# Tutorial: Time series analysis in R

*D G Rossiter*
*Cornell University, Section of Soil & Crop Sciences*
*ISRIC–World Soil Information*
南京师范大学地理学学院

March 3, 2020

磨刀不误砍柴工
"Take time to sharpen your axe before you start to chop firewood."

## Contents

# 1 Introduction

Time series are repeated measurements of one or more variables over time. They arise in many applications in earth, natural and water resource studies, including ground and surface water monitoring (quality and quantity), climate monitoring and modelling, agricultural or other production statistics. Time series have several characteristics that make their analysis different from that of non-temporal variables:

1. The variable may have a **trend** over time;
2. The variable may exhibit one or more **cycles** of different period;
3. There will almost always be **serial correlation** between subsequent observations;
4. The variable will almost always have a **"white noise"** component, i.e. random variation not explained by any of the above.

This tutorial presents some aspects of time series analysis (shorthand "TSA"[1]), using the **R environment for statistical computing and visualisation** [12, 14] and its dialect of the **S language**. R users seriously interested in this topic are well-advised to refer to the text of Metcalfe and Cowpertwait [13][2], part of the excellent *UseR!* series from Springer[3]. The Comprehensive R Action Network (CRAN) has a "Task View" on time series analysis[4]. This lists all the R packages applicable to TSA, categorized and with a brief description of each. In addition, Shumway and Stoffer [16] is an advanced text which uses R for its examples. Venables and Ripley [18] include a chapter on time series analysis in S (both R and S-PLUS dialects), mostly using examples from Diggle [8].

Good introductions to the concepts of time series analysis are Diggle [8] for biological applications, Box [4] for forecasting and control, Hipel and McLeod [10] (available on the web) for water resources and environmental modelling, and Salas [15] from the Handbook of Hydrology for hydrological applications. Davis [7] includes a brief introduction to time series for geologists. Wilks [20, Ch. 8] discusses time series for climate analysis, with emphasis on simulation.

The tutorial is organized as a set of **tasks** followed by **questions** to check your understanding; **answers** are at the end of each section. If you are ambitious, there are also some challenges: tasks and questions with no solution provided, that require the integration of skills learned in the section.

[1] Not to be confused with a bureaucratic monster with the same initials

[2] `http://link.springer.com/book/10.1007/978-0-387-88698-5`

[3] `http://link.springer.com/bookseries/6991`

[4] `http://cran.r-project.org/web/views/TimeSeries.html`

# 2 Loading and examining a time series

We use two datasets[5] to illustrate different research questions and operations of time series analysis:

1. Monthly groundwater levels (§2.1);
2. Daily rainfall amounts (§2.2).

These also illustrate some of the problems with importing external datasets into R and putting data into a form suitable for time-series analysis.

All the datasets in this exercise are assumed to be stored in the `ds_tsa` "datasets" subdirectory, under the directory where this tutorial is stored.

## 2.1 Example 1: Groundwater level

The first example dataset is a series of measurements of the depth to groundwater (in meters) in two wells used for irrigation in the Anatolian plateau, Turkey, from January 1975 through December 2004 CE. These are provided as text files `anatolia_hati.txt` and `anatolia_alibe.txt`.

---

**Q1 :** *What are some things that a water manager would want to know about this time series?* *Jump to A1* •

---

**Task 1 :** Start R and switch to the directory where the example datasets are stored. •

---

**Task 2 :** Examine the file for the first well. •

You could review the file in a plain-text editor; here we use the `file.show` function:

```
file.show("./ds_tsa/anatolia_hati.txt")
```

Here are the first and last lines of the file:

```
34.36
34.45
34.7
...
55.96
55.55
54.83
```

---

**Q2 :** *Can you tell from this file that it is a time series?* *Jump to A2* •

[5] kindly provided by colleagues in the University of Twente/faculty ITC's Water Resources department, https://www.itc.nl/WRS

**Task 3 :** Read this file into R and examine its structure. •

Using the `scan` function to read a file into a vector:

```
gw <- scan("./ds_tsa/anatolia_hati.txt")
str(gw)

##  num [1:360] 34.4 34.5 34.7 34.8 34.9 ...
```

**Q3 :** *What is the structure?* *Jump to A3* •

**Task 4 :** Convert the vector of measurements for this well into a time series and examine its structure and attributes. •

The `ts` function converts a vector into a time series; it has several arguments, of which enough must be given to specify the series:

- `start` : starting date of the series;
- `end` : ending date of the series;
- `frequency` : number of observations in the series per unit of time;
- `deltat` : fraction of the sampling period between successive observations.

Only one of `frequency` or `deltat` should be provided, they are two ways to specify the same information. The ending date `end` is optional if either `frequency` or `deltat` are given, because the end will just be when the vector ends.

In this example we know from the metadata that the series begins in January 1975 and ends in December 2004; it is a monthly series. The simplest way to specify it is by starting date and frequency.

After the series is established, we examine its structure with `str` and attributes with `attributes`.

```
gw <- ts(gw, start=1975, frequency=12)
str(gw)

##  Time-Series [1:360] from 1975 to 2005: 34.4 34.5 34.7 34.8 34.9 ...

attributes(gw)

## $tsp
## [1] 1975.000 2004.917   12.000
##
## $class
## [1] "ts"

start(gw)

## [1] 1975    1

end(gw)

## [1] 2004   12
```

In the above example we also show the `start` and `end` functions to show the starting and ending dates in a time series.

**Task 5 :** Print the time series; also show the the time associated with each measurement, and the position of each observation in the cycle. •

The generic `print` method specializes into `print.ts` to show the actual values; `time` shows the time associated with each measurement, and `cycle` shows the position of each observation in the cycle.

```
print(gw)
```

Here we just show the first and last two years.

```
       Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
1975 34.36 34.45 34.70 34.80 34.88 35.16 35.60 35.86 35.86 35.70
1976 35.22 35.18 34.98 35.20 35.51 35.32 34.45 34.54 34.39 34.18
...
2003 53.32 52.48 51.37 51.07 50.71 52.78 54.35 55.46 56.52 55.70
2004 53.08 52.41 51.71 51.66 52.84 54.11 55.28 56.11 57.02 55.96
       Nov   Dec
1975 35.48 35.28
1976 33.92 33.73
...
2003 54.47 54.21
2004 55.55 54.83
```

