University of Salford, MSc Data Science

Module: Applied Statistics and Data Visualisation

Date: Trimester 1, 2025-2026

Session: Workshop Week 10

Topic: Time Series Analysis in R

Tools: RStudio

Instructors: Dr Kaveh Kiani, Nathan Topping (Based on a book by Dr Avril Coghlan)

Objectives:

After completing this workshop, you will be able to:

- Work with the Family of SARIMA Models (AR, MA, ARMA, ARIMA and SARIMA)

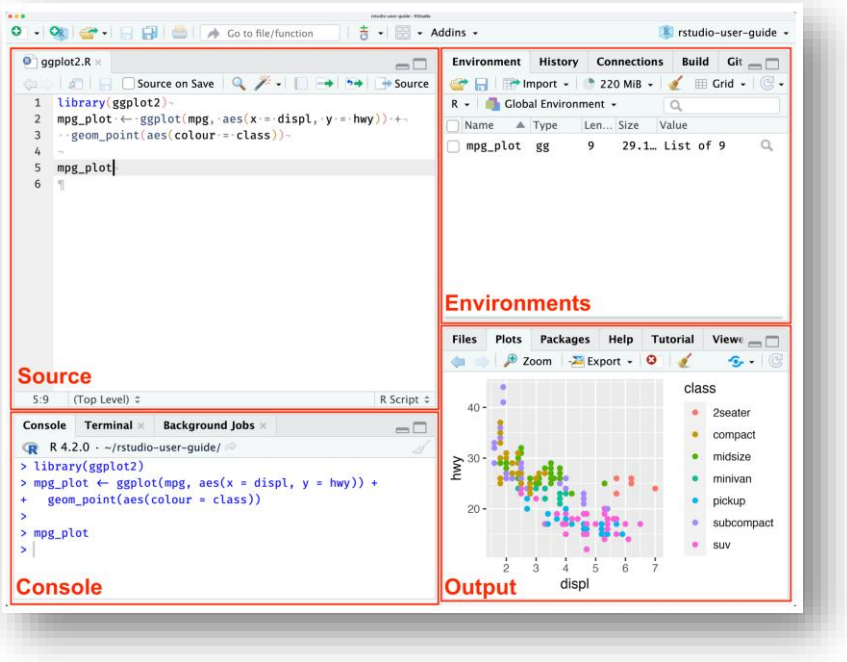
Contents

For your own benefit, we strongly encourage you to type out the R scripts yourself in RStudio, rather than copying and pasting them directly from the workshop notes. Typing the code helps reinforce key concepts, improves your understanding of the syntax, and strengthens your coding skills. While copying and pasting may seem faster, actively engaging with the code will lead to a deeper understanding and make you more proficient in R programming over time.

Take your time, experiment, and learn by doing!

Part 1: Working in RStudio

Take sometime later to read about the RStudio pane layout, but for now, just focus on knowing that there are **4 main panes**, **Source, Environments, Console and Output**
https://docs.posit.co/ide/user/ide/guide/ui/ui-panes.html

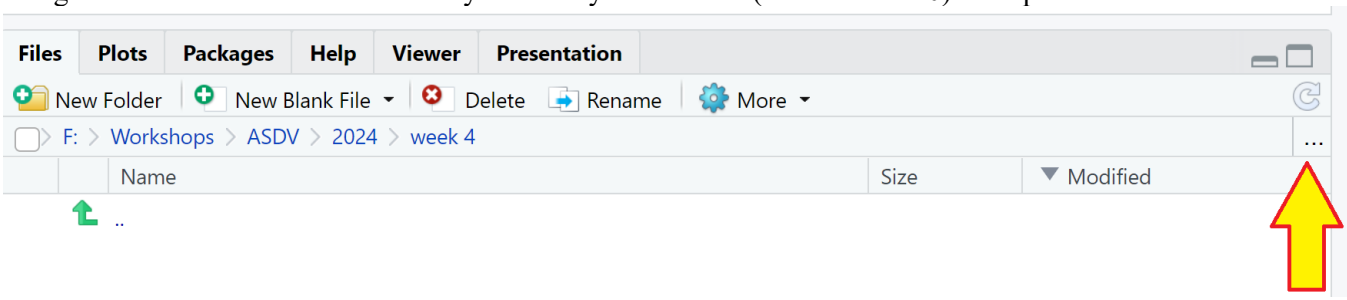


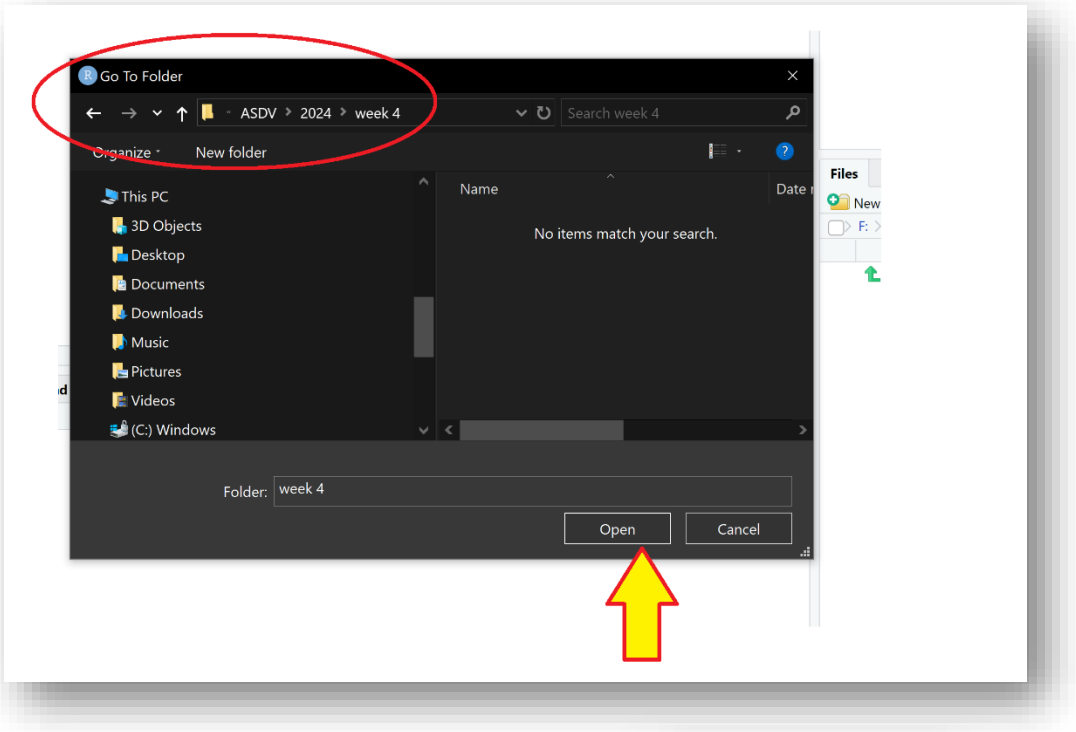
You may work through the workshops as follows:

1. Create a folder in the university PCs F drive or your personal OneDrive or your laptop to save datasets and R scripts. A good practice is to create a folder named **ASDV** (or similar) for all the R workshops that you will have throughout the trimester and then create a subfolder for each week. So, you can have **ASDV\Week10** for this week. Each week, you should **download the data for the week's workshop from Blackboard** and save it in this folder, as well as saving your scripts and the workshop materials in this folder too.

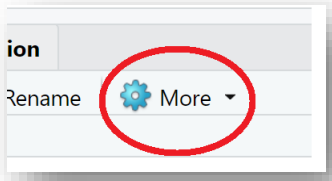
2. Start the RStudio (search in the Windows search box)

3. In the **Output Pane** (on the bottom right side of the RStudio window) click on the ellipse sign (…) to navigate to and select the folder where you saved your data file (ASDV\week10) and open the folder.

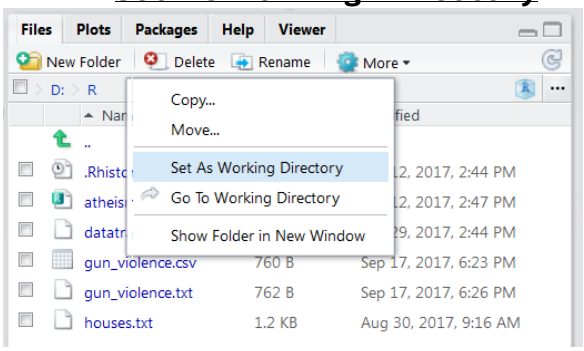




4. Then select the **More** tab



And then **`Set As Working Directory`**



5. From the **Source Pane** open a new R script window:

`File → New File → R script`

Part 2: Packages

Packages are collections of R functions, data, and compiled code in a well-defined format. A package is either a **Base Package** or a **User Contributed Package**.

There are a set of **Base (or Standard) Packages** which are considered part of the R source code and automatically available as part of the R installation and we can directly use functions from the standard base packages.

However, the more we work with R, we will come to realize that there are many users contributed packages that have been created to add specific functionality. There are 10,000+ **User Contributed Packages** and growing. To use **User Contributed Packages** will require installation (Many packages can be installed from the **CRAN** repositories).

We will be using three packages throughout the workshop today.

1. stats
2. TTR
3. forecast

TTR and forecast packages are **User Contributed Packages**, and we must install and call them before using. Package 1, "stats" is a **Base Package**, and it is ready to use.

```
install.packages("TTR")

install.packages("forecast")



library(TTR)

library(forecast)
```

Part 3: Source of the workshop:

The source of the workshop is second chapter of the:

A Little Book of R For Time Series
Release 0.2
Avril Coghlan
Sep 10, 2018

HTML version of the book:
https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/

PDF version of the book:
https://buildmedia.readthedocs.org/media/pdf/a-little-book-of-r-for-time-series/latest/a-little-book-of-r-for-time-series.pdf

License of the book:
https://creativecommons.org/licenses/by/3.0/

Updates of the workshop notes compare to the book:

The book	Workshop	Page
forecast.HoltWinters()	forecast()	30,38,…
plot.forecast()	plot()	30,38,…
acf(rainseriesforecasts2$residuals, lag.max=20)	acf(rainseriesforecasts2$residuals, lag.max=20 , na.action = na.pass)	31, …
---	Removing Null values	34,...
Forecast.Arima()	forecast()	60,…

Part 4: Chapter 2 - Using R for Time Series Analysis

ARIMA Models

2.1 ARIMA Models

Exponential smoothing methods are useful for making forecasts, and make no assumptions about the correlations between successive values of the time series. However, if you want to make prediction intervals for forecasts made using exponential smoothing methods, the prediction intervals require that the forecast errors are uncorrelated and are normally distributed with mean zero and constant variance.

While exponential smoothing methods do not make any assumptions about correlations between successive values of the time series, in some cases you can make a better predictive model by taking correlations in the data into account.

Autoregressive Integrated Moving Average (ARIMA) models include an explicit statistical model for the irregular component of a time series, that allows for non-zero autocorrelations in the irregular component.

