An introduction to GARMA models

Richard Hunt

2020-10-22

Introduction to long memory models and GARMA models.

GARMA models are a type of time series models with special properties. These models are related to the "arima" models which can be fit in R by the standard "arima" function or the "Arima" function in the **forecast** package.

If you are not very familiar with ARIMA models, it might be advisable to first have a look at the excellent online book by Rob Hyndman and George Athanasopoulos:- Forecasting: Principals and Practice.

GARMA models are a type of **long memory** model, also known as **fractal** models or **fractionally differenced** models. The reason for this is that they can model proceses which have a high correlation between observations which are far apart in time.

For instance, consider the data on the minimum level of the Nile (Tousson 1925), as measured from 622 to 1281 AD (from the **longmemo** package):

```r
library(tidyverse)
library(forecast)

data(NileMin,package='longmemo')
# we'll just set the correct start year, for display purposes.
NileMin<-ts(as.numeric(NileMin),start=622,frequency=1)

ggtsdisplay(NileMin,lag.max=350, main='NileMin time series.', theme=theme_bw())
```

NileMin time series.
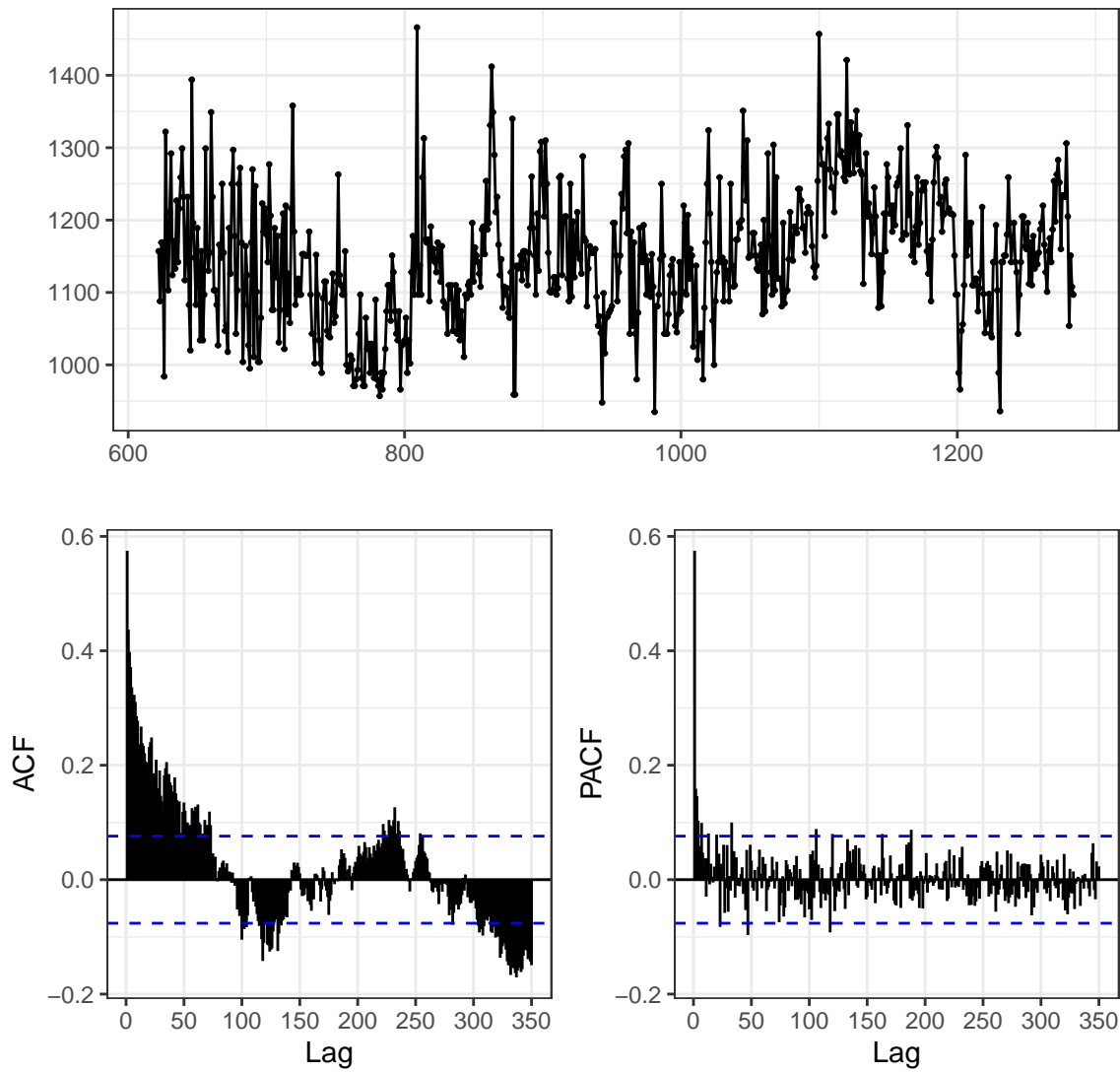
The first thing to note here is that the ACF (the autocorrelation function) is showing highly significant correlations between Nile measurements, even at 350 years apart!