Note how the month names are automatically assigned. From the `ts` documentation: “Values of 4 and 12 are assumed in print methods to imply a quarterly and monthly series respectively.” Specifically `print.ts` makes this assumption.

```
time(gw)
```

Again, only the beginning and end of the full series:

```
          Jan      Feb      Mar      Apr      May      Jun      Jul
1975 1975.000 1975.083 1975.167 1975.250 1975.333 1975.417 1975.500
1976 1976.000 1976.083 1976.167 1976.250 1976.333 1976.417 1976.500
...
2003 2003.000 2003.083 2003.167 2003.250 2003.333 2003.417 2003.500
2004 2004.000 2004.083 2004.167 2004.250 2004.333 2004.417 2004.500
          Aug      Sep      Oct      Nov      Dec
1975 1975.583 1975.667 1975.750 1975.833 1975.917
1976 1976.583 1976.667 1976.750 1976.833 1976.917
...
2003 2003.583 2003.667 2003.750 2003.833 2003.917
2004 2004.583 2004.667 2004.750 2004.833 2004.917
```

```
cycle(gw)
```

Again, only the beginning and end of the full series:

```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1975   1   2   3   4   5   6   7   8   9  10  11  12
1976   1   2   3   4   5   6   7   8   9  10  11  12
...
2003   1   2   3   4   5   6   7   8   9  10  11  12
2004   1   2   3   4   5   6   7   8   9  10  11  12
```

**Q4 :** *What are the units of each of these?* *Jump to A4* •

**Task 6 :** Determine the series' frequency and interval between observations. •

Of course we know this from the metadata, but this information can also be extracted from the series object with the `frequency` and `deltat`, methods, as in the arguments to `ts`.

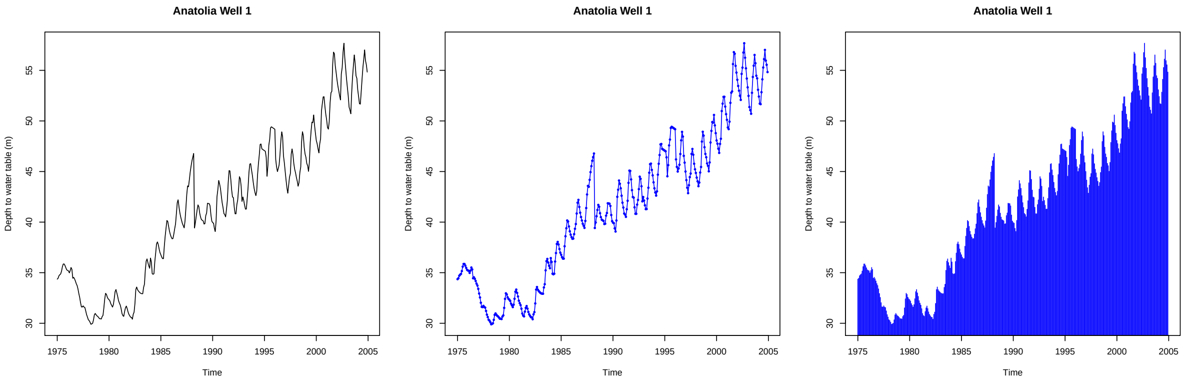
```
frequency(gw)

## [1] 12

deltat(gw)

## [1] 0.08333333
```

**Q5 :** *What are the units of each of these?* *Jump to A5* •

**Task 7 :** Plot the time series. •

The generic `plot` method specializes to `plot.ts` for time-series objects. There are several ways to visualise this, we show a few possibilities here:

```
par(mfrow=c(1,3))
# pdf("AnatoliaWell1.pdf", width=10, height=5)
plot(gw, ylab="Depth to water table (m)", main="Anatolia Well 1")
# dev.off()
plot(gw, type="o", pch=20, cex=0.6, col="blue",
     ylab="Depth to water table (m)", main="Anatolia Well 1")
plot(gw, type="h", col="blue", ylab="Depth to water table (m)",
     main="Anatolia Well 1")
par(mfrow=c(1,1))
```

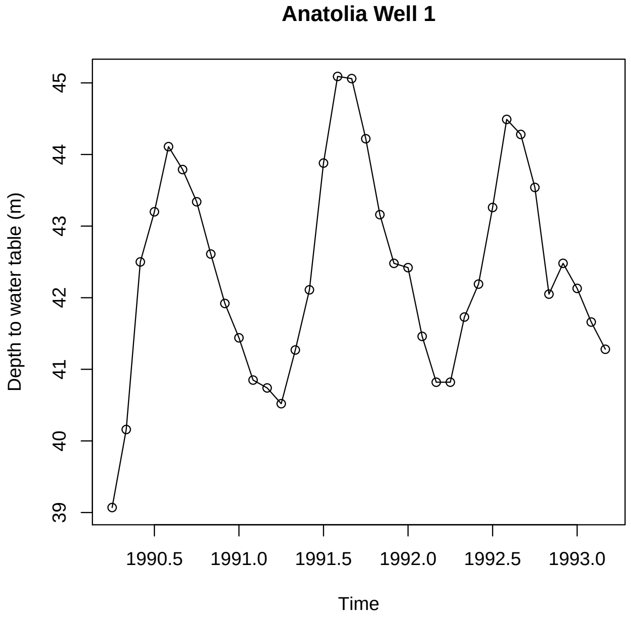


**Q6 :** *What are the outstanding features of this time series?* *Jump to A6* •

**Task 8** : Plot three cycles of the time series, from the shallowest depth at the end of the winter rains (April) beginning in 1990, to see annual behaviour. •

We use the `window` function to extract just part of the series. The `start` and `end` can have both a cycle (here, year) and position in cycle (here, month), connected with the `c` 'catentate' function:

```
window(gw, start=c(1990,4), end=c(1993,3))

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1990                   39.07 40.16 42.50 43.20 44.11 43.79 43.34
## 1991 41.44 40.85 40.74 40.52 41.27 42.11 43.88 45.09 45.06 44.22
## 1992 42.42 41.46 40.82 40.82 41.73 42.19 43.26 44.49 44.28 43.54
## 1993 42.13 41.66 41.28
##        Nov   Dec
## 1990 42.61 41.92
## 1991 43.16 42.48
## 1992 42.05 42.48
## 1993

plot(window(gw, start=c(1990,4), end=c(1993,3)), type="o",
     ylab="Depth to water table (m)", main="Anatolia Well 1")
```



**Q7** : *What is the annual cycle?* *Jump to A7* •

This region of Turkey has a typical Mediterranean climate: hot and dry in the summer, cool and moist in the winter. These wells are used for irrigation in the summer months.