2.1.1 Differencing a Time Series

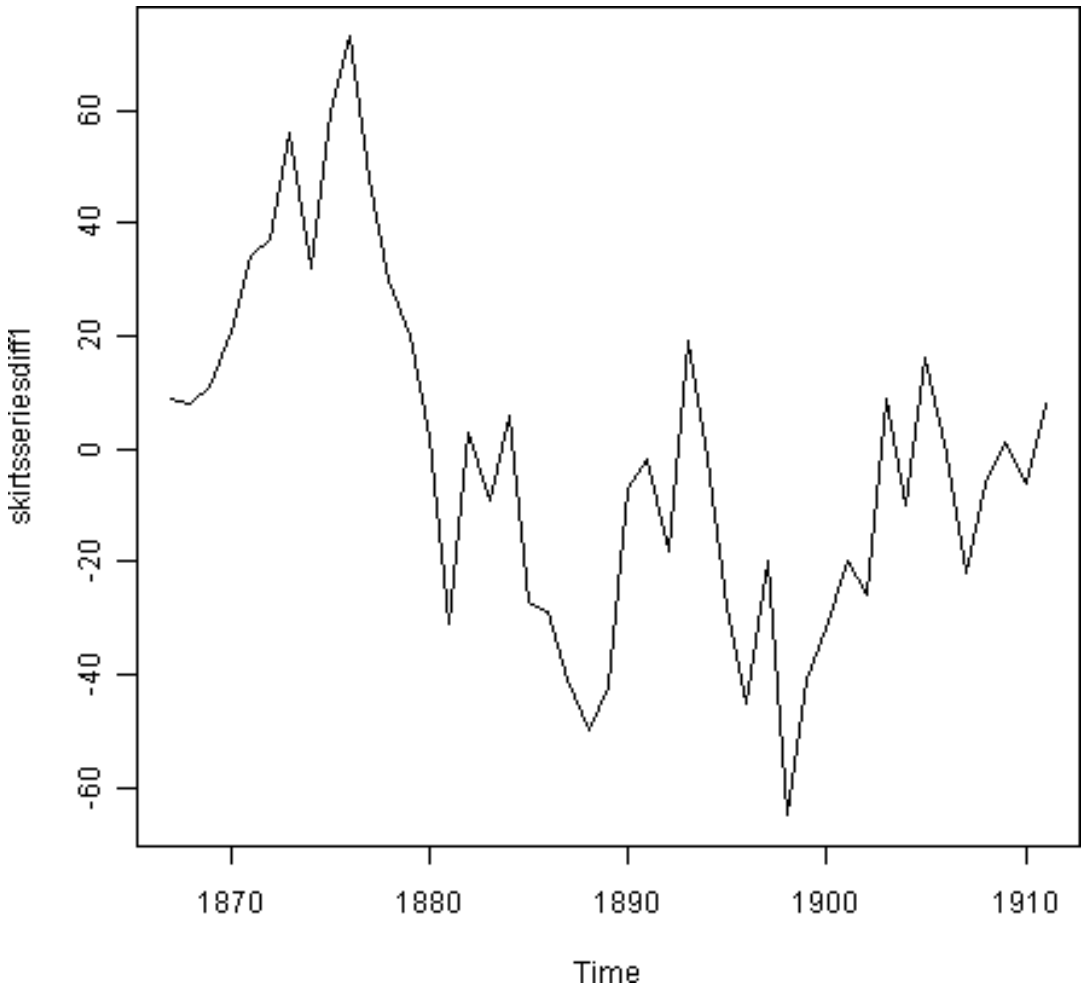
ARIMA models are defined for stationary time series. Therefore, if you start off with a non-stationary time series, you will first need to 'difference' the time series until you obtain a stationary time series.

If you have to difference the time series d times to obtain a stationary series, then you have an ARIMA(p,d,q) model, where d is the order of differencing used.

You can difference a time series using the "diff()" function in R. For example, the time series of the annual diameter of women's skirts at the hem, from 1866 to 1911 is not stationary in mean, as the level changes a lot over time:

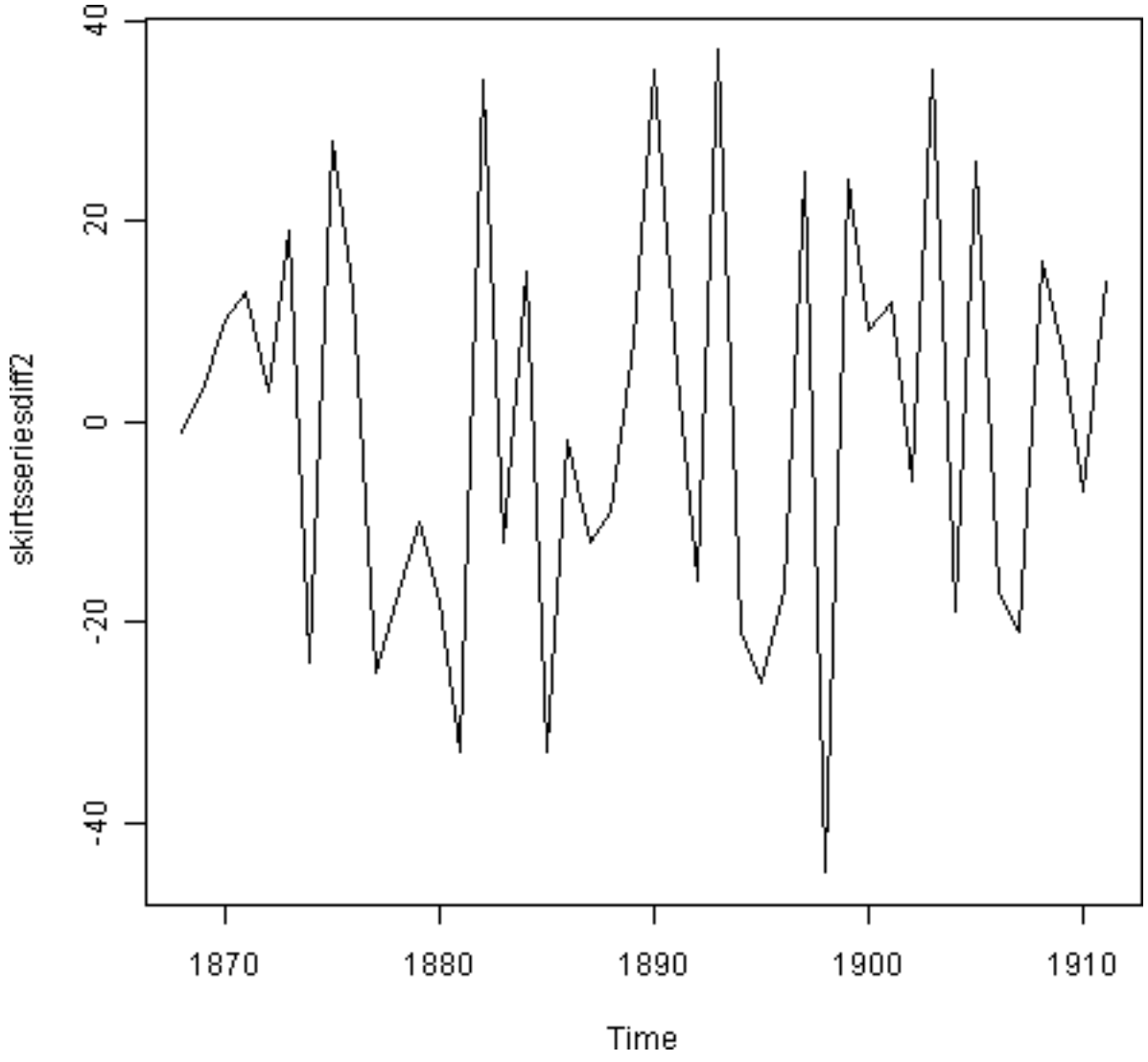
We can difference the time series (which we stored in "skirtsseries", see above) once, and plot the differenced series, by typing:

```
> skirtsseriesdiff1 <- diff(skirtsseries, differences=1)
> plot.ts(skirtsseriesdiff1)
```



The resulting time series of first differences (above) does not appear to be stationary in mean. Therefore, we can difference the time series twice, to see if that gives us a stationary time series:

```
> skirtsseriesdiff2 <- diff(skirtsseries, differences=2)
> plot.ts(skirtsseriesdiff2)
```



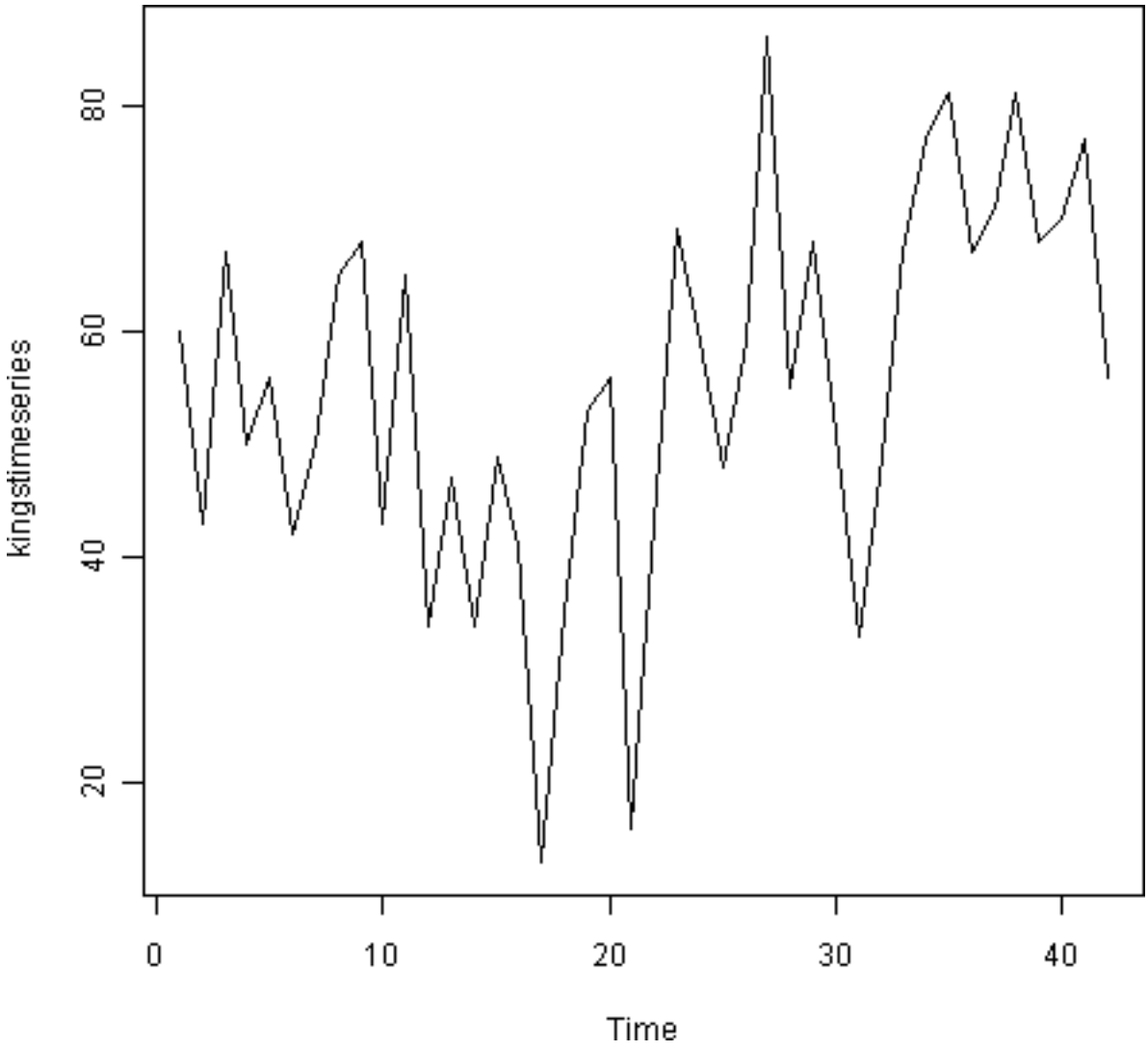
Formal tests for stationarity

Formal tests for stationarity called "unit root tests" are available in the fUnitRoots package, available on CRAN, but will not be discussed here.

The time series of second differences (above) does appear to be stationary in mean and variance, as the level of the series stays roughly constant over time, and the variance of the series appears roughly constant over time. Thus, it appears that we need to difference the time series of the diameter of skirts twice in order to achieve a stationary series.