We won't attempt to address the question of how it is that a flood or minima level of a river flow today could be significantly related to what was happening 350 years ago, but scientists believe this is a real affect, not an artifact of measurement. This time series has been investigated many times by researchers.

The phenomena of long memory has been identified in a number of different types of time series from sunspots (Gray, Zhang, and Woodward 1989), the timing of xrays from Cygnus X1 (Greenhough et al. 2002), inflation rates, exchange rates (Caporale and Gil-Alana 2014) and even in non-coding DNA (Lopes and Nunes 2006).

Generically, long memory models (sometimes known as 'fractionally differenced' models) can also have 'short memory' components, and a number of these have been explored. One popular type of short memory model is the ARMA model. There are a number of R packages which can fit these models, including **fracdiff**, **longmemo**, **rugarch** and even **forecast**, among others.

These models are considered useful because they produce long range forecasts which tend to be more accurate than other short-memory models, particularly if the model assumptions are correct.

At time of writing there are few R packages on CRAN which can fit GARMA models.

Package **tswge** provides a function "est.garma.wge" which can fit a GARMA model, but unfortunately can only do so via a grid search technique (searching a grid of values of the likelihood function), which tends to be slow and can result in rather inaccurate results. It also does not allow any MA terms to be fitted.

Package **waveslim** has functions "spp.mle" and "spp2.mle" which can identify the parameters of a white noise process only (ie a process without any short memory component), using a wavelet method.

Package **VGAM** to have a function called "garma" but this actually fits very different models - it fits short-memory models which can include count data. Package **gsarima** also fits these short memory count models.