The `window` function may also be used to extract a single element, by specifying the same start and end date:

```
window(gw, start=c(1990,1), end=c(1990,1))
```

```
##        Jan
## 1990 39.95
```

Another way to view a time series is as the **differences** between successive measurements.

**Task 9 :** Compute and view the difference for lag 1 (one month), lag 2 (two months), and lag 12 (one year), for the period 1990 – 1992. •

The `diff` function computes differences, by default for successive measurements; the `lag` argument specifies different lags (intervals between measurements):

```
window(gw, 1990, c(1992,12))

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1990 39.95 39.89 39.38 39.07 40.16 42.50 43.20 44.11 43.79 43.34
## 1991 41.44 40.85 40.74 40.52 41.27 42.11 43.88 45.09 45.06 44.22
## 1992 42.42 41.46 40.82 40.82 41.73 42.19 43.26 44.49 44.28 43.54
##        Nov   Dec
## 1990 42.61 41.92
## 1991 43.16 42.48
## 1992 42.05 42.48

diff(window(gw, 1990, c(1992,12)))

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1990       -0.06 -0.51 -0.31  1.09  2.34  0.70  0.91 -0.32 -0.45
## 1991 -0.48 -0.59 -0.11 -0.22  0.75  0.84  1.77  1.21 -0.03 -0.84
## 1992 -0.06 -0.96 -0.64  0.00  0.91  0.46  1.07  1.23 -0.21 -0.74
##        Nov   Dec
## 1990 -0.73 -0.69
## 1991 -1.06 -0.68
## 1992 -1.49  0.43

diff(window(gw, 1990, c(1992,12)), lag=2)

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1990             -0.57 -0.82  0.78  3.43  3.04  1.61  0.59 -0.77
## 1991 -1.17 -1.07 -0.70 -0.33  0.53  1.59  2.61  2.98  1.18 -0.87
## 1992 -0.74 -1.02 -1.60 -0.64  0.91  1.37  1.53  2.30  1.02 -0.95
##        Nov   Dec
## 1990 -1.18 -1.42
## 1991 -1.90 -1.74
## 1992 -2.23 -1.06

diff(window(gw, 1990, c(1992,12)), lag=12)

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1991  1.49  0.96  1.36  1.45  1.11 -0.39  0.68  0.98  1.27  0.88
## 1992  0.98  0.61  0.08  0.30  0.46  0.08 -0.62 -0.60 -0.78 -0.68
##        Nov   Dec
## 1991  0.55  0.56
## 1992 -1.11  0.00
```

**Q8 :** *What happens to the length of resulting series with different lags?* *Jump to A8* •

**Q9 :** *Do you expect the one-month differences to be the same for each* *month within a year? Are they?* *Jump to A9* •

---

**Q10** : *Do you expect the one-month differences to be the same for the same month interval in different years? Are they?* *Jump to A10* •

Specifying the `differences` argument to `diff` computes higher-order differences, that is, differences of differences (order 2), etc.

---

**Task 10** : Compute the first, second and third order differences of the series from 1990 – 1993. •

```
diff(window(gw, 1990, c(1993,12)), lag=12, differences=1)

##         Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1991  1.49  0.96  1.36  1.45  1.11 -0.39  0.68  0.98  1.27  0.88
## 1992  0.98  0.61  0.08  0.30  0.46  0.08 -0.62 -0.60 -0.78 -0.68
## 1993 -0.29  0.20  0.46  0.47  0.60  1.21  1.62  1.24  1.50  1.62
##         Nov   Dec
## 1991  0.55  0.56
## 1992 -1.11  0.00
## 1993  2.57  1.45

diff(window(gw, 1990, c(1993,12)), lag=12, differences=2)

##         Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1992 -0.51 -0.35 -1.28 -1.15 -0.65  0.47 -1.30 -1.58 -2.05 -1.56
## 1993 -1.27 -0.41  0.38  0.17  0.14  1.13  2.24  1.84  2.28  2.30
##         Nov   Dec
## 1992 -1.66 -0.56
## 1993  3.68  1.45

diff(window(gw, 1990, c(1993,12)), lag=12, differences=3)

##         Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1993 -0.76 -0.06  1.66  1.32  0.79  0.66  3.54  3.42  4.33  3.86
##         Nov   Dec
## 1993  5.34  2.01
```

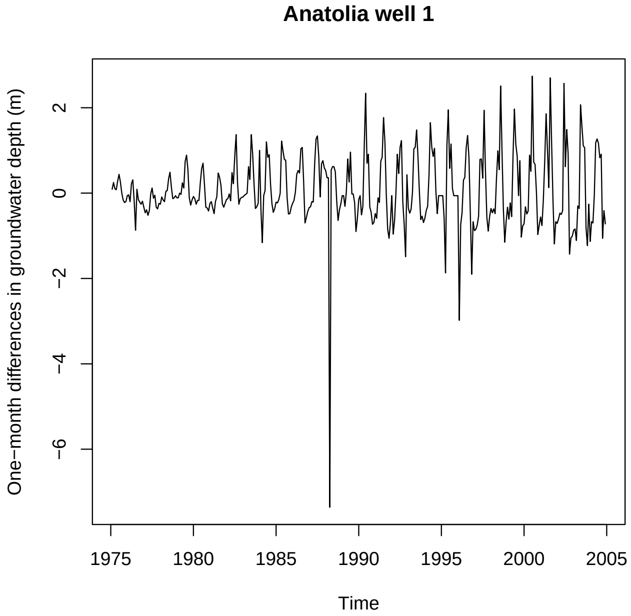
---

**Q11** : *What is the interpretation of these differences?* *Jump to A11* •

---

**Task 11** : Plot the first differences for the whole time series. •

```
plot(diff(gw), ylab="One-month differences in groundwater depth (m)",
     main="Anatolia well 1")
```



This gives us a very different view of the sequence, compared to the time series itself.

**Q12** : *What are the outstanding features of the first differences? Jump to A12* •

**TASK 12** : Identify the months with the largest extractions and recharges. •

The `which` function identifies elements in a vector which meet some criterion. The `which.min` and `which.max` are shorthand for finding the minimum and maximum.