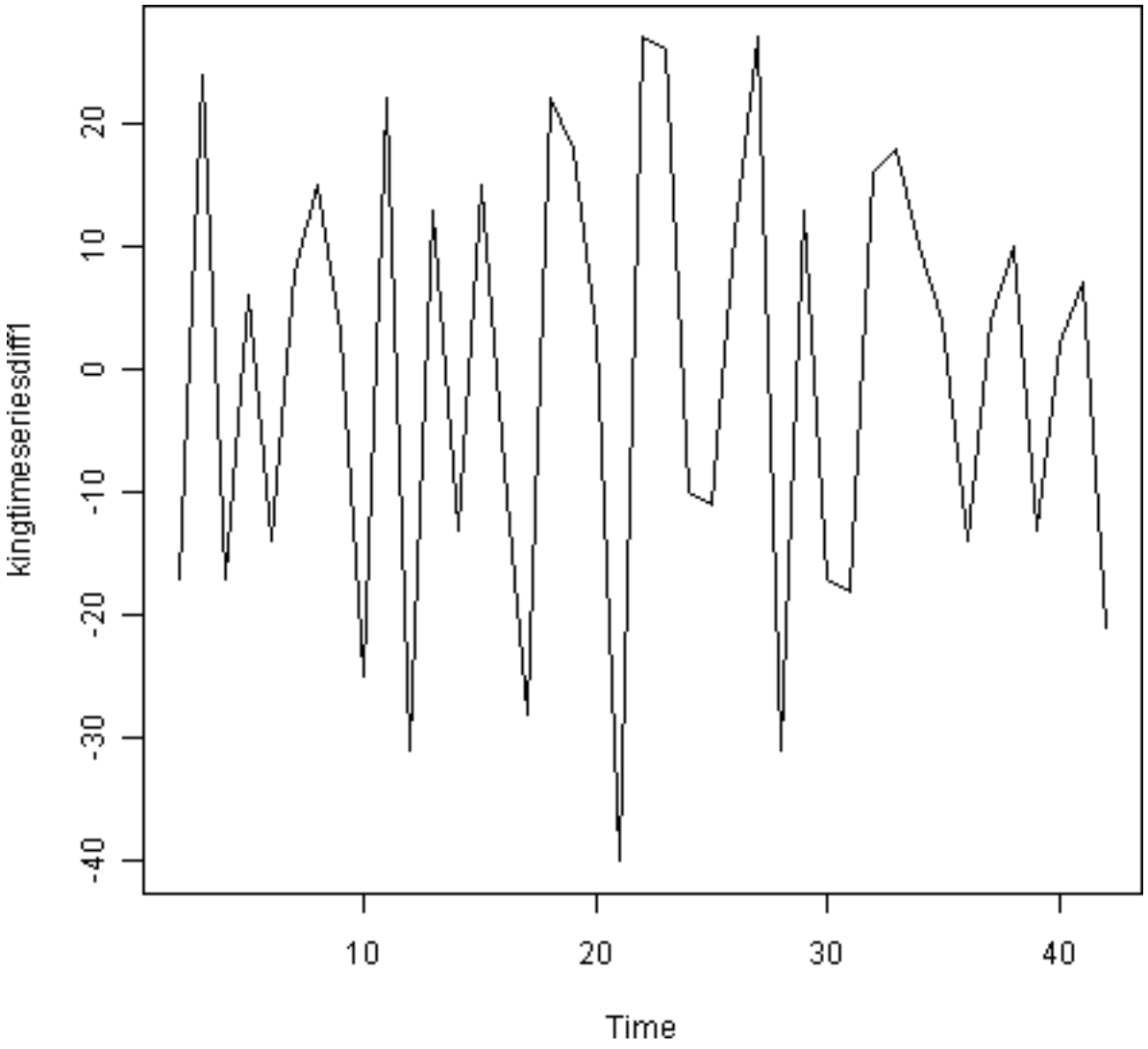
If you need to difference your original time series data d times in order to obtain a stationary time series, this means that you can use an ARIMA(p,d,q) model for your time series, where d is the order of differencing used. For example, for the time series of the diameter of women's skirts, we had to difference the time series twice, and so the order of differencing (d) is 2. This means that you can use an ARIMA(p,2,q) model for your time series. The next step is to figure out the values of p and q for the ARIMA model.

Another example is the time series of the age of death of the successive kings of England (see above):



From the time plot (above), we can see that the time series is not stationary in mean. To calculate the time series of first differences, and plot it, we type:

```
> kingtimeseriesdiff1 <- diff(kingstimeseries, differences=1)
> plot.ts(kingtimeseriesdiff1)
```



The time series of first differences appears to be stationary in mean and variance, and so an ARIMA(p,1,q) model is probably appropriate for the time series of the age of death of the kings of England. By taking the time series of first differences, we have removed the trend component of the time series of the ages at death of the kings, and are left with an irregular component. We can now examine whether there are correlations between successive terms of this irregular component; if so, this could help us to make a predictive model for the ages at death of the kings.

2.1.2 Selecting a Candidate ARIMA Model

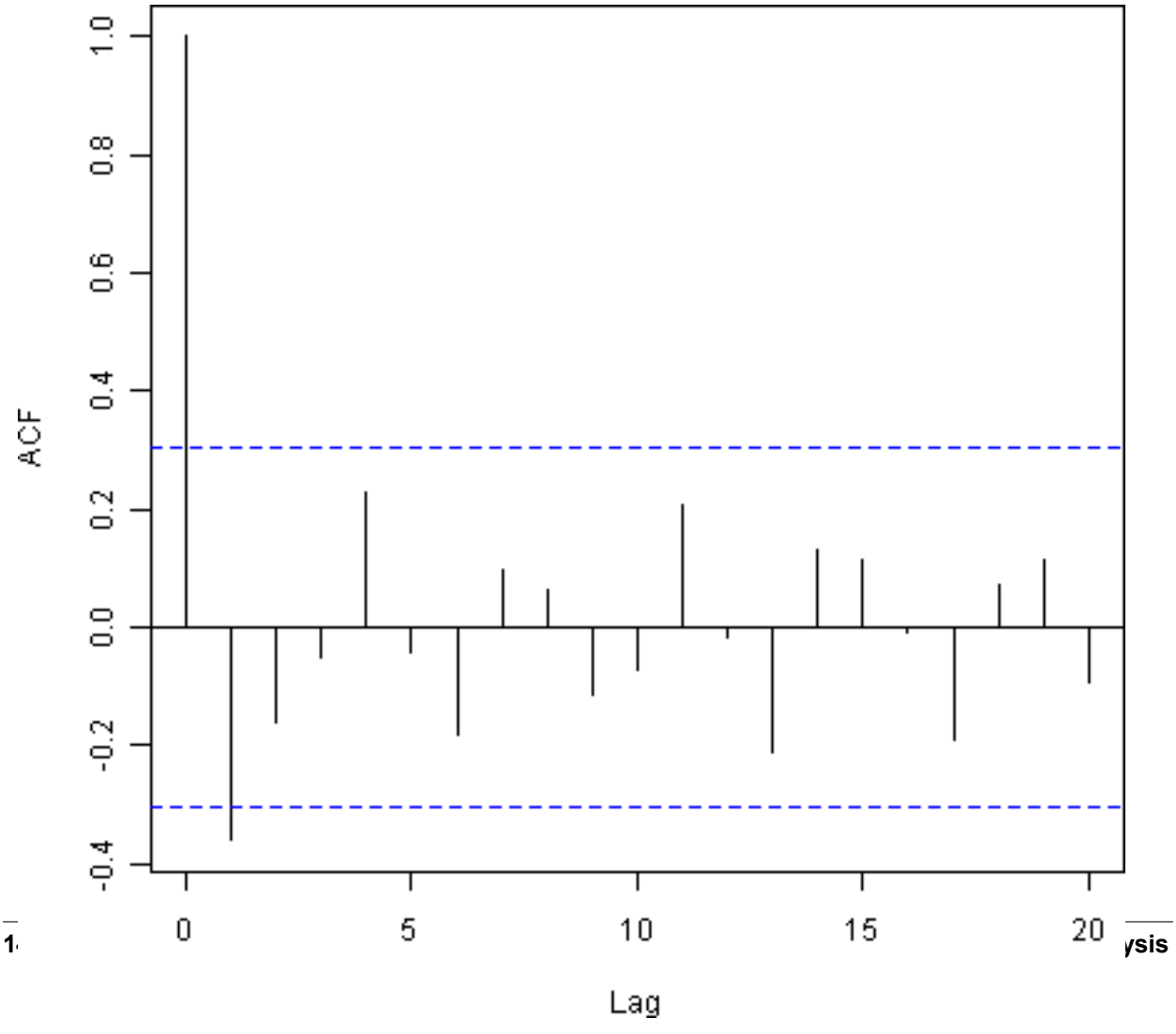
If your time series is stationary, or if you have transformed it to a stationary time series by differencing d times, the next step is to select the appropriate ARIMA model, which means finding the values of most appropriate values of p and q for an ARIMA(p,d,q) model. To do this, you usually need to examine the correlogram and partial correlogram of the stationary time series.

To plot a correlogram and partial correlogram, we can use the "acf()" and "pacf()" functions in R, respectively. To get the actual values of the autocorrelations and partial autocorrelations, we set "plot=FALSE" in the "acf()" and "pacf()" functions.

Example of the Ages at Death of the Kings of England

For example, to plot the correlogram for lags 1-20 of the once differenced time series of the ages at death of the kings of England, and to get the values of the autocorrelations, we type:

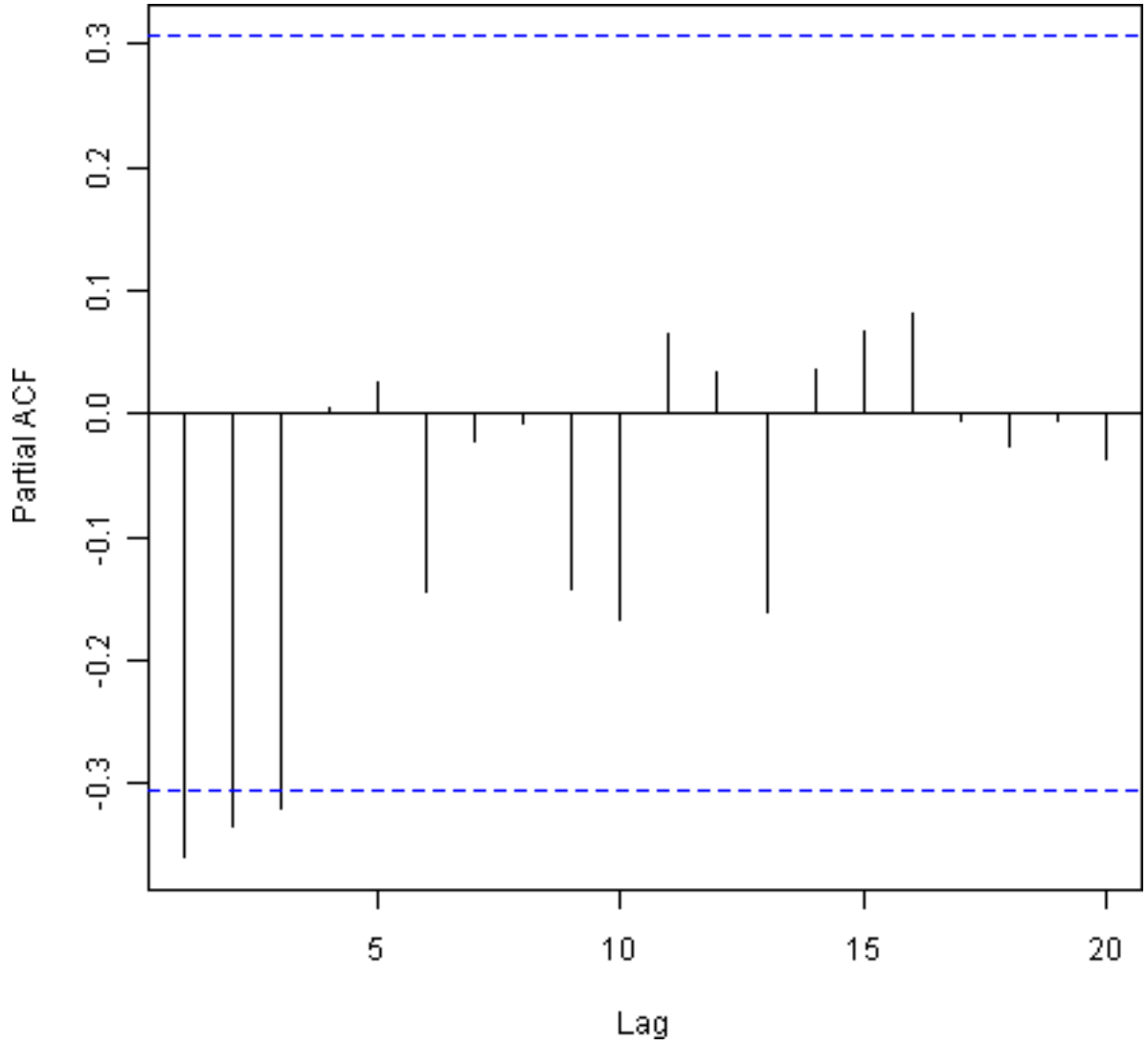
```
> acf(kingtimeseriesdiff1, lag.max=20)             # plot a correlogram
> acf(kingtimeseriesdiff1, lag.max=20, plot=FALSE) # get the autocorrelation values
  Autocorrelations of series 'kingtimeseriesdiff1', by lag
     0      1      2      3      4      5      6      7      8      9     10
 1.000 -0.360 -0.162 -0.050  0.227 -0.042 -0.181  0.095  0.064 -0.116 -0.071
    11     12     13     14     15     16     17     18     19     20
 0.206 -0.017 -0.212  0.130  0.114 -0.009 -0.192  0.072  0.113 -0.093
```



We see from the correlogram that the autocorrelation at lag 1 (-0.360) exceeds the significance bounds, but all other autocorrelations between lags 1-20 do not exceed the significance bounds.