The models fit by the **garma** package are models which are not just long memory models, but where the long memory phenomena *cycles* in strength. An example of this is the Southern Oscillation Index.

```r
soi <-
  read_fwf('soi.long.data',fwf_widths(c(5, rep(7,12)), c("year", 1:12)),col_types='nnnnnnnnnnnnn') %>%
  gather(mth_no,soi,-year) %>%
  mutate(mth = lubridate::make_date(year,as.numeric(mth_no),1)) %>%
  select(mth,soi) %>%
  arrange(mth)

soi_ts <- ts(soi$soi,start=c(1951,2),frequency=12)

ggtsdisplay(soi_ts, lag.max=400, main='Southern Oscillation Index', theme=theme_bw())
```
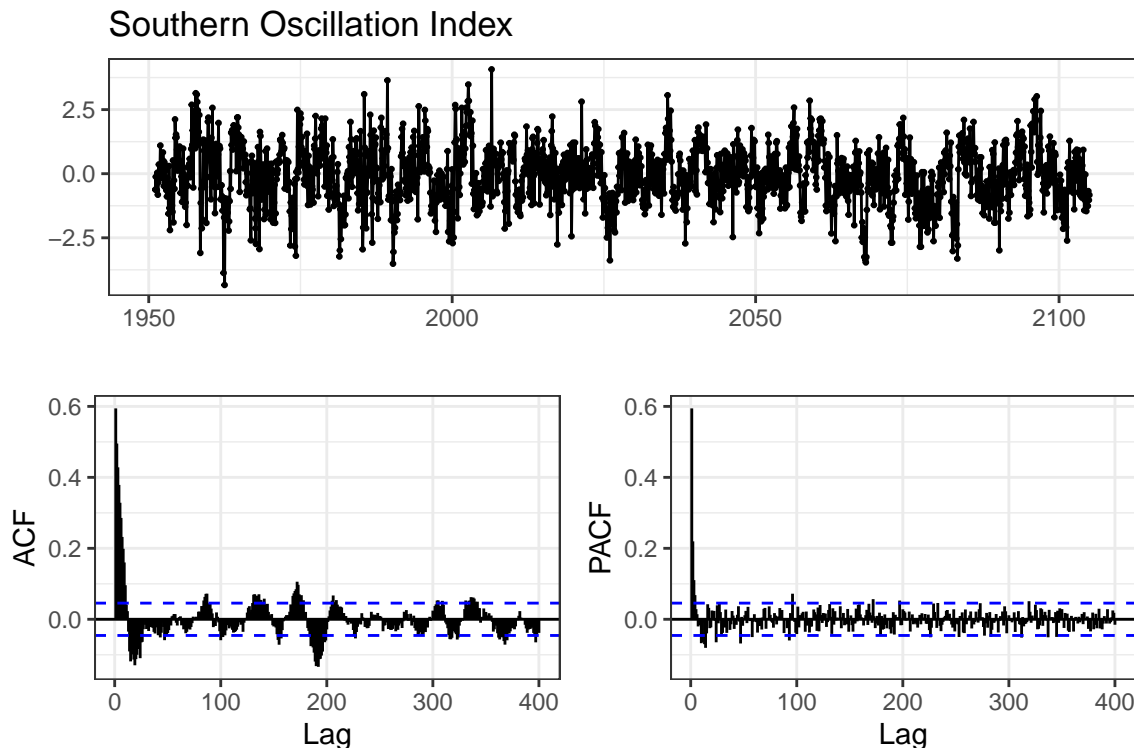

The first hing to note is that there are significant correlations in the ACF even up to lag 400, so this qualifies as a long memory process.

But also the ACF shows the cyclical nature which is very typical of Gegenbauer models.

When a long memory Gegenbauer model is combined with a short memory ARMA model, the result is known as a GARMA model.

GARMA stands for Gegenbauer AR-MA, and the AR and MA letters have their usual meaning in time series of "auto-regressive" and "moving average". 'Gegenbauer' refers to a mathematician Leopold Gegenbauer who lived in the 19th century and developed a series of orthogonal polynomials now called Gegenbauer polynomials, and sometimes called 'ultraspherical' polynomials (Szegò 1959). These polynomials feature strongly in the mathematics of GARMA models (Gray, Zhang, and Woodward 1989), (Gray, Zhang, and Woodward 1994).

All long memory models, including GARMA models, share one thing in common. When you examine them on the frequency domain, they will show evidence of an unbounded (read: very large) peak in the spectrum. The more traditional models - known as "short memory" models do not have such a peak - whilst the spectrum may vary up and down, a short memory model has a spectrum which has absolute bounds beyond which it will not go.

In the chart below we illustrate two long memory processes, one which would be well modelled by an ARFIMA model and the other which would be modelled by a Gegenbauer model.

```r
spectrum_nilemin <- spectrum(NileMin, plot=FALSE)
spectrum_soi     <- spectrum(soi_ts,  plot=FALSE)

# now munge these lists together into a single dataframe.
spec_df <- rbind(data.frame(freq=spectrum_nilemin$freq,
                            spec=spectrum_nilemin$spec,
                            process='NileMin'),
                 data.frame(freq=spectrum_soi$freq,
                            spec=spectrum_soi$spec,
                            process='SOI'))

# and plot
ggplot(spec_df, aes(x=freq,y=spec)) +
  geom_line() +
  facet_wrap(.~process,scales='free_y') +
  ggtitle('Spectrum of NileMin and SOI') +
  ylab('Intensity') +
  xlab(bquote('Frequency (0 -' ~ pi ~')' )) + xlim(0,pi) +
  theme_bw()
#> Warning: Removed 447 row(s) containing missing values (geom_path).
```