We first find the most extreme extractions and recharge, and then all of these where the level changed by more than 2 m in one month, in either direction:

```
i <- which.min(diff(gw))
diff(gw)[i]

## [1] -7.36

time(gw)[i]

## [1] 1988.167

cycle(gw)[i]

## [1] 3

i <- which.max(diff(gw))
diff(gw)[i]

## [1] 2.74
```

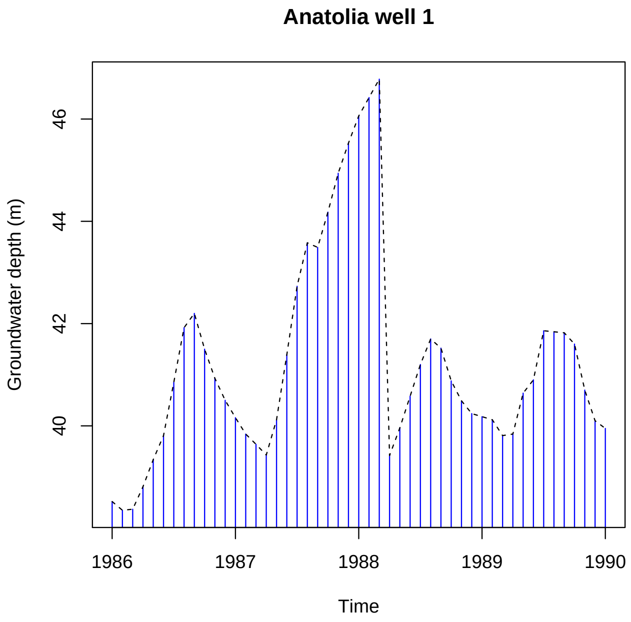
```
time(gw)[i]

## [1] 2000.417

cycle(gw)[i]

## [1] 6

i <- which(abs(diff(gw)) > 2)
diff(gw)[i]

## [1] -7.36  2.34 -2.98  2.51  2.74  2.70  2.57  2.07

time(gw)[i]

## [1] 1988.167 1990.333 1996.000 1998.500 2000.417 2001.500 2002.333
## [8] 2003.333

cycle(gw)[i]

## [1] 3 5 1 7 6 7 5 5
```

Note the use of the `time` function on the series to get the dates corresponding to the selected measurements; the resulting vector of dates is subsetted with the `[]` indexing operator. Similarly, the `cycle` function is used to display the position in the cycle, here the month.

The extreme recharge (about 7.5 m) in March 1988 draws our attention; is this correct, given than no other month in the time series had more than about 2.5 m recharge?

First, we take a closer look at the measurements for this and surrounding years:

```
plot(window(gw, start=1986, end=1990), ylab="Groundwater depth (m)",
     main="Anatolia well 1", type="h", col="blue")
lines(window(gw, start=1986, end=1990), lty=2)
```



**Q13** : *Is the pattern for Spring 1987 – Spring 1988 consistent with the surrounding years?* Jump to A13 •

Here are the actual measurements for the relevant time period:

```
window(gw, start=c(1987,3), end=c(1988,6))

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1987             39.64 39.43 40.12 41.39 42.73 43.58 43.49 44.18
## 1988 46.06 46.42 46.78 39.42 39.96 40.58
##        Nov   Dec
## 1987 44.94 45.53
## 1988
```

**Q14** : *How could this anomaly be explained?* Jump to A14 •

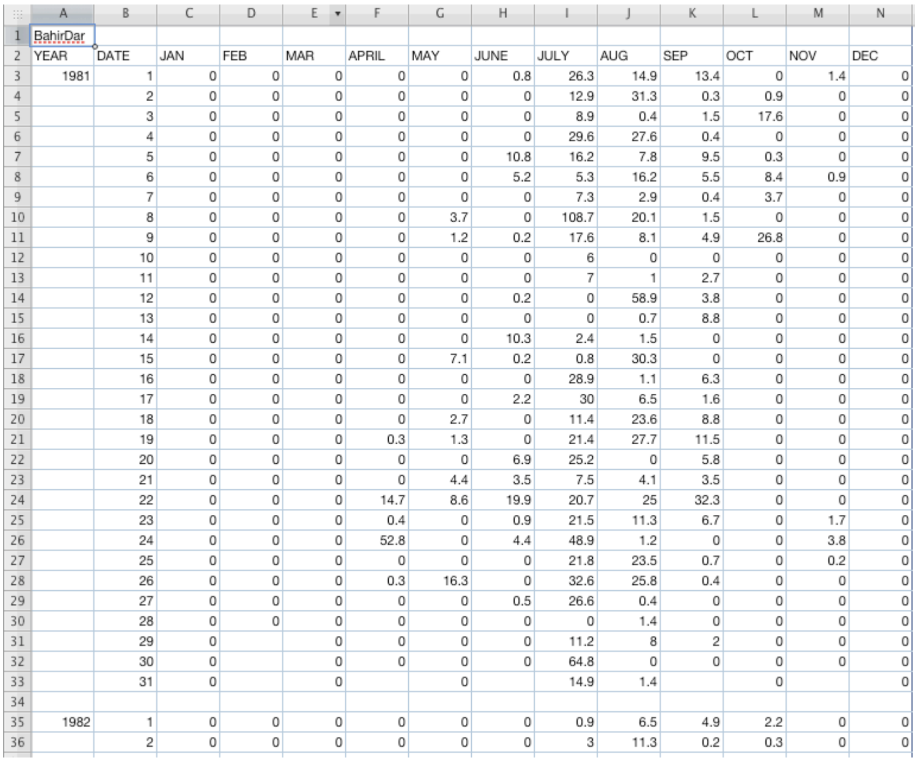
## 2.2 Example 2: Daily rainfall

The second example dataset is a series of daily rainfall records from a station in the Lake Tana basin, Ethiopia, for 26 years (1981 – 2005).

**Q15** : *What could be some research questions for this time series?* Jump to A15 •

This is a typical "messy" Excel file; we must do some substantial manipulations to get it into useful format for R. It is possible to reformat and clean up the dataset in Excel, but much more difficult than the systematic approach we can take in R.