To plot the partial correlogram for lags 1-20 for the once differenced time series of the ages at death of the English kings, and get the values of the partial autocorrelations, we use the "pacf()" function, by typing:

```
> pacf(kingtimeseriesdiff1, lag.max=20)             # plot a partial correlogram
> pacf(kingtimeseriesdiff1, lag.max=20, plot=FALSE) # get the partial autocorrelation
→values
Partial autocorrelations of series 'kingtimeseriesdiff1', by lag
     1      2      3      4      5      6      7      8      9     10     11
-0.360 -0.335 -0.321  0.005  0.025 -0.144 -0.022 -0.007 -0.143 -0.167  0.065
    12     13     14     15     16     17     18     19     20
 0.034 -0.161  0.036  0.066  0.081 -0.005 -0.027 -0.006 -0.037
```



The partial correlogram shows that the partial autocorrelations at lags 1, 2 and 3 exceed the significance bounds, are negative, and are slowly decreasing in magnitude with increasing lag (lag 1: -0.360, lag 2: -0.335, lag 3:-0.321). The partial autocorrelations tail off to zero after lag 3.

Since the correlogram is zero after lag 1, and the partial correlogram tails off to zero after lag 3, this means that the following ARMA (autoregressive moving average) models are possible for the time series of first differences:

- an ARMA(3,0) model, that is, an autoregressive model of order p=3, since the partial autocorrelogram is zero after lag 3, and the autocorrelogram tails off to zero (although perhaps too abruptly for this model to be appropriate)

- an ARMA(0,1) model, that is, a moving average model of order q=1, since the autocorrelogram is zero after lag 1 and the partial autocorrelogram tails off to zero
- an ARMA(p,q) model, that is, a mixed model with p and q greater than 0, since the autocorrelogram and partial correlogram tail off to zero (although the correlogram probably tails off to zero too abruptly for this model to be appropriate)

We use the principle of parsimony to decide which model is best: that is, we assume that the model with the fewest parameters is best. The ARMA(3,0) model has 3 parameters, the ARMA(0,1) model has 1 parameter, and the ARMA(p,q) model has at least 2 parameters. Therefore, the ARMA(0,1) model is taken as the best model.

An ARMA(0,1) model is a moving average model of order 1, or MA(1) model. This model can be written as: X_t - mu = Z_t - (theta * Z_t-1), where X_t is the stationary time series we are studying (the first differenced series of ages at death of English kings), mu is the mean of time series X_t, Z_t is white noise with mean zero and constant variance, and theta is a parameter that can be estimated.

MA (moving average) model is usually used to model a time series that shows short-term dependencies between successive observations. Intuitively, it makes good sense that a MA model can be used to describe the irregular component in the time series of ages at death of English kings, as we might expect the age at death of a particular English king to have some effect on the ages at death of the next king or two, but not much effect on the ages at death of kings that reign much longer after that.

Shortcut: the auto.arima() function

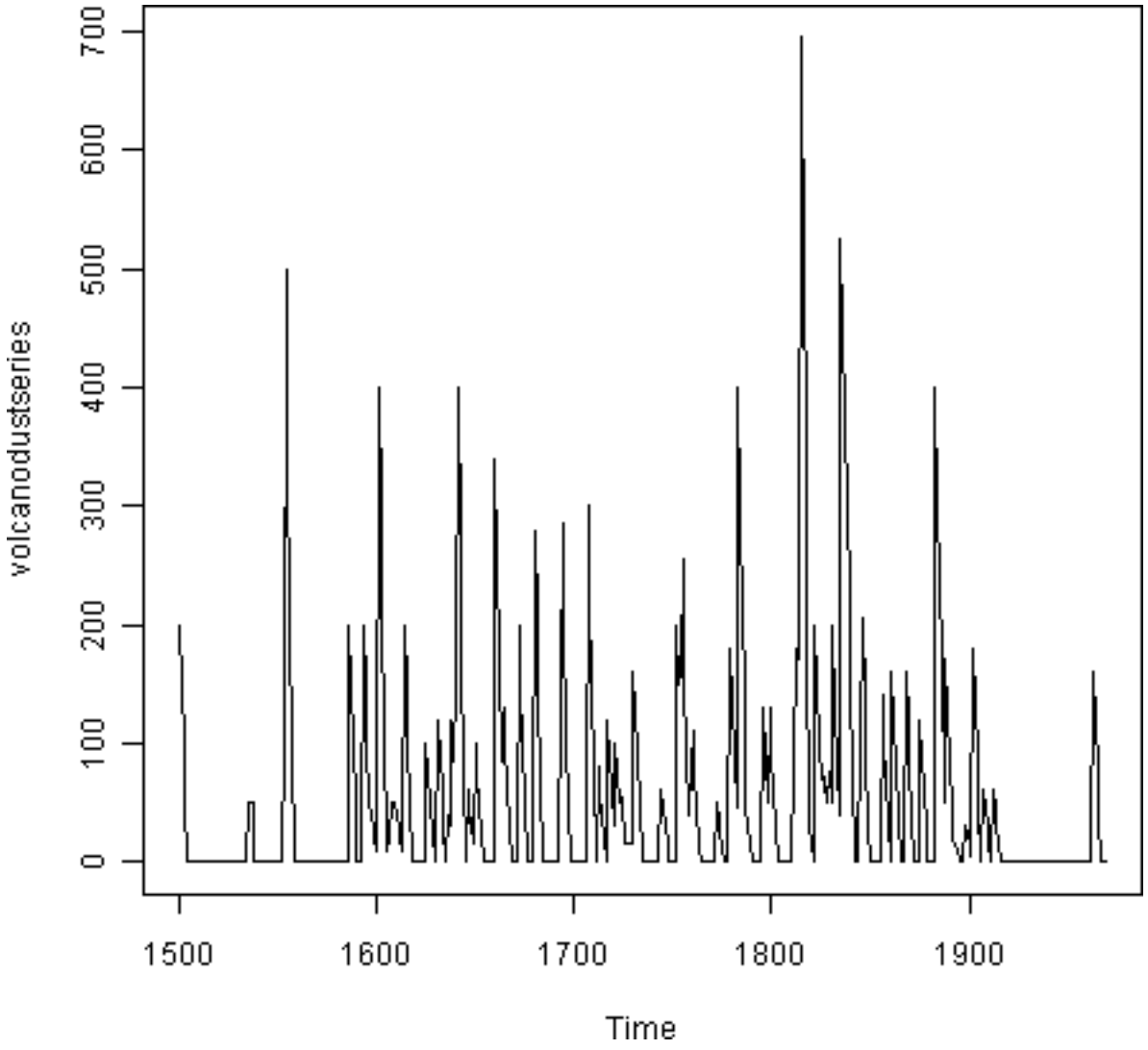
The auto.arima() function can be used to find the appropriate ARIMA model, eg., type "library(forecast)", then "auto.arima(kings)". The output says an appropriate model is ARIMA(0,1,1).

Since an ARMA(0,1) model (with p=0, q=1) is taken to be the best candidate model for the time series of first differences of the ages at death of English kings, then the original time series of the ages of death can be modelled using an ARIMA(0,1,1) model (with p=0, d=1, q=1, where d is the order of differencing required).

Example of the Volcanic Dust Veil in the Northern Hemisphere

Let's take another example of selecting an appropriate ARIMA model. The file file http://robjhyndman.com/tsdldata/annual/dvi.dat contains data on the volcanic dust veil index in the northern hemisphere, from 1500-1969 (original data from Hipel and Mcleod, 1994). This is a measure of the impact of volcanic eruptions' release of dust and aerosols into the environment. We can read it into R and make a time plot by typing:

```
> volcanodust <- scan("http://robjhyndman.com/tsdldata/annual/dvi.dat", skip=1)
> volcanodustseries <- ts(volcanodust,start=c(1500))
> plot.ts(volcanodustseries)
```

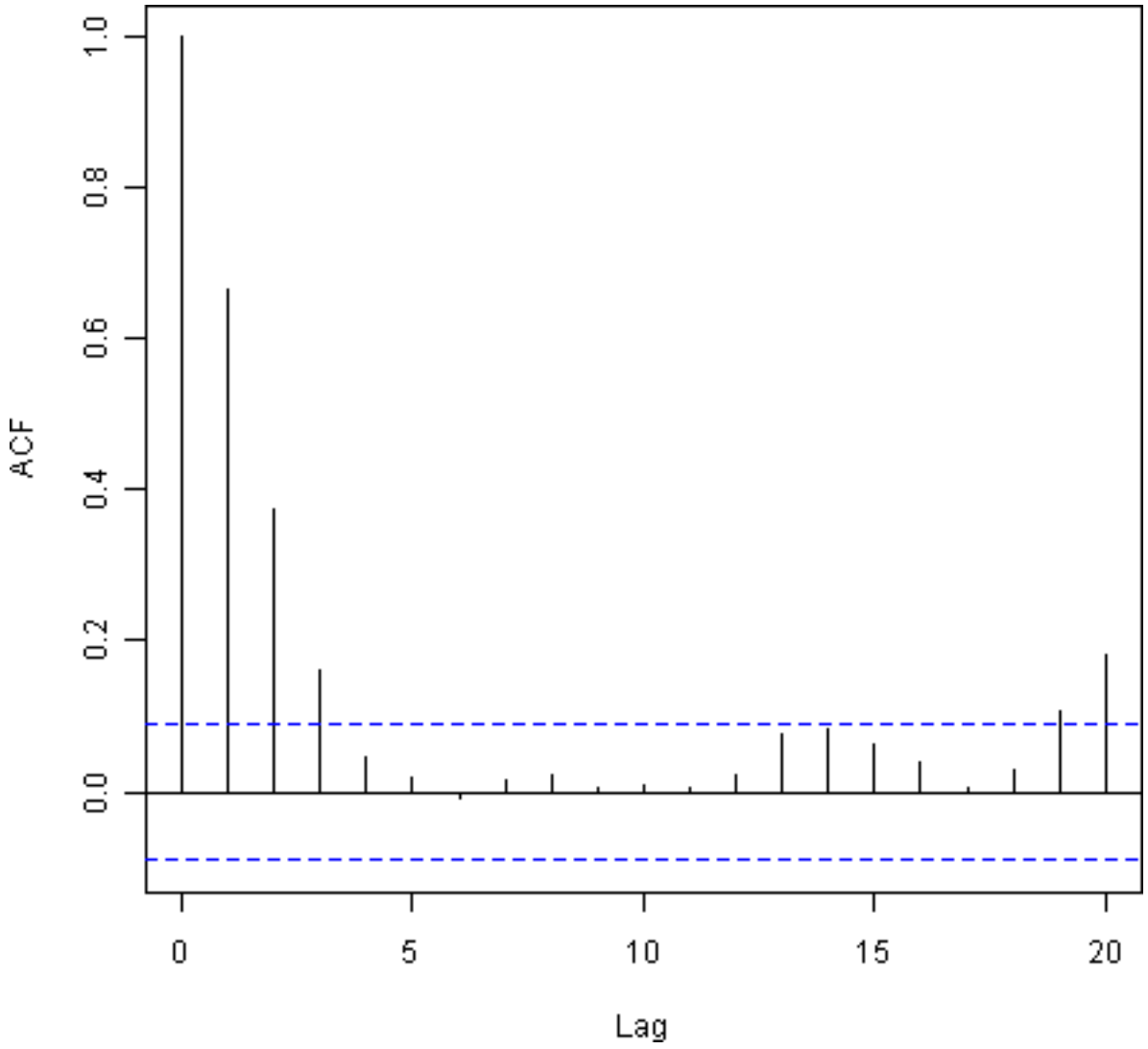


From the time plot, it appears that the random fluctuations in the time series are roughly constant in size over time, so an additive model is probably appropriate for describing this time series.