Spectrum of NileMin and SOI
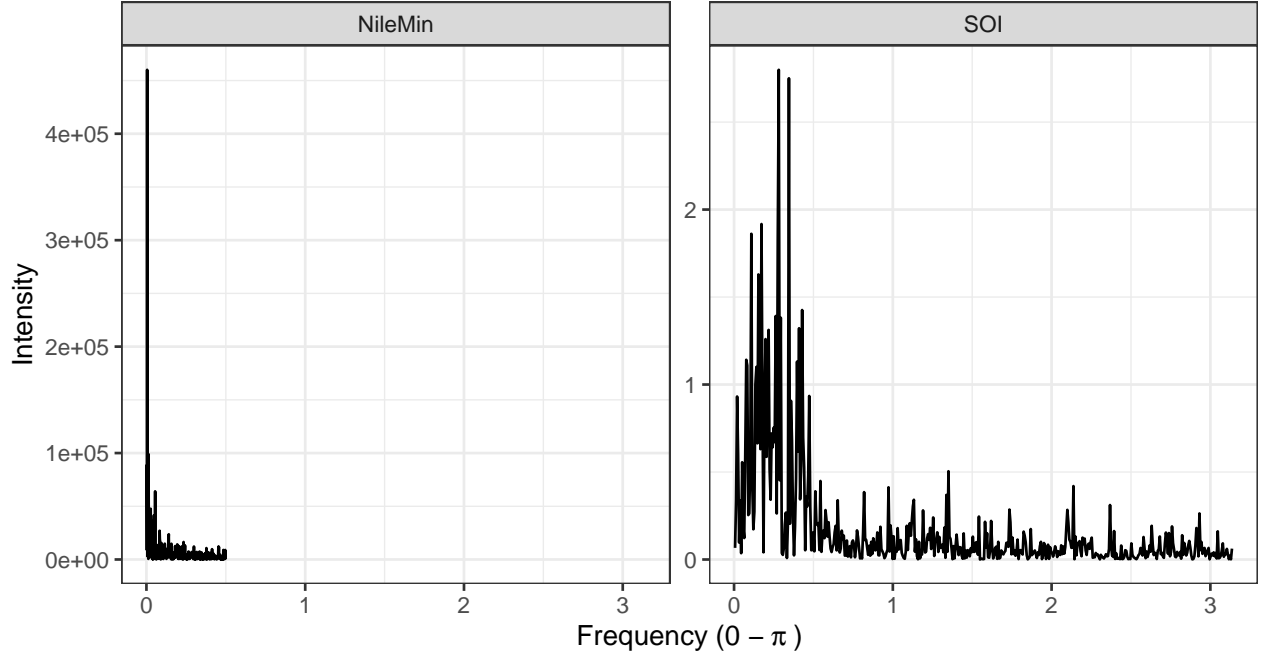
From the above you can see that the NileMin spectrum shows an (essentially) unbounded peak at 0 which is the marker of a traditional long memory process.

However the SOI has 3 main peaks, all separated away from 0. Although these are not as large as the NileMin peaks, compared with the rest of the spectrum they are quite large, so it is not inappropriate to at least try to model the SOI by a 3-factor (or $k = 3$) Gegenbauer model. This was first pointed out by (Lustig, Charlot, and Marimoutou 2017).

Technical details of the model.

The GARMA model as fit by the **garma** package is specified as

$$\phi(B) \prod_{i=1}^{k} (1 - 2u_i B + B^2)^{d_i} (1 - B)^{id} (X_t - \mu) = \theta(B)\epsilon_t$$

where

1. $\phi(B)$ represents the short-memory Autoregressive component of order p,
2. $\theta(B)$ represents the short-memory Moving Average component of order q,
3. $(1 - 2u_i B + B^2)^{d_i}$ represents the long-memory Gegenbauer component (there may in general be k of these),
4. id represents integer differencing (currently only id=0 or 1 is supported).
5. X_t represents the observed process,
6. ϵ_t represents the random component of the model - these are assumed to be uncorrelated but identically distributed variates. Generally the routines in this package will work best if these have an approximate Gaussian distribution.
7. B represents the Backshift operator, defined by $BX_t = X_{t-1}$.