Here is a screenshot of the original file:



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | BahirDar | | | | | | | | | | | | | |
| 2 | YEAR | DATE | JAN | FEB | MAR | APRIL | MAY | JUNE | JULY | AUG | SEP | OCT | NOV | DEC |
| 3 | 1981 | 1 | 0 | 0 | 0 | 0 | 0 | 0.8 | 26.3 | 14.9 | 13.4 | 0 | 1.4 | 0 |
| 4 | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 12.9 | 31.3 | 0.3 | 0.9 | 0 | 0 |
| 5 | | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 8.9 | 0.4 | 1.5 | 17.6 | 0 | 0 |
| 6 | | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 29.6 | 27.6 | 0.4 | 0 | 0 | 0 |
| 7 | | 5 | 0 | 0 | 0 | 0 | 0 | 10.8 | 16.2 | 7.8 | 9.5 | 0.3 | 0 | 0 |
| 8 | | 6 | 0 | 0 | 0 | 0 | 0 | 5.2 | 5.3 | 16.2 | 5.5 | 8.4 | 0.9 | 0 |
| 9 | | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 7.3 | 2.9 | 0.4 | 3.7 | 0 | 0 |
| 10 | | 8 | 0 | 0 | 0 | 0 | 3.7 | 0 | 108.7 | 20.1 | 1.5 | 0 | 0 | 0 |
| 11 | | 9 | 0 | 0 | 0 | 0 | 1.2 | 0.2 | 17.6 | 8.1 | 4.9 | 26.8 | 0 | 0 |
| 12 | | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 |
| 13 | | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 1 | 2.7 | 0 | 0 | 0 |
| 14 | | 12 | 0 | 0 | 0 | 0 | 0 | 0.2 | 0 | 58.9 | 3.8 | 0 | 0 | 0 |
| 15 | | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.7 | 8.8 | 0 | 0 | 0 |
| 16 | | 14 | 0 | 0 | 0 | 0 | 0 | 10.3 | 2.4 | 1.5 | 0 | 0 | 0 | 0 |
| 17 | | 15 | 0 | 0 | 0 | 0 | 7.1 | 0.2 | 0.8 | 30.3 | 0 | 0 | 0 | 0 |
| 18 | | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 28.9 | 1.1 | 6.3 | 0 | 0 | 0 |
| 19 | | 17 | 0 | 0 | 0 | 0 | 0 | 2.2 | 30 | 6.5 | 1.6 | 0 | 0 | 0 |
| 20 | | 18 | 0 | 0 | 0 | 0 | 2.7 | 0 | 11.4 | 23.6 | 8.8 | 0 | 0 | 0 |
| 21 | | 19 | 0 | 0 | 0 | 0.3 | 1.3 | 0 | 21.4 | 27.7 | 11.5 | 0 | 0 | 0 |
| 22 | | 20 | 0 | 0 | 0 | 0 | 0 | 6.9 | 25.2 | 0 | 5.8 | 0 | 0 | 0 |
| 23 | | 21 | 0 | 0 | 0 | 0 | 4.4 | 3.5 | 7.5 | 4.1 | 3.5 | 0 | 0 | 0 |
| 24 | | 22 | 0 | 0 | 0 | 14.7 | 8.6 | 19.9 | 20.7 | 25 | 32.3 | 0 | 0 | 0 |
| 25 | | 23 | 0 | 0 | 0 | 0.4 | 0 | 0.9 | 21.5 | 11.3 | 6.7 | 0 | 1.7 | 0 |
| 26 | | 24 | 0 | 0 | 0 | 52.8 | 0 | 4.4 | 48.9 | 1.2 | 0 | 0 | 3.8 | 0 |
| 27 | | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 21.8 | 23.5 | 0.7 | 0 | 0.2 | 0 |
| 28 | | 26 | 0 | 0 | 0 | 0.3 | 16.3 | 0 | 32.6 | 25.8 | 0.4 | 0 | 0 | 0 |
| 29 | | 27 | 0 | 0 | 0 | 0 | 0 | 0.5 | 26.6 | 0.4 | 0 | 0 | 0 | 0 |
| 30 | | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.4 | 0 | 0 | 0 | 0 |
| 31 | | 29 | 0 | | 0 | 0 | 0 | 0 | 11.2 | 8 | 2 | 0 | 0 | 0 |
| 32 | | 30 | 0 | | 0 | 0 | 0 | 0 | 64.8 | 0 | 0 | 0 | 0 | 0 |
| 33 | | 31 | 0 | | 0 | | 0 | | 14.9 | 1.4 | | 0 | | 0 |
| 34 | | | | | | | | | | | | | | |
| 35 | 1982 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.9 | 6.5 | 4.9 | 2.2 | 0 | 0 |
| 36 | | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 11.3 | 0.2 | 0.3 | 0 | 0 |

This is a nice format to view the data but not to analyze as a time series.

First, to quote the *R Data Import/Export* manual [17, §8]:

> "The most common R data import/export question seems to be: 'how do I read an Excel spreadsheet?' ... The first piece of advice is to avoid doing so if possible!"

The recommended procedure is to export each sheet of the Excel file to a separate **comma-separated value** ("CSV") file, using Excel or another program that can read Excel files and write CSV files.

We have done this[6]; the CSV file is `Tana_Daily.csv`.

> **Note:** However, recently several packages have been written which work well to exchange Excel files and R data structures, for example the `readxl` package, which has an `excel_sheets` function to list the sheets in an Excel workbook, and a `read_excel` function to read an Excel sheet as a data frame.

---

**Task 13** : View the contents of the CSV file. •

Again we use the `file.show` function:

```
file.show("./ds_tsa/Tana_Daily.csv")
```

Here are the first and last lines of the file:

```
BahirDar,,,,,,,,,,,,,
```

[6] using the Numbers application on Mac OS X 10.6

```
YEAR,DATE,JAN,FEB,MAR,APRIL,MAY,JUNE,JULY,AUG,SEP,OCT,NOV,DEC
1981,1,0,0,0,0,0,0,0.8,26.3,14.9,13.4,0,1.4,0
,2,0,0,0,0,0,0,12.9,31.3,0.3,0.9,0,0
...
,28,0.0,0.0,0.0,0.0,28.0,33.2,28.1,0.9,36.0,0.5,0,
,29,0.0,,0.0,0.0,0.5,24.0,18.5,2.8,6.8,32.7,0,
,30,0.0,,0.0,0.0,0.0,3.6,32.1,11.5,1.0,3,0,
,31,0.0,,0.0,,15.0,,85.7,0.6,,0,,
```

---

**Task 14** : Read the CSV file into an R object and examine its structure. •

We use the very flexible `read.csv` function, which is a version of the more general `read.table` method. These have quite some useful optional arguments (see `?read.table` for details); here we use:

- `skip=1` to skip over the first line, which is just the station name;
- `header=T` to specify that the first line read (after skipping) contains the variable names, here the months;
- `colClasses` to specify the data type of each field (column); this is because we want to treat all the rainfall amounts as character strings to fix the "trace amount" entries specified with the string `TR`;
- `na.strings="N.A"` to specify that any string that is identically `N.A` is a missing value; note blanks are also considered missing.
- `blank.lines.skip=T` to skip blank lines.