Furthermore, the time series appears to be stationary in mean and variance, as its level and variance appear to be roughly constant over time. Therefore, we do not need to difference this series in order to fit an ARIMA model, but can fit an ARIMA model to the original series (the order of differencing required, d, is zero here).

We can now plot a correlogram and partial correlogram for lags 1-20 to investigate what ARIMA model to use:

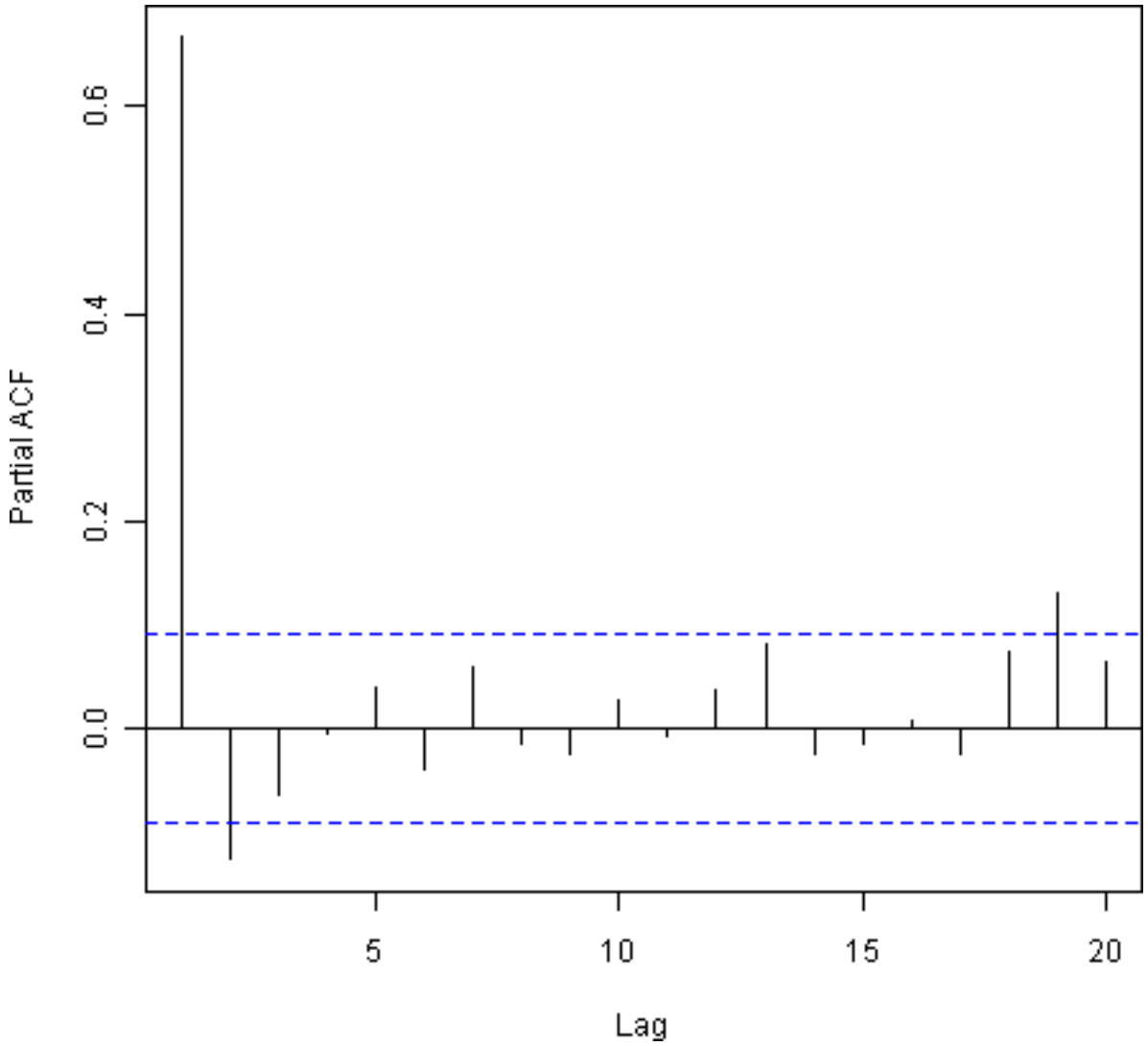
```
> acf(volcanodustseries, lag.max=20)             # plot a correlogram
> acf(volcanodustseries, lag.max=20, plot=FALSE) # get the values of the␣
→autocorrelations
  Autocorrelations of series 'volcanodustseries', by lag
     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.666  0.374  0.162  0.046  0.017 -0.007  0.016  0.021  0.006  0.010
    11     12     13     14     15     16     17     18     19     20
 0.004  0.024  0.075  0.082  0.064  0.039  0.005  0.028  0.108  0.182
```



We see from the correlogram that the autocorrelations for lags 1, 2 and 3 exceed the significance bounds, and that the autocorrelations tail off to zero after lag 3. The autocorrelations for lags 1, 2, 3 are positive, and decrease in magnitude with increasing lag (lag 1: 0.666, lag 2: 0.374, lag 3: 0.162).

The autocorrelation for lags 19 and 20 exceed the significance bounds too, but it is likely that this is due to chance, since they just exceed the significance bounds (especially for lag 19), the autocorrelations for lags 4-18 do not exceed the signifiance bounds, and we would expect 1 in 20 lags to exceed the 95% significance bounds by chance alone.

```
> pacf(volcanodustseries, lag.max=20)
> pacf(volcanodustseries, lag.max=20, plot=FALSE)
  Partial autocorrelations of series 'volcanodustseries', by lag
     1      2      3      4      5      6      7      8      9     10     11
 0.666 -0.126 -0.064 -0.005  0.040 -0.039  0.058 -0.016 -0.025  0.028 -0.008
    12     13     14     15     16     17     18     19     20
 0.036  0.082 -0.025 -0.014  0.008 -0.025  0.073  0.131  0.063
```



From the partial autocorrelogram, we see that the partial autocorrelation at lag 1 is positive and exceeds the significance bounds (0.666), while the partial autocorrelation at lag 2 is negative and also exceeds the significance bounds (-0.126). The partial autocorrelations tail off to zero after lag 2.

Since the correlogram tails off to zero after lag 3, and the partial correlogram is zero after lag 2, the following ARMA models are possible for the time series:

- an ARMA(2,0) model, since the partial autocorrelogram is zero after lag 2, and the correlogram tails off to zero after lag 3, and the partial correlogram is zero after lag 2
- an ARMA(0,3) model, since the autocorrelogram is zero after lag 3, and the partial correlogram tails off to zero (although perhaps too abruptly for this model to be appropriate)
- an ARMA(p,q) mixed model, since the correlogram and partial correlogram tail off to zero (although the partial correlogram perhaps tails off too abruptly for this model to be appropriate)

Shortcut: the auto.arima() function

Again, we can use auto.arima() to find an appropriate model, by typing "auto.arima(volcanodust)", which gives us

ARIMA(1,0,2), which has 3 parameters. However, different criteria can be used to select a model (see auto.arima() help page). If we use the "bic" criterion, which penalises the number of parameters, we get ARIMA(2,0,0), which is ARMA(2,0): " auto.arima(volcanodust,ic='bic') ".

The ARMA(2,0) model has 2 parameters, the ARMA(0,3) model has 3 parameters, and the ARMA(p,q) model has at least 2 parameters. Therefore, using the principle of parsimony, the ARMA(2,0) model and ARMA(p,q) model are equally good candidate models.

An ARMA(2,0) model is an autoregressive model of order 2, or AR(2) model. This model can be written as:

X_t - mu = (Beta1 * (X_t-1 - mu)) + (Beta2 * (Xt-2 - mu)) + Z_t,

- where X_t is the stationary time series we are studying (the time series of volcanic dust veil index),
- mu is the mean of time series X_t,
- Beta1 and Beta2 are parameters to be estimated,
- and Z_t is white noise with mean zero and constant variance.

An AR (autoregressive) model is usually used to model a time series which shows longer term dependencies between successive observations. Intuitively, it makes sense that an AR model could be used to describe the time series of volcanic dust veil index, as we would expect volcanic dust and aerosol levels in one year to affect those in much later years, since the dust and aerosols are unlikely to disappear quickly.

If an ARMA(2,0) model (with p=2, q=0) is used to model the time series of volcanic dust veil index, it would mean that an ARIMA(2,0,0) model can be used (with p=2, d=0, q=0, where d is the order of differencing required). Similarly, if an ARMA(p,q) mixed model is used, where p and q are both greater than zero, than an ARIMA(p,0,q) model can be used.

2.1.3 Forecasting Using an ARIMA Model

Once you have selected the best candidate ARIMA(p,d,q) model for your time series data, you can estimate the parameters of that ARIMA model, and use that as a predictive model for making forecasts for future values of your time series.

You can estimate the parameters of an ARIMA(p,d,q) model using the "arima()" function in R.

Example of the Ages at Death of the Kings of England

For example, we discussed above that an ARIMA(0,1,1) model seems a plausible model for the ages at deaths of the kings of England. You can specify the values of p, d and q in the ARIMA model by using the "order" argument of the "arima()" function in R. To fit an ARIMA(p,d,q) model to this time series (which we stored in the variable "kingstimeseries", see above), we type:

```
kingstimeseriesarima <- arima(kingstimeseries, order=c(0,1,1)) # fit an ARIMA(0,1,1) model
> kingstimeseriesarima

  ARIMA(0,1,1)
    Coefficients:
             ma1
          -0.7218
  s.e.    0.1208
  sigma^2 estimated as 230.4:  log likelihood = -170.06
  AIC = 344.13   AICc = 344.44   BIC = 347.56
```

As mentioned above, if we are fitting an ARIMA(0,1,1) model to our time series, it means we are fitting an an ARMA(0,1) model to the time series of first differences. An ARMA(0,1) model can be written X_t - mu = Z_t - (theta * Z_t-1), where theta is a parameter to be estimated. From the output of the "arima()" R function (above), the estimated value of theta (given as 'ma1' in the R output) is -0.7218 in the case of the ARIMA(0,1,1) model fitted to the time series of ages at death of kings.

Specifying the confidence level for prediction intervals

You can specify the confidence level for prediction intervals in forecast() by using the "level" argument. For example, to get a 99.5% prediction interval, we would type:

```
> kingstimeseriesforecasts <- forecast(kingstimeseriesarima, h=5, level=c(99.5))
> kingstimeseriesforecasts
```

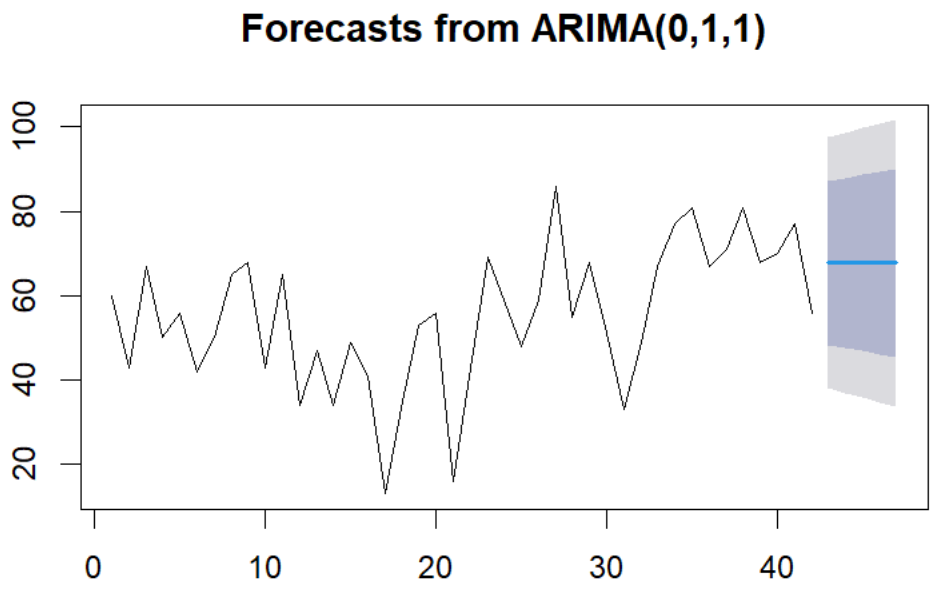
We can then use the ARIMA model to make forecasts for future values of the time series, using the "forecast()" function in the "forecast" R package. For example, to forecast the ages at death of the next five English kings without setting the level of significance, we type:

```
> library("forecast") # load the "forecast" R library
> kingstimeseriesforecasts <- forecast(kingstimeseriesarima, h=5)
> kingstimeseriesforecasts
    Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
43       67.75063 48.29647 87.20479 37.99806  97.50319
44       67.75063 47.55748 87.94377 36.86788  98.63338
45       67.75063 46.84460 88.65665 35.77762  99.72363
46       67.75063 46.15524 89.34601 34.72333 100.77792
47       67.75063 45.48722 90.01404 33.70168 101.79958
```

The original time series for the English kings includes the ages at death of 42 English kings. The forecast() function gives us a forecast of the age of death of the next five English kings (kings 43-47), as well as 80% and 95% prediction intervals for those predictions. The age of death of the 42nd English king was 56 years (the last observed value in our time series), and the ARIMA model gives the forecasted age at death of the next five kings as 67.8 years.

