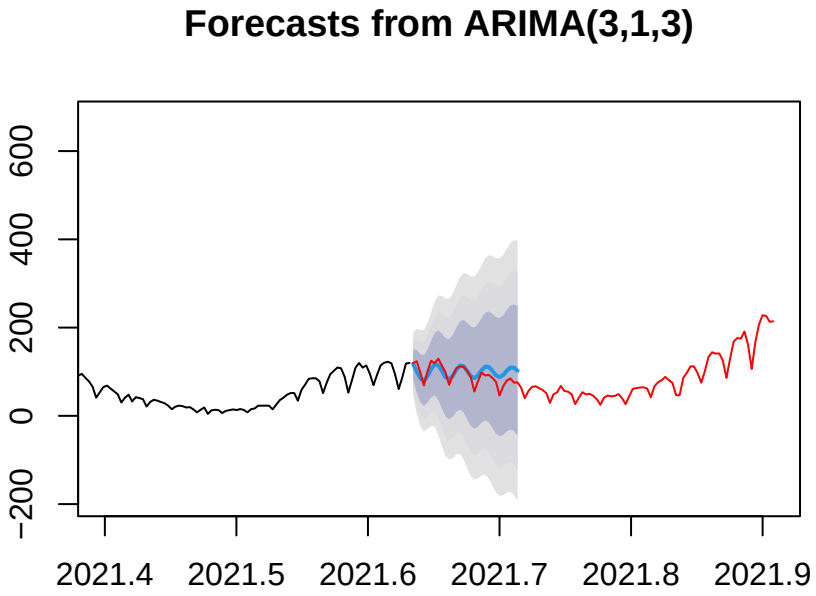
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
d0 <- decimal_date(ymd("2021-08-20"))
(fit <- auto.arima(window(cpm, start=st, end=d0)))
```

```
## Warning: The chosen seasonal unit root test encountered an error when testing for the first differen
## From stl(): series is not periodic or has less than two periods
## 0 seasonal differences will be used. Consider using a different unit root test.
```

```
## Series: window(cpm, start = st, end = d0)
## ARIMA(3,1,3)
##
## Coefficients:
##          ar1     ar2      ar3      ma1      ma2     ma3
##       0.3336  0.1430  -0.8509  -0.3474  -0.4194  0.8007
## s.e.  0.0379  0.0456   0.0456   0.0429   0.0810  0.0532
##
## sigma^2 estimated as 722.8:  log likelihood=-2664.38
## AIC=5342.75   AICc=5342.95   BIC=5373.12
```

```
plot(forecast(fit, 30, level=c(80, 95, 99)), # 30 giorni
     xlim=c(2021.4, end(cpm)))
lines(window(cpm, start=d0), col="red")
```



Vediamo le predizioni odierne:

```
(fit <- auto.arima(cpm))
```

```
## Warning: The chosen seasonal unit root test encountered an error when testing for the first differen
## From stl(): series is not periodic or has less than two periods
## 0 seasonal differences will be used. Consider using a different unit root test.
```

```
## Series: cpm
## ARIMA(3,1,3)
```

```
##
## Coefficients:
##          ar1     ar2      ar3      ma1      ma2     ma3
##       0.3432  0.1318  -0.8429  -0.3603  -0.4080  0.7962
## s.e.  0.0365  0.0438   0.0419   0.0385   0.0682  0.0451
##
## sigma^2 estimated as 645:  log likelihood=-3102.14
## AIC=6218.28   AICc=6218.45   BIC=6249.8
```

```
plot(forecast(fit, 30, level=c(80, 95, 99)),
     xlim=c(end(cpm)-4/54, end(cpm)+10/54)
)
abline(v=end(cpm))
```