Note that optional arguments `sep` and `dec` can be used if the separator between fields is not the default "`,`" or the decimal point is not the default "`.`".

```
tana <- read.csv("./ds_tsa/Tana_Daily.csv", skip=1, header=T,
                 colClasses=c(rep("integer",2), rep("character",12)),
                 blank.lines.skip=T,na.strings=c("N.A","NA"," "))
str(tana)

## 'data.frame': 831 obs. of  14 variables:
##  $ YEAR : int  1981 NA NA NA NA NA NA NA NA NA ...
##  $ DATE : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ JAN  : chr  "0" "0" "0" "0" ...
##  $ FEB  : chr  "0" "0" "0" "0" ...
##  $ MAR  : chr  "0" "0" "0" "0" ...
##  $ APRIL: chr  "0" "0" "0" "0" ...
##  $ MAY  : chr  "0" "0" "0" "0" ...
##  $ JUNE : chr  "0.8" "0" "0" "0" ...
##  $ JULY : chr  "26.3" "12.9" "8.9" "29.6" ...
##  $ AUG  : chr  "14.9" "31.3" "0.4" "27.6" ...
##  $ SEP  : chr  "13.4" "0.3" "1.5" "0.4" ...
##  $ OCT  : chr  "0" "0.9" "17.6" "0" ...
##  $ NOV  : chr  "1.4" "0" "0" "0" ...
##  $ DEC  : chr  "0" "0" "0" "0" ...

tana[1:35,]

##    YEAR DATE JAN FEB MAR APRIL  MAY JUNE  JULY  AUG  SEP  OCT NOV DEC
## 1  1981    1   0   0   0     0    0  0.8  26.3 14.9 13.4    0 1.4   0
## 2    NA    2   0   0   0     0    0    0  12.9 31.3  0.3  0.9   0   0
## 3    NA    3   0   0   0     0    0    0   8.9  0.4  1.5 17.6   0   0
## 4    NA    4   0   0   0     0    0    0  29.6 27.6  0.4    0   0   0
## 5    NA    5   0   0   0     0    0 10.8  16.2  7.8  9.5  0.3   0   0
## 6    NA    6   0   0   0     0    0  5.2   5.3 16.2  5.5  8.4 0.9   0
## 7    NA    7   0   0   0     0    0    0   7.3  2.9  0.4  3.7   0   0
## 8    NA    8   0   0   0     0  3.7    0 108.7 20.1  1.5    0   0   0
## 9    NA    9   0   0   0     0  1.2  0.2  17.6  8.1  4.9 26.8   0   0
## 10   NA   10   0   0   0     0    0    0     6    0    0    0   0   0
## 11   NA   11   0   0   0     0    0    0     7    1  2.7    0   0   0
## 12   NA   12   0   0   0     0    0  0.2     0 58.9  3.8    0   0   0
## 13   NA   13   0   0   0     0    0    0     0  0.7  8.8    0   0   0
## 14   NA   14   0   0   0     0    0 10.3   2.4  1.5    0    0   0   0
## 15   NA   15   0   0   0     0  7.1  0.2   0.8 30.3    0    0   0   0
## 16   NA   16   0   0   0     0    0    0  28.9  1.1  6.3    0   0   0
## 17   NA   17   0   0   0     0    0  2.2    30  6.5  1.6    0   0   0
## 18   NA   18   0   0   0     0  2.7    0  11.4 23.6  8.8    0   0   0
## 19   NA   19   0   0   0   0.3  1.3    0  21.4 27.7 11.5    0   0   0
## 20   NA   20   0   0   0     0    0  6.9  25.2    0  5.8    0   0   0
## 21   NA   21   0   0   0     0  4.4  3.5   7.5  4.1  3.5    0   0   0
## 22   NA   22   0   0   0  14.7  8.6 19.9  20.7   25 32.3    0   0   0
## 23   NA   23   0   0   0   0.4    0  0.9  21.5 11.3  6.7    0 1.7   0
## 24   NA   24   0   0   0  52.8    0  4.4  48.9  1.2    0    0 3.8   0
## 25   NA   25   0   0   0     0    0    0  21.8 23.5  0.7    0 0.2   0
## 26   NA   26   0   0   0   0.3 16.3    0  32.6 25.8  0.4    0   0   0
## 27   NA   27   0   0   0     0    0  0.5  26.6  0.4    0    0   0   0
## 28   NA   28   0   0   0     0    0    0     0  1.4    0    0   0   0
## 29   NA   29   0       0     0    0    0  11.2    8    2    0   0   0
## 30   NA   30   0       0     0    0    0  64.8    0    0    0   0   0
## 31   NA   31   0       0          0       14.9  1.4         0       0
## 32   NA   NA
## 33 1982    1   0   0   0     0    0    0   0.9  6.5  4.9  2.2   0   0
## 34   NA    2   0   0   0     0    0    0     3 11.3  0.2  0.3   0   0
## 35   NA    3   0   0  30     0    0  0.3  11.8 24.6  5.7 <NA>   0   0
```

We can see that each year has 31 lines followed by one lines with only two `NA` missing values. Only the first line of each year has the year number in the first field.

The `read.csv` function creates a so-called **dataframe**, which is a matrix with named fields (columns).

---

**Q16** : *What is in each field of the dataframe?* *Jump to A16* •

There are many missing values, which can be identified with `is.na` function; the `sum` function then shows how many in total. We also compute the number of blank records; the `na.rm` argument is needed to remove missing values before summing:

```
sum(is.na(tana[,3:14]))

## [1] 170

sum(tana[,3:14]=="",na.rm=TRUE)

## [1] 457
```

Note that the null values are *not* missing, they are for days that do not exist (e.g. 30-February).