We can plot the observed ages of death for the first 42 kings, as well as the ages that would be predicted for these 42 kings and for the next 5 kings using our ARIMA(0,1,1) model, by typing:

```
> plot(kingstimeseriesforecasts)
```

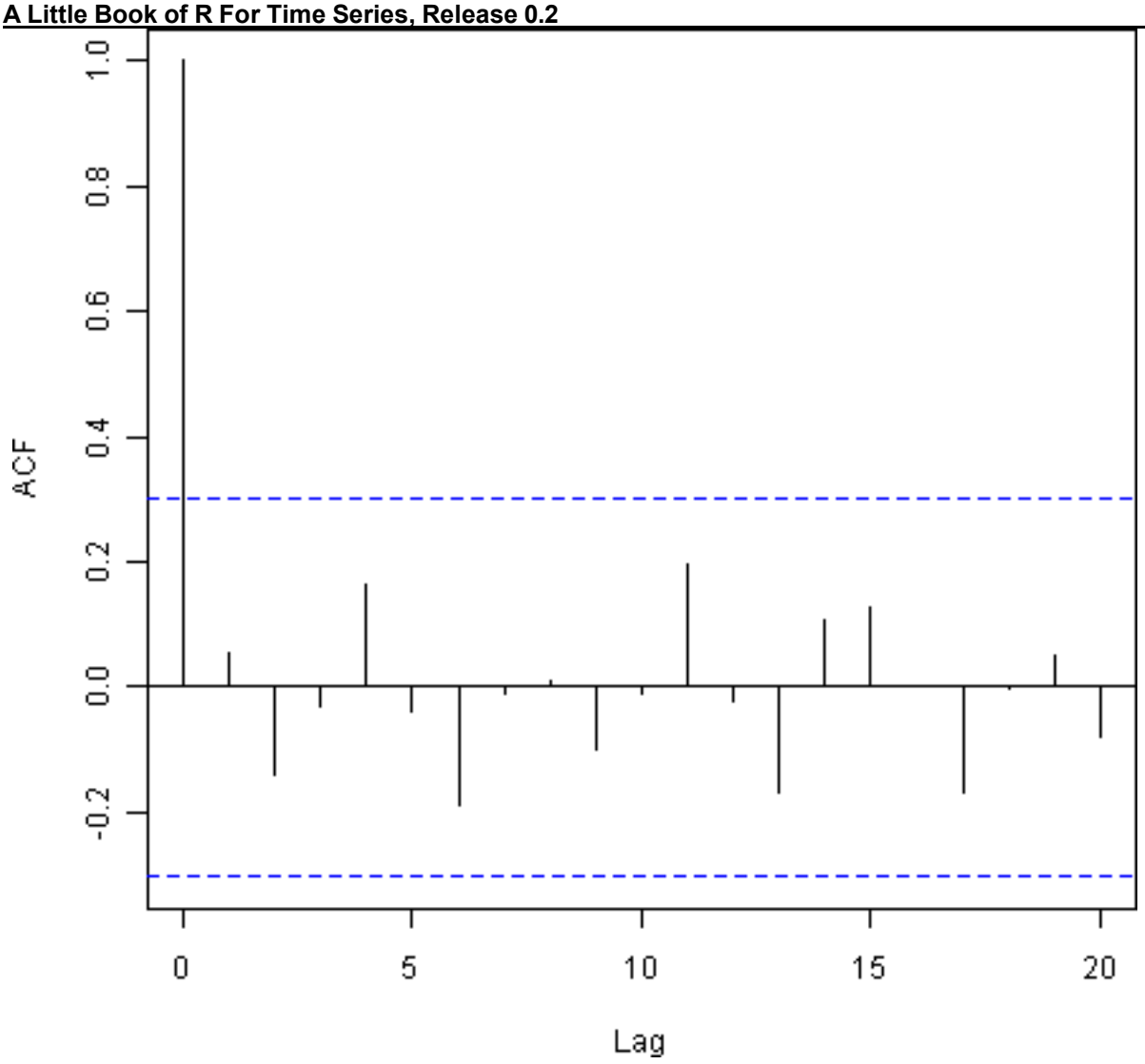


As in the case of exponential smoothing models, it is a good idea to investigate whether the forecast errors of an ARIMA model are normally distributed with mean zero and constant variance, and whether there are correlations be- tween successive forecast errors.

For example, we can make a correlogram of the forecast errors for our ARIMA(0,1,1) model for the ages at death of kings, and perform the Ljung-Box test for lags 1-20, by typing:

```
> acf(kingstimeseriesforecasts$residuals, lag.max=20)
> Box.test(kingstimeseriesforecasts$residuals, lag=20, type="Ljung-Box")

  Box-Ljung test
  data:  kingstimeseriesforecasts$residuals
  X-squared = 13.5844, df = 20, p-value = 0.851
```

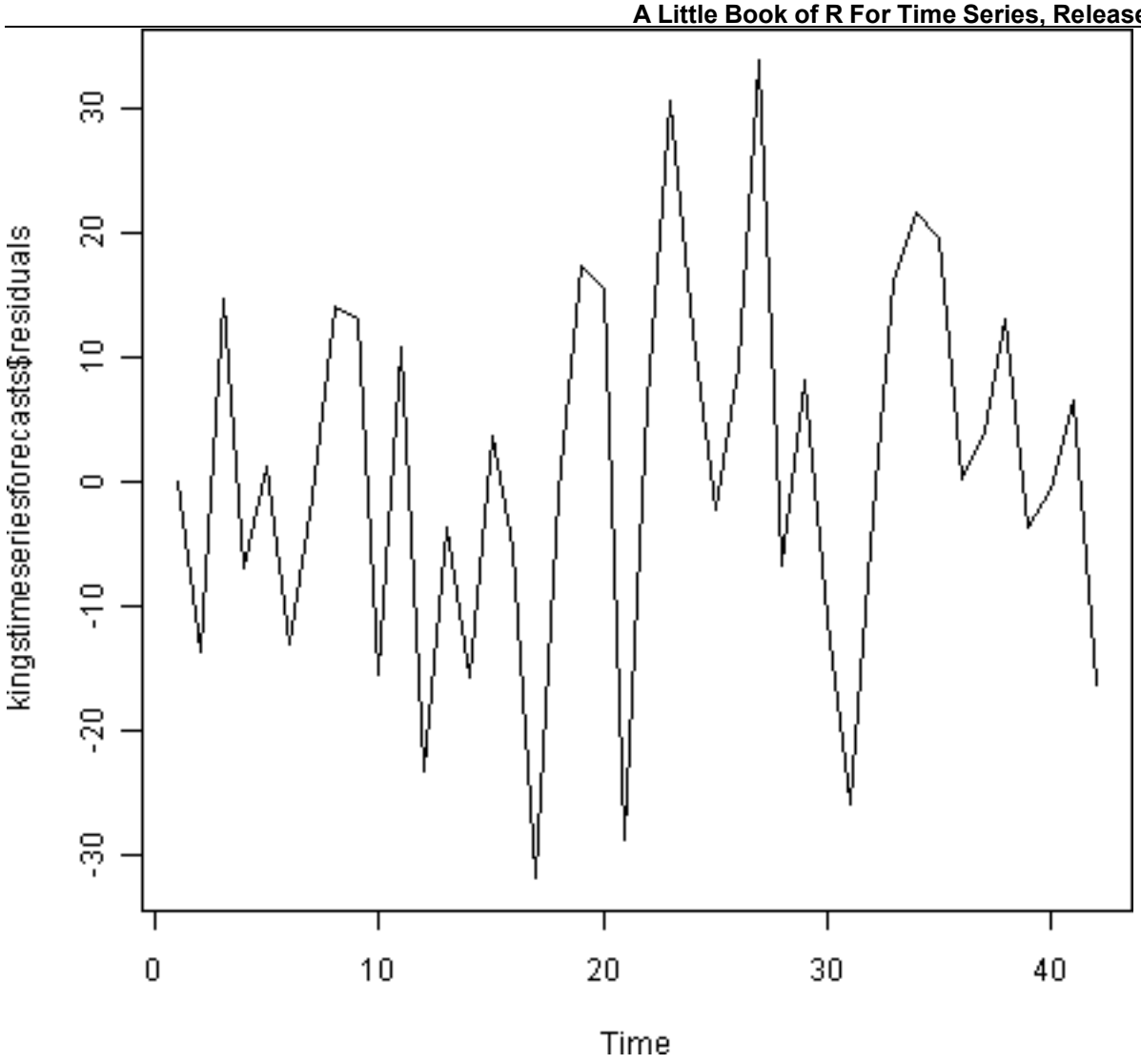


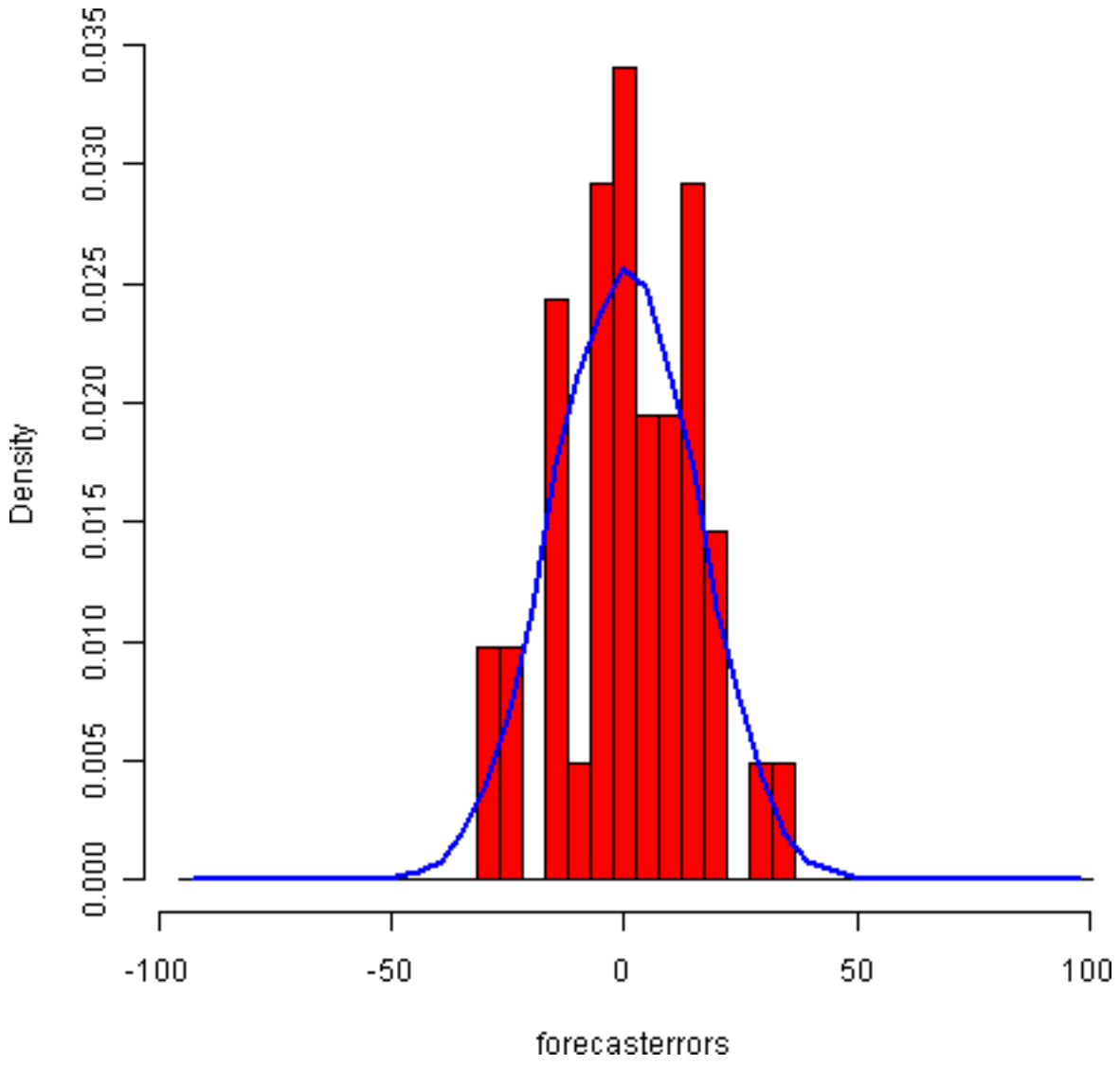
Since the correlogram shows that none of the sample autocorrelations for lags 1-20 exceed the significance bounds, and the p-value for the Ljung-Box test is 0.9, we can conclude that there is very little evidence for non-zero autocorrelations in the forecast errors at lags 1-20.