When $k = 0$, then this is just a short memory model as fit by the stats "arima" function.

Fitting a short memory model

We have deliberately kept the fitting process close to that of the "arima" and "forecast::Arima" functions.

To illustrate basic usage of the routine, we will first look at fitting a simple ARIMA model to the "AirPassengers" data supplied with R. To achieve stationarity with this data, we'll need to seasonally difference it. The "arima" function can do this but unfortunately "garma" as yet does not fit a seasonal model - seasonality is essentially modelled by the Gegenbauer components, but we won't initially use that.

To ensure the results are as comparable as possible we will work with the seasonally differenced series, and specify the estimation method as 'CSS' for both.

```
library(garma)
#>
#>
#> Package garma: 0.9.6
data(AirPassengers)

ap  <- as.numeric(diff(AirPassengers,12))

# Arima model
arima_mdl <- arima(ap,order=c(9,1,0),method='CSS')
summary(arima_mdl)
#>
#> Call:
#> arima(x = ap, order = c(9, 1, 0), method = "CSS")
#>
#> Coefficients:
#>          ar1      ar2      ar3      ar4      ar5     ar6      ar7      ar8
#>      -0.3255  -0.0136  -0.1326  -0.2340  -0.0154  0.0319  -0.1483  -0.0906
#> s.e.   0.0860   0.0919   0.0917   0.0924   0.0956  0.0952   0.0931   0.0940
#>          ar9
#>      0.2214
#> s.e.  0.0970
#>
#> sigma^2 estimated as 130.1:  part log likelihood = -504.76
#>
#> Training set error measures:
#>                    ME     RMSE      MAE  MPE MAPE      MASE       ACF1
#> Training set 0.2632469 10.96598 8.401014 -Inf  Inf 0.8918418 0.01807268

# GARMA model
# Note in the below we specify k=0.
# This tells the routine is not to fit a Gegenbauer/GARMA model.
garma_mdl <- garma(ap,order=c(9,1,0),k=0,method='CSS')
summary(garma_mdl)
#>
#> Call:
#> garma(x = ap, order = c(9, 1, 0), k = 0, method = "CSS")
#>
#> No Mean term was fitted.
#> Drift (trend) term was fitted.
#>
#> Coefficients:
#>          ar1       ar2       ar3       ar4        ar5      ar6       ar7
#>      -0.31827  -0.01366  -0.13574  -0.23170  -0.009474  0.03382  -0.14804
#> s.e.   0.08599   0.09175   0.09157   0.09266   0.096243  0.09587   0.09376
#>          ar8       ar9
#>      -0.08961   0.22243
```

```
#> s.e.   0.09465   0.09788
#>
#>
#>
#> sigma^2 estimated as 123.4608: part log likelihood = -501.323976
```

We note here that (Box et al. 2015) advise taking logarithms as the more correct way to model this data, however we are illustrating the similarity of methods here and feel additional code may obscure the intention. The interested reader may care to repeat this using logs themselves.

As can be seen above, the coefficients produced are similar but not identical - the log-likelihood from the "garma" run is a larger value than that produced by the "arima" run, indicating that the routine has in fact found a (slightly) better solution (estimating parameters like these involves non-linear optimisation - "arima" uses R's built-in optimiser called "optim"; garma however by default uses "solnp" from the **Rsolnp** package).

Fitting a GARMA model.

In this section we'll look at fitting a GARMA model to the Sunspots data, which is supplied with R. This data has been analysed many times in the literature; the first time with a GARMA model was by (Gray, Zhang, and Woodward 1989). Generally, authors have used a standard subset of this data from 1749 to 1924.

The Sunspot data consists of counts of sunspots as observed over a considerable period. In terms of ARIMA models, (Box et al. 2015) suggest an AR(9) model may be best.