We can see all the values with the `unique` function, and sort these for easy viewing with `sort`. To get all the months as one vector we stack columns 3 through 14 with the `stack` function:

```
head(sort(unique(stack(tana[,3:14])$values)))

## [1] ""      "0"     "0.0"   "0.01" "0.1"   "0.2"

tail(sort(unique(stack(tana[,3:14])$values)))

## [1] "9.8"   "9.9"   "90.5" "95.5" "tr"    "TR"
```

---

**Q17** : *What are the meanings of the* `""` *(empty string),* `0`, `0.0`, *and* `"TR"` *(also written* `"tr"`*) values?* *Jump to A17* •

Let's see how many of each of the problematic values there are, and how many zeroes:

```
sum(tana[,3:14]=="TR", na.rm=TRUE)

## [1] 69

sum(tana[,3:14]=="tr", na.rm=TRUE)

## [1] 2

sum(tana[,3:14]=="0.01", na.rm=TRUE)

## [1] 14

sum(tana[,3:14]=="0", na.rm=TRUE)

## [1] 4920
```

The trace values are conventionally set to half the measurement precision, or one order of magnitude smaller, or (since they have very little effect on rainfall totals) to zero.

---

**Q18** : *What is the measurement precision?* *Jump to A18* •

---

**Task 15** : Set the trace values and any measurements below `0.1` to zero. •

For this we use the very useful `recode` function in the `car` "Companion to Applied Regression" package from John Fox [9]. The `require` function loads the library if it is not already loaded.

```
require(car)

## Loading required package:  car

## Loading required package:  carData

for (i in 3:14) {
tana[,i] <- recode(tana[,i], "c('TR','tr','0.01')='0'")
}
```

```
head(sort(unique(stack(tana[,3:14])$values)),12)

##  [1] ""     "0"    "0.0" "0.1" "0.2" "0.3" "0.4" "0.5" "0.6" "0.7" "0.8"
## [12] "0.9"
```

```
tail(sort(unique(stack(tana[,3:14])$values)),12)

##  [1] "9.0"  "9.1"  "9.2"  "9.3"  "9.4"  "9.5"  "9.6"  "9.7"  "9.8"
## [10] "9.9"  "90.5" "95.5"
```

The problematic values have been replaced by zeroes, as we can verify by repeated the summary of “problematic” values:

```
sum(tana[,3:14]=="TR", na.rm=TRUE)

## [1] 0

sum(tana[,3:14]=="tr", na.rm=TRUE)

## [1] 0

sum(tana[,3:14]=="0.01", na.rm=TRUE)

## [1] 0

sum(tana[,3:14]=="0", na.rm=TRUE)

## [1] 5005
```

**Task 16** : Organize the daily values as one long vector of values, as required for time series analysis. •

This is complicated by the varying month length, so it is not possible simply to stack the 12 vectors. Also, the sequence must be by year; but each month’s vector contains all the years. Finally, February had 29 days in 1984, 1988, . . . 2004, so these years had 366, not 365 days. To detect seasonality we need equal-length years. Fortunately in this case, February is a dry month, and there was no rain on this date in any of the leap years:

```
tana[tana$DATE==29,"FEB"]

##  [1] ""    NA    ""    NA    ""    NA    "0"   NA    ""    NA    ""
## [12] NA    ""    NA    "0"   NA    ""    NA    ""    NA    ""    NA
## [23] "0.0" NA    ""    NA    NA    NA    NA    NA    "0"   NA    NA
## [34] NA    NA    NA    NA    NA    "0"   NA    ""    NA    ""    NA
## [45] ""    NA    "0.0" NA    ""    NA    ""
```

So a simple if somewhat inelegant solution is to ignore records for 29-February[7]

We use the c “catentate” function to stack vectors, specifying their length by our knowledge of the length of each month. Note the zero-length “months” which will match with the first two columns of the dataframe (fields YEAR and DATE).

```
tana.ppt <- NULL;
month.days <- c(0,0,31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
for (yr.first.row in seq(from=1, by=32, length=(2006 - 1981 + 1))) {
 for (month.col in 3:14) {
tana.ppt <-
    c(tana.ppt, tana[yr.first.row:(yr.first.row + month.days[month.col]-1),
                             month.col])
 }
```

[7] Apologies to anyone born on this date!

```
};
str(tana.ppt)

##  chr [1:9490] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" ...

rm(month.days, yr.first.row, month.col)
```

Check that this is an integral number of years:

```
length(tana.ppt)/365

## [1] 26
```

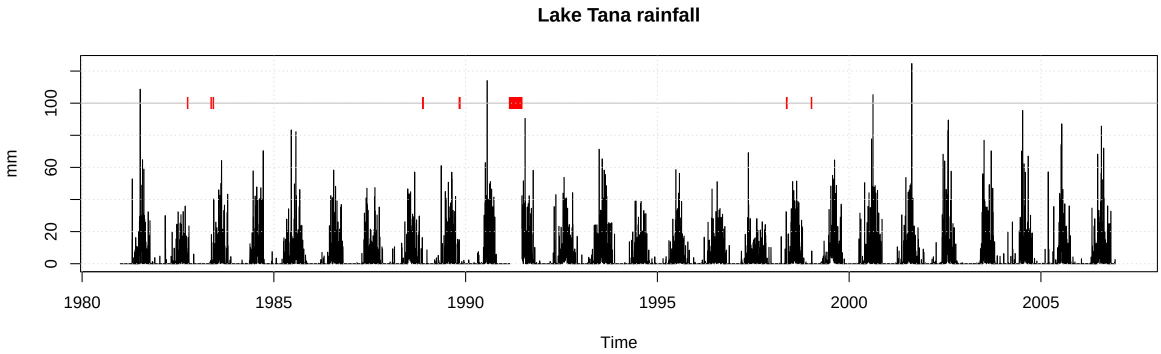
**Task 17** : Convert this to a time series with the appropriate metadata. •

Again, the `ts` function is used to convert the series; the `frequency` argument specifies a cycle of 365 days and the `start` argument specifies the beginning of the series (first day of 1981):

```
tana.ppt <- ts(tana.ppt, start=1981, frequency=365)
str(tana.ppt)

##  Time-Series [1:9490] from 1981 to 2007: 0 0 0 0 ...
```

We enhance the plot by highlighting the missing values, showing them as red vertical bars near the top of the plot:

```
plot(tana.ppt, main="Lake Tana rainfall", ylab="mm")
abline(h=100, col="gray")
points(xy.coords(x=time(tana.ppt), y=100, recycle=T),
       pch=ifelse(is.na(tana.ppt),"l",""), col="red")
grid()
```