To investigate whether the forecast errors are normally distributed with mean zero and constant variance, we can make a time plot and histogram (with overlaid normal curve) of the forecast errors:

```
> plot.ts(kingstimeseriesforecasts$residuals)            # time plot forecast error

> plotForecastErrors(kingstimeseriesforecasts$residuals) # make a histogram
```


kingstimeseriesforecasts$residuals
30
20
10
0
-10
-20
-30
0
10
20
30
40
Time



The time plot of the in-sample forecast errors shows that the variance of the forecast errors seems to be roughly constant over time (though perhaps there is slightly higher variance for the second half of the time series). The histogram of the time series shows that the forecast errors are roughly normally distributed, and the mean seems to be close to zero. Therefore, it is plausible that the forecast errors are normally distributed with mean zero and constant variance.

Since successive forecast errors do not seem to be correlated, and the forecast errors seem to be normally distributed with mean zero and constant variance, the ARIMA(0,1,1) does seem to provide an adequate predictive model for the ages at death of English kings.

Example of the Volcanic Dust Veil in the Northern Hemisphere

We discussed above that an appropriate ARIMA model for the time series of volcanic dust veil index may be an ARIMA(2,0,0) model. To fit an ARIMA(2,0,0) model to this time series, we can type:

```
> volcanodustseriesarima <- arima(volcanodustseries, order=c(2,0,0))
> volcanodustseriesarima
  ARIMA(2,0,0) with non-zero mean
```

(continues on next page)

(continued from previous page)

```
  Coefficients:
     ar1      ar2  intercept
  0.7533  -0.1268    57.5274
  s.e.  0.0457   0.0458     8.5958
  sigma^2 estimated as 4870:  log likelihood = -2662.54
  AIC = 5333.09   AICc = 5333.17   BIC = 5349.7
```

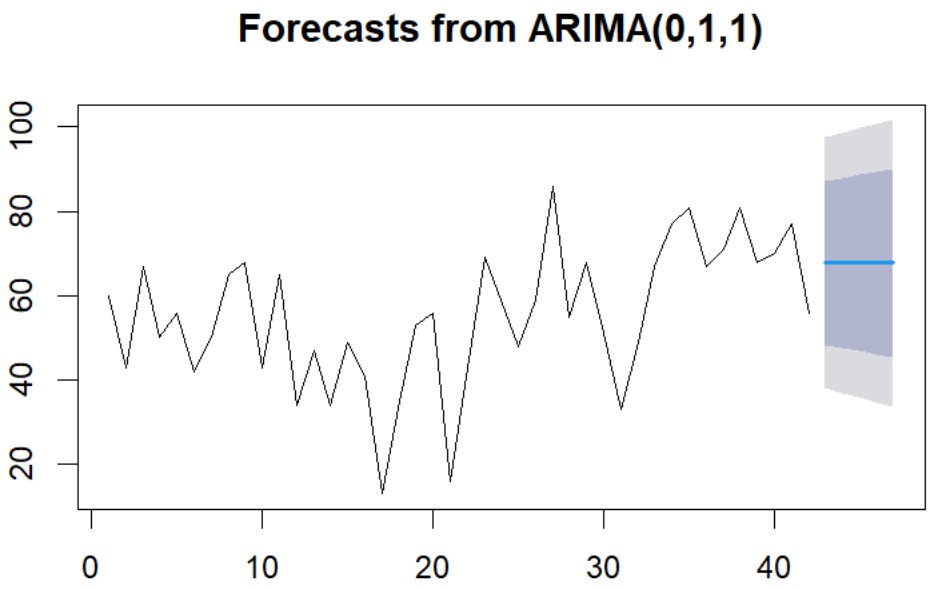
As mentioned above, an ARIMA(2,0,0) model can be written as: written as: X_t - mu = (Beta1 * (X_t-1 - mu)) + (Beta2 * (Xt-2 - mu)) + Z_t, where Beta1 and Beta2 are parameters to be estimated. The output of the arima() function tells us that Beta1 and Beta2 are estimated as 0.7533 and -0.1268 here (given as ar1 and ar2 in the output of arima()).

Now we have fitted the ARIMA(2,0,0) model, we can use the "forecast()" model to predict future values of the volcanic dust veil index. The original data includes the years 1500-1969. To make predictions for the years 1970-2000 (31 more years), we type:

```
> volcanodustseriesforecasts <- forecast(volcanodustseriesarima, h=31)
> volcanodustseriesforecasts
  Point       Forecast      Lo 80     Hi 80      Lo 95    Hi 95
  1970       21.48131 -67.94860 110.9112 -115.2899 158.2526
  1971       37.66419 -74.30305 149.6314 -133.5749 208.9033
  1972       47.13261 -71.57070 165.8359 -134.4084 228.6737
  1973       52.21432 -68.35951 172.7881 -132.1874 236.6161
  1974       54.84241 -66.22681 175.9116 -130.3170 240.0018
  1975       56.17814 -65.01872 177.3750 -129.1765 241.5327
  1976       56.85128 -64.37798 178.0805 -128.5529 242.2554
  1977       57.18907 -64.04834 178.4265 -128.2276 242.6057
  1978       57.35822 -63.88124 178.5977 -128.0615 242.7780
  1979       57.44283 -63.79714 178.6828 -127.9777 242.8634
  1980       57.48513 -63.75497 178.7252 -127.9356 242.9059
  1981       57.50627 -63.73386 178.7464 -127.9145 242.9271
  1982       57.51684 -63.72330 178.7570 -127.9040 242.9376
  1983       57.52212 -63.71802 178.7623 -127.8987 242.9429
  1984       57.52476 -63.71538 178.7649 -127.8960 242.9456
  1985       57.52607 -63.71407 178.7662 -127.8947 242.9469
  1986       57.52673 -63.71341 178.7669 -127.8941 242.9475
  1987       57.52706 -63.71308 178.7672 -127.8937 242.9479
  1988       57.52723 -63.71291 178.7674 -127.8936 242.9480
  1989       57.52731 -63.71283 178.7674 -127.8935 242.9481
  1990       57.52735 -63.71279 178.7675 -127.8934 242.9481
  1991       57.52737 -63.71277 178.7675 -127.8934 242.9482
  1992       57.52738 -63.71276 178.7675 -127.8934 242.9482
  1993       57.52739 -63.71275 178.7675 -127.8934 242.9482
  1994       57.52739 -63.71275 178.7675 -127.8934 242.9482
  1995       57.52739 -63.71275 178.7675 -127.8934 242.9482
  1996       57.52739 -63.71275 178.7675 -127.8934 242.9482
  1997       57.52739 -63.71275 178.7675 -127.8934 242.9482
  1998       57.52739 -63.71275 178.7675 -127.8934 242.9482
  1999       57.52739 -63.71275 178.7675 -127.8934 242.9482
  2000       57.52739 -63.71275 178.7675 -127.8934 242.9482
```

We can plot the original time series, and the forecasted values, by typing:

```
> plot(volcanodustseriesforecasts)
```

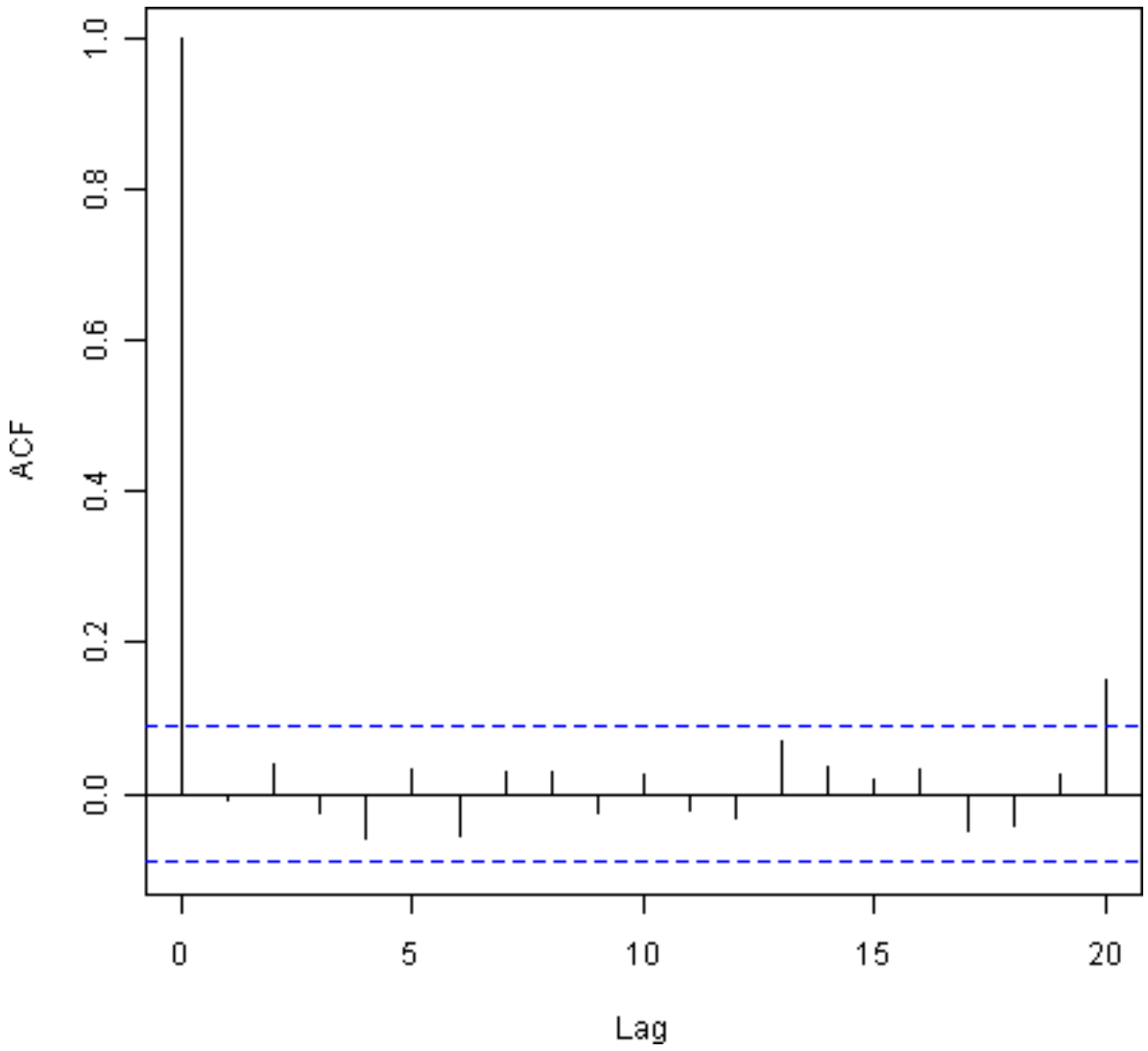


One worrying thing is that the model has predicted negative values for the volcanic dust veil index, but this variable can only have positive values! The reason is that the arima() and forecast() functions don't know that the variable can only take positive values. Clearly, this is not a very desirable feature of our current predictive model.