First we fit an ARIMA model to this data:

```
library(garma)

data(sunspot.year)

# Next we subset the data to ensure we are using the years used in the literature.
sunspots <- ts(sunspot.year[49:224],start=1749,end=1924)

# Now as in Gray et al 1989 we fit a GARMA(1,0) model:
sunspots_arima_mdl <- arima(sunspots, order=c(9,0,0),method='CSS')

summary(sunspots_arima_mdl)
#>
#> Call:
#> arima(x = sunspots, order = c(9, 0, 0), method = "CSS")
#>
#> Coefficients:
#>          ar1      ar2      ar3     ar4      ar5     ar6      ar7      ar8
#>       1.2419  -0.4976  -0.1406  0.1943  -0.1767  0.0979  -0.1139  0.1435
#> s.e.  0.0753   0.1195   0.1245  0.1247   0.1245  0.1234   0.1266  0.1228
#>          ar9  intercept
#>       0.0570    45.1427
#> s.e.  0.0766     5.6748
#>
#> sigma^2 estimated as 213.5:  part log likelihood = -721.72
#>
#> Training set error measures:
#>                       ME     RMSE      MAE  MPE MAPE      MASE         ACF1
#> Training set -0.0002709438 14.23214 10.49252 -Inf  Inf 0.6405913 -0.001348032
```

Next we look at a GARMA model, but instead of 9 AR parameters, we will only fit 1.

7

```r
library(garma)

data(sunspot.year)

# Next we subset the data to ensure we are using the years used in the literature.
sunspots <- ts(sunspot.year[49:224],start=1749,end=1924)

# Now as in Gray et al 1989 we fit a GARMA(1,0) model:
sunspots_garma_mdl <- garma(sunspots, order=c(1,0,0),k=1,method='CSS')

summary(sunspots_garma_mdl)
#>
#> Call:
#> garma(x = sunspots, order = c(1, 0, 0), k = 1, method = "CSS")
#>
#> Mean term was fitted.
#> No Drift (trend) term was fitted.
#>
#> Coefficients:
#>       intercept         u1        fd1       ar1
#>          45.139   0.847533    0.42278    0.4937
#> s.e.      3.622   0.006664    0.06761    0.1056
#>
#>                      Factor1
#> Gegenbauer frequency:    0.0890
#> Gegenbauer Period:      11.2305
#> Gegenbauer Exponent:     0.4228
#>
#>
#> sigma^2 estimated as 228.0985: part log likelihood = -727.553615
```

For the GARMA model, we have specified **method='CSS'** to ensure we are using a method as close as possible to that used by (Gray, Zhang, and Woodward 1989). The "garma" function uses a frequency-domain method known as the "whittle" method by default, since this method not only produces very accurate results very quickly, but also has a lot of theoretical results available to support its use - for example (Giraitis, Hidalgo, and Robinson 2001).

The following table compares the values found by Gray et al and by the "garma" function:

Parameter	Gray et al	garma
intercept	44.78	45.1386899
u	0.85	0.8475333
d	0.42	0.4227757
ar1	0.49	0.4936761

As you can see, the results are quite close (Gray et al only published their results to 2 decimal places).

Notice also that the routine displays the Gegenbauer Period - in this case 11.2 years - which corresponds nicely with the "known" 11 year sunspot cycle.

Also shown is the degree of fractional differencing and the Fractional Dimension of the original series.

Whilst this model does fit the data quite well, is quite simple and does not produce very effective forecasts. A better model is the GARMA(8,0) model, which was also examined by (Gray, Zhang, and Woodward 1989),

however to compare with the ARIMA model, we can fit the same number of AR terms - 9 - and then forecast below, for the next sunspot cycle.

```r
library(yardstick)
#> For binary classification, the first factor level is assumed to be the event.
#> Use the argument `event_level = "second"` to alter this as needed.
#>
#> Attaching package: 'yardstick'
#> The following object is masked from 'package:forecast':
#>
#>     accuracy
#> The following object is masked from 'package:readr':
#>
#>     spec

(sunspots_garma_mdl <- garma(sunspots, order = c(9, 0, 0), k = 1))
#>
#> Call:
#> garma(x = sunspots, order = c(9, 0, 0), k = 1)
#>
#> Mean term was fitted.
#> No Drift (trend) term was fitted.
#>
#> Convergence Code: 0
#> Optimal Value found: 18996.57687257
#>
#> Coefficients:
#>         intercept          u1        fd1        ar1         ar2         ar3       ar4
#>             45.03   8.413e-01   0.13327    0.97884    -0.27454    -0.16856    0.1825
#> s.e.        14.27   1.971e-05   0.03432    0.05818     0.04528     0.01828    0.0195
#>              ar5         ar6        ar7        ar8        ar9
#>         -0.14300     0.03694    0.01263    0.04781     0.0966
#> s.e.     0.02095     0.01867    0.01828    0.01793     0.0141
#>
#>                      Factor1
#> Gegenbauer frequency:     0.0909
#> Gegenbauer Period:       11.0000
#> Gegenbauer Exponent:      0.1333
#>
#>
#> sigma^2 estimated as 1356.3524: log likelihood = 3023.399111, aic = -6020.798222


compare_df <- data.frame(yr=1925:1935,
                         Actuals=as.numeric(sunspot.year[225:235]),
                         ARIMA=forecast(sunspots_arima_mdl,h=11)$mean,
                         GARMA=forecast(sunspots_garma_mdl,h=11)$mean)


cat(sprintf('\n\nEvaluating Test set data from 1925 to 1936.\n\nARIMA RMSE: %.2f\nGARMA RMSE: %.2f\n',
            yardstick::rmse(compare_df,Actuals,ARIMA)$.estimate,
            yardstick::rmse(compare_df,Actuals,GARMA)$.estimate))
#>
#>
#> Evaluating Test set data from 1925 to 1936.
#>
```

```
#> ARIMA RMSE: 13.59
#> GARMA RMSE: 11.67
```

The RMSE for the ARIMA model is 13.59 and the RMSE for the GARMA model is 11.67 - which shows in terms of future predictions the GARMA model has a smaller error.

But visually there appears to be little difference between the models.

```
arima_df <- data.frame(yr=1925:1935, sunspots=forecast(sunspots_arima_mdl,h=11)$mean,grp='AR(11) foreca

garma_df <- data.frame(yr=1925:1935, sunspots=forecast(sunspots_garma_mdl,h=11)$mean,grp='GARMA(1),k=1

actuals_df <- data.frame(yr=1749:1935,sunspots=as.numeric(sunspot.year[49:235]),grp='Actuals')

df <- rbind(actuals_df,garma_df,arima_df)

ggplot(df,aes(x=yr,y=sunspots,color=grp)) + geom_line() +
  scale_color_manual(name='',values=c('Actuals'='darkgray','AR(11) forecasts'='darkgreen','GARMA(1),k=1
  xlab('') + ylab('Sunspot Counts') +
  geom_vline(xintercept=1924,color='red',linetype='dashed',size=0.3) +
  theme_bw() +
  ggtitle('Sunspot Forecasts: Comparing ARIMA(9,0,0) and GARMA(9,0),k=1')
```