Again, we should investigate whether the forecast errors seem to be correlated, and whether they are normally distributed with mean zero and constant variance. To check for correlations between successive forecast errors, we can make a correlogram and use the Ljung-Box test:

```
> acf(volcanodustseriesforecasts$residuals, lag.max=20)
> Box.test(volcanodustseriesforecasts$residuals, lag=20, type="Ljung-Box")

  Box-Ljung test
  data:  volcanodustseriesforecasts$residuals
  X-squared = 24.3642, df = 20, p-value = 0.2268
```

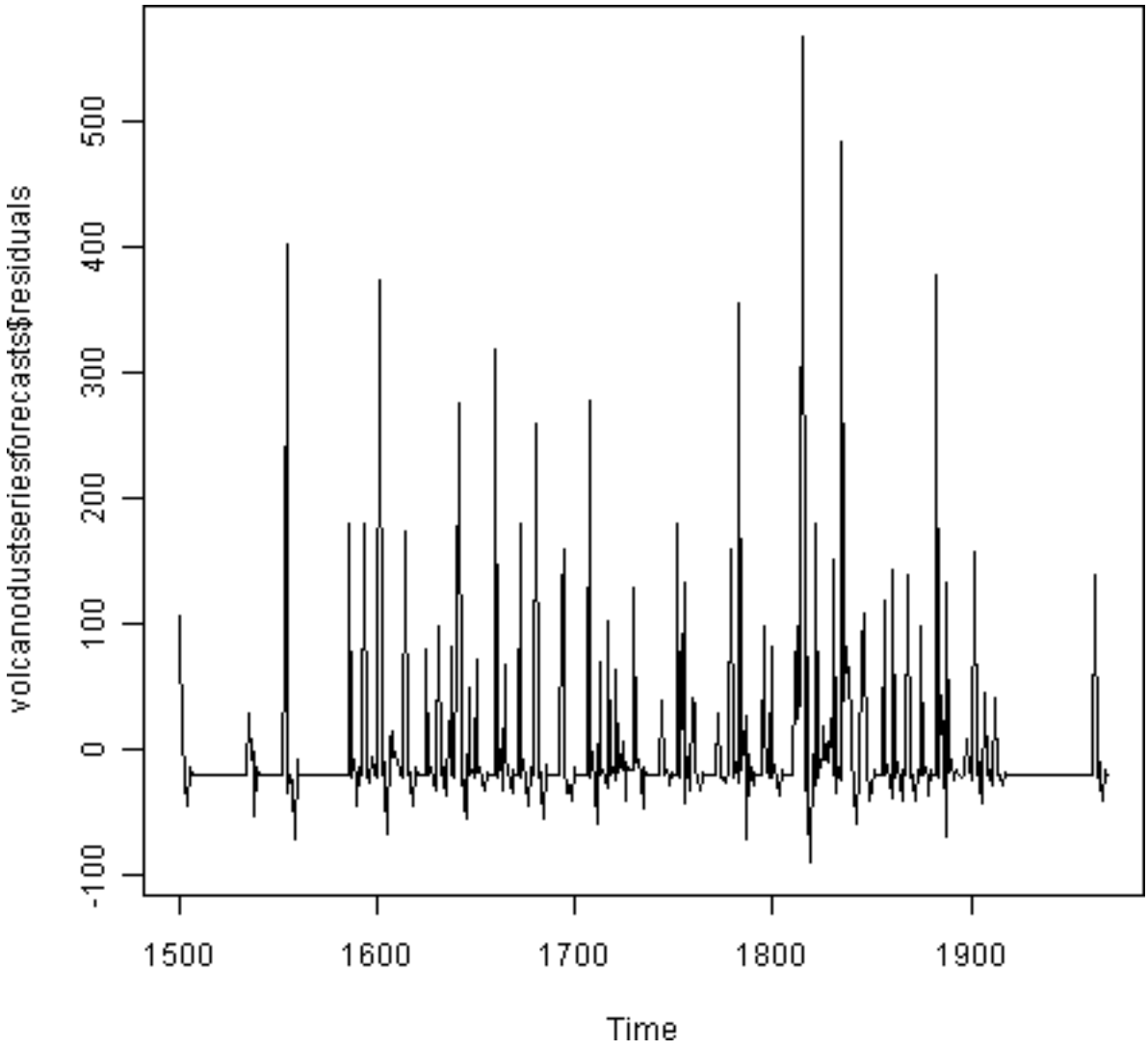


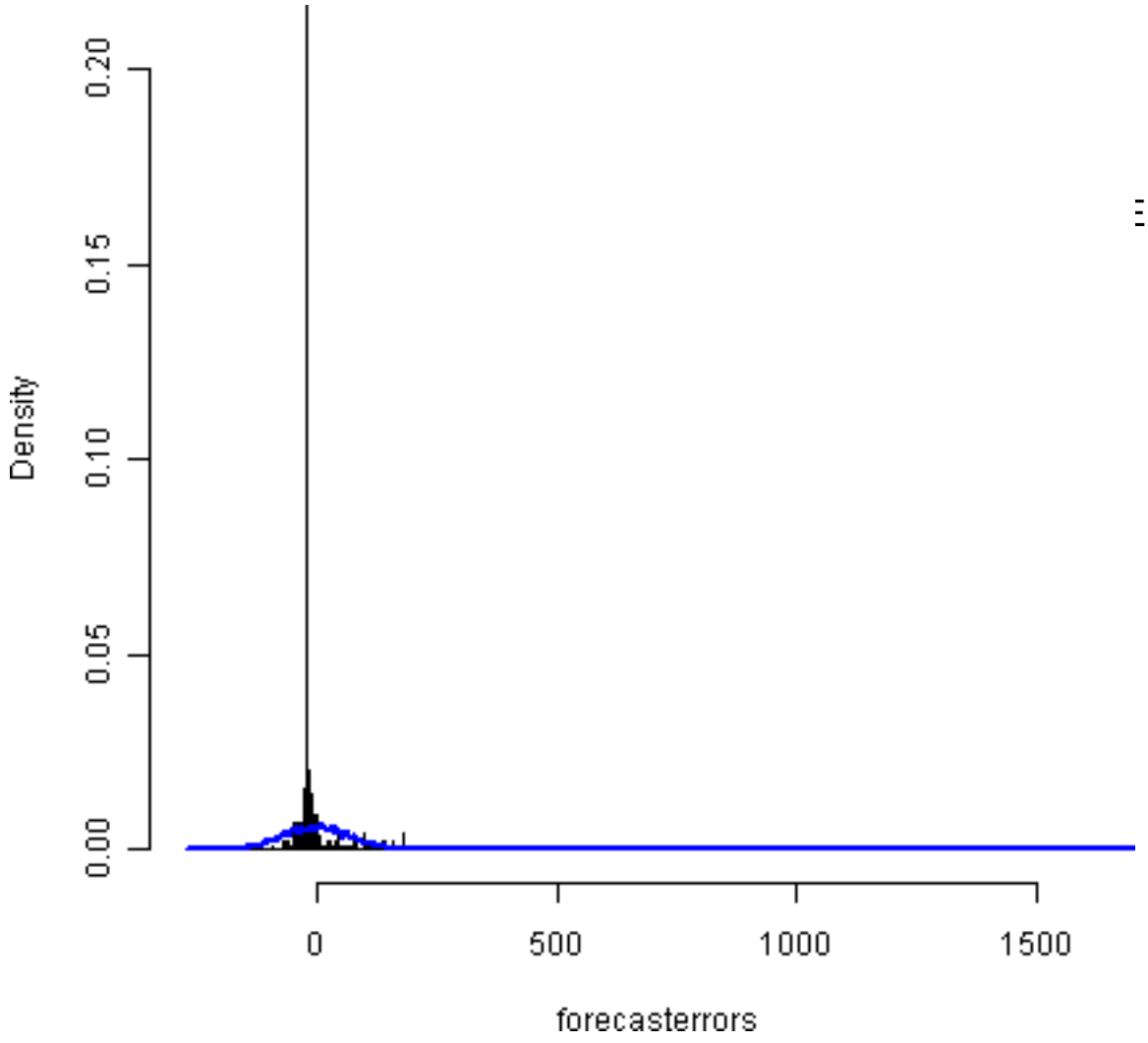
The correlogram shows that the sample autocorrelation at lag 20 exceeds the significance bounds. However, this is probably due to chance, since we would expect one out of 20 sample autocorrelations to exceed the 95% significance bounds. Furthermore, the p-value for the Ljung-Box test is 0.2, indicating that there is little evidence for non-zero autocorrelations in the forecast errors for lags 1-20.

To check whether the forecast errors are normally distributed with mean zero and constant variance, we make a time plot of the forecast errors, and a histogram:

```
> plot.ts(volcanodustseriesforecasts$residuals)            # plot of errors

> plotForecastErrors(volcanodustseriesforecasts$residuals) # make a histogram
```


volcanodustseriesforecasts$residuals
-100
0
100
200
300
400
500
1500
1600
1700
1800
1900
Time



The time plot of forecast errors shows that the forecast errors seem to have roughly constant variance over time. However, the time series of forecast errors seems to have a negative mean, rather than a zero mean. We can confirm this by calculating the mean forecast error, which turns out to be about -0.22:

```
> mean(volcanodustseriesforecasts$residuals)
  -0.2205417
```

The histogram of forecast errors (above) shows that although the mean value of the forecast errors is negative, the distribution of forecast errors is skewed to the right compared to a normal curve. Therefore, it seems that we cannot comfortably conclude that the forecast errors are normally distributed with mean zero and constant variance! Thus, it is likely that our ARIMA(2,0,0) model for the time series of volcanic dust veil index is not the best model that we could make, and could almost definitely be improved upon!

2.2 SARIMA Models:

As you finish this ARIMA workshop, remember that many real-world time series contain both trend and seasonality. When seasonality is present, ARIMA on its own is not enough — you need SARIMA, the seasonal extension of ARIMA. SARIMA models handle:

- trend through ordinary differencing d
- repeating seasonal patterns through seasonal differencing D
- short-term dynamics with AR and MA terms
- seasonal dynamics with seasonal AR and MA terms

In R, running a SARIMA model is straightforward using the forecast package.

How to run a SARIMA model in R: (In this workshop, you don't need to run an actual example; the following line of code is for demonstration and is not associated with a specific dataset.)

```
library(forecast)


# To fit:

fit <- arima(y, order = c(p, d, q), seasonal = c(P, D, Q), period = s)

summary(fit)

# To forecast:

fc <- forecast(fit, h = 12)

plot(fc)
```

Or let R choose the best orders automatically:

```
fit <- auto.arima(y, seasonal = TRUE)

summary(fit)
```

2.3 Links and Further Reading

Here are some links for further reading.

For a more in-depth introduction to R, a good online tutorial is available on the "Kickstarting R" website, cran.r-project.org/doc/contrib/Lemon-kickstart.

There is another nice (slightly more in-depth) tutorial to R available on the "Introduction to R" website, cran.r- project.org/doc/manuals/R-intro.html.

You can find a list of R packages for analysing time series data on the CRAN Time Series Task View webpage.

To learn about time series analysis, I would highly recommend the book "Time series" (product code M249/02) by the Open University, available from the Open University Shop.

There are two books available in the "Use R!" series on using R for time series analyses, the first is Introductory Time Series with R by Cowpertwait and Metcalfe, and the second is Analysis of Integrated and Cointegrated Time Series with R by Pfaff.

2.4 Acknowledgements

I am grateful to Professor Rob Hyndman, for kindly allowing me to use the time series data sets from his Time Series Data Library (TSDL) in the examples in this booklet.

Many of the examples in this booklet are inspired by examples in the excellent Open University book, "Time series" (product code M249/02), available from the Open University Shop.

Thank you to Ravi Aranke for bringing auto.arima() to my attention, and Maurice Omane-Adjepong for bringing unit root tests to my attention, and Christian Seubert for noticing a small bug in plotForecastErrors(). Thank you for other comments to Antoine Binard and Bill Johnston.

2.5 Contact

I will be grateful if you will send me (Avril Coghlan) corrections or suggestions for improvements to my email address alc@sanger.ac.uk

2.6 License