Sunspot Forecasts: Comparing ARIMA(9,0,0) and GARMA(9,0),k=1
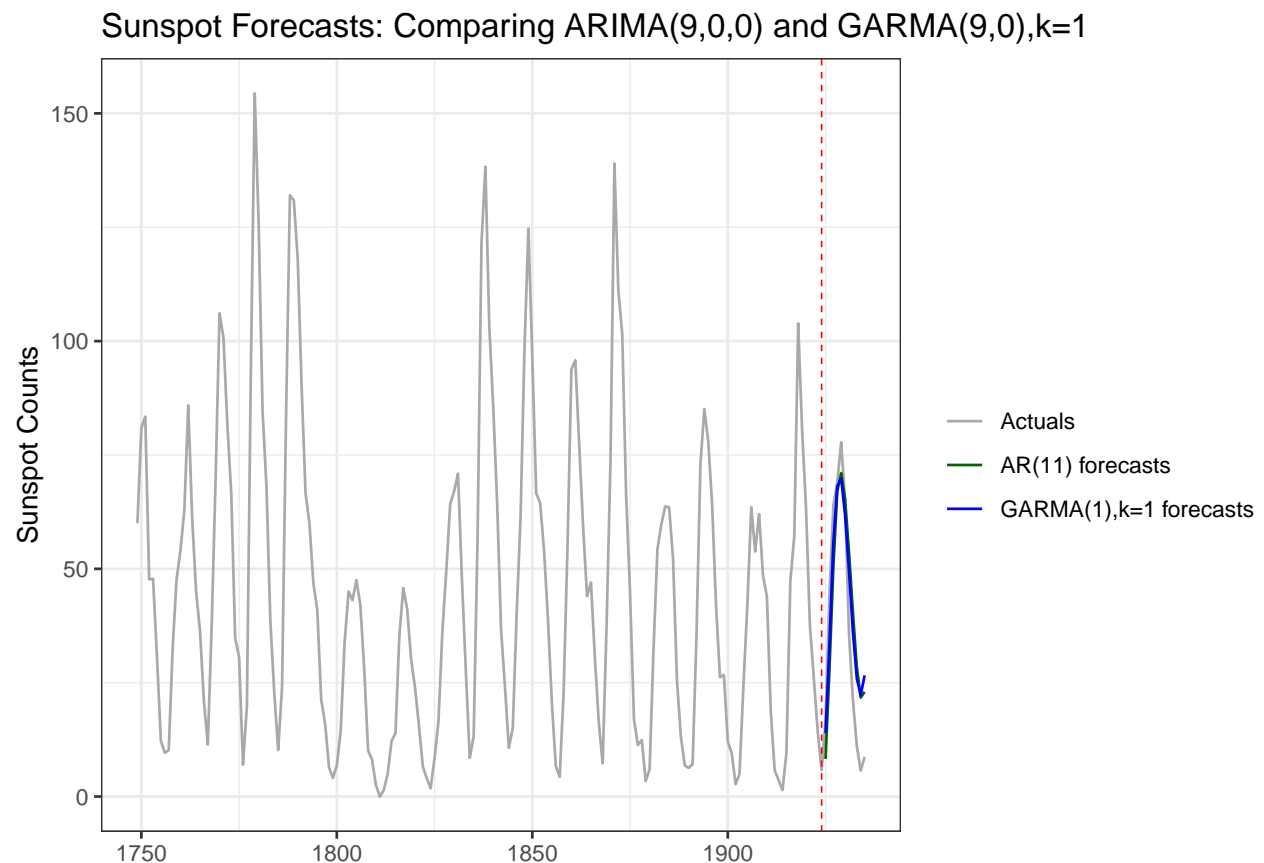
References

Box, George EP, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time Series Analysis : Forecasting and Control.* John Wiley & Sons Inc., Hoboken New Jersey.

Caporale, G, and L Gil-Alana. 2014. "Long-Run and Cyclical Dynamics in the Us Stock Market." *Journal of Forecasting* 33: 147–61.

Giraitis, L, J Hidalgo, and P Robinson. 2001. "Gaussian Estimation of Parametric Spectral Density with Unknown Pole." *The Annals of Statistics* 29 (4): 987–1023.

Gray, H, N Zhang, and W Woodward. 1989. "On Generalized Fractional Processes." *Journal of Time Series Analysis* 10 (3): 233–57.

———. 1994. "On Generalized Fractional Processes - a Correction." *Journal of Time Series Analysis* 15 (5): 561–62.

Greenhough, J, S Chapman, S Chaty, R Dendy, and G Rowlands. 2002. "Characterising Anomalous Transport in Accretion Disksfrom X-Ray Observations." *Astronomy and Astrophysics* 385: 693–700.

Lopes, S, and M Nunes. 2006. "Long Memory Analysis in Dna Sequences." *Physica A* 361: 569–88.

Lustig, A, P Charlot, and V Marimoutou. 2017. "The Memory of Enso Revisited by a 2-Factor Gegenbauer Process." *International Journal of Climatology* 37 (5): 2295–2303.

Szegò, G. 1959. *Orthogonal Polynomials.* AMS, New York.

Tousson, O. 1925. *Mémoire Sur L'Histoire Du Nil; Mémoire de L'Institut d'Egypte.*