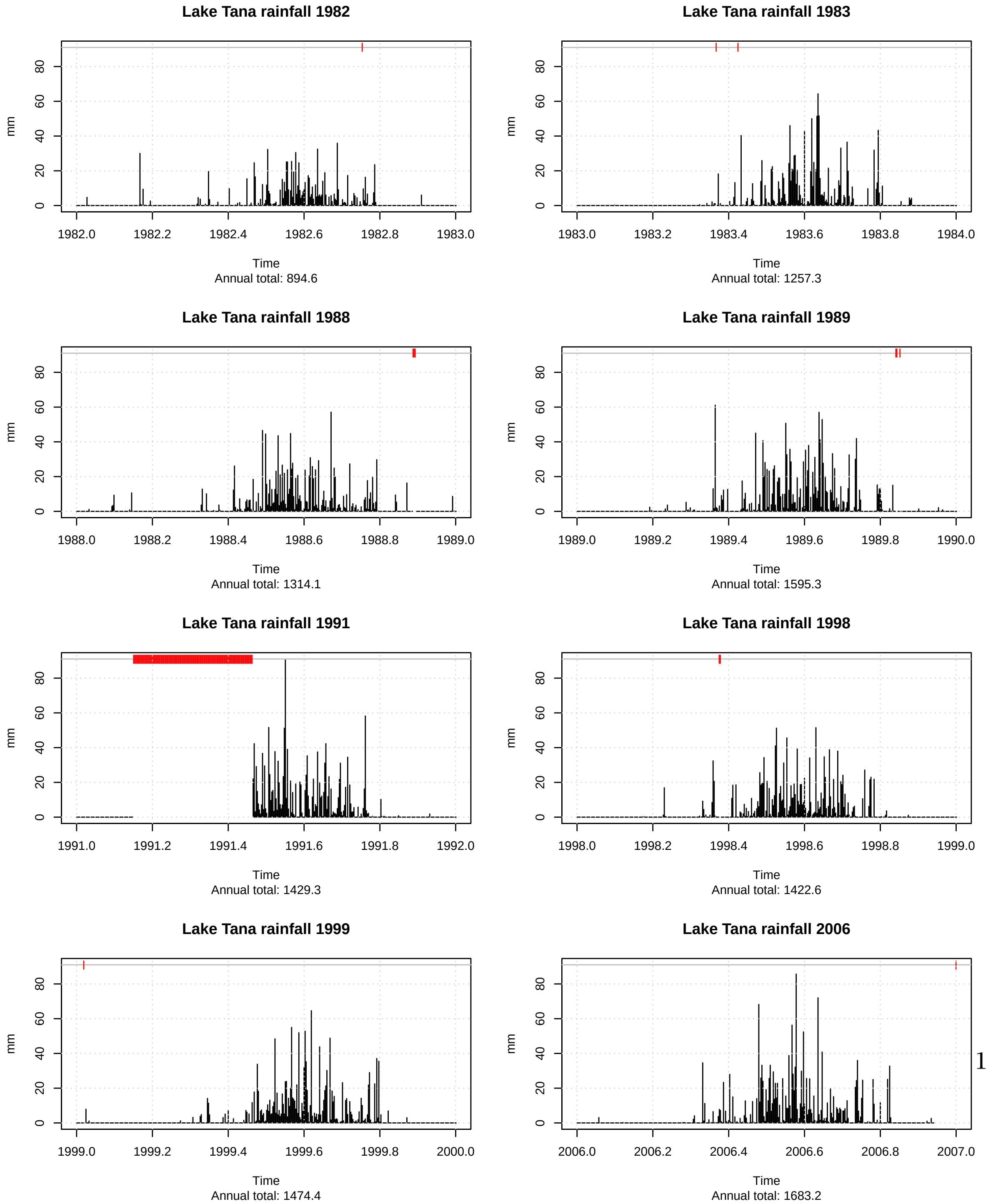
There are six years with a few missing observations, and a long series of missing observations in 1991.

To zoom in on the within-year structure, we display one-year windows for the years with missing values, with these highlighted, and the most recent year. The `points` function is used to place points on top of a bar graph created with the `plot` function (with the optional `type="h"` argument). To compare several years side-by-side, we compute the maximum daily rainfall with the `max` function, and use it to set a common limit with the optional `ylim` argument to `plot`.

> **Note:** A small complication is that `ylim` requires numeric arguments, but `max` returns a character value, even on a numeric time series. This must be converted to a number with `as.numeric` before use in `ylim`. This is also the case for the `sum` function used in the graph subtitle.
>
> Also, the optional `extend` to the `window` function, allows the time series to be extended by the `start` and `end` arguments.

```
yrs <- c(1982, 1983, 1988, 1989,1991,1998,1999, 2006); ymax <- 0
for (i in yrs) {
    ymax <- as.numeric(max(ymax, window(tana.ppt,
                                        start=i, end=i+1, extend=TRUE),
                           na.rm=T))
}
(ymax <- ceiling(ymax))

## [1] 91
```

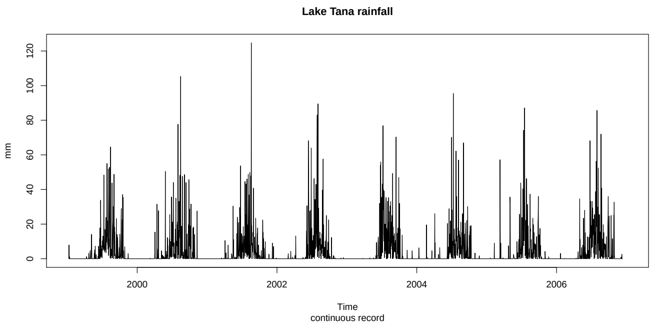
```
par(mfrow=c(4,2))
for (i in yrs) {
plot(window(tana.ppt, start=i, end=i+1, extend=TRUE),
     type="h", ylab="mm", ylim=c(0,ymax));
title(main=paste("Lake Tana rainfall", i),
      sub=paste("Annual total:",
                sum(as.numeric(window(tana.ppt, start=i, end=i+1)),
                    na.rm=T)))
abline(h=ymax, col="gray")
points(xy.coords(x=time(window(tana.ppt, start=i, end=i+1, extend=TRUE)),
                 y=ymax, recycle=T),
       pch=ifelse(is.na(window(tana.ppt, start=i, end=i+1, extend=TRUE)),"l",""),
       col="red")
grid()
}

## Warning in window.default(x, ...):  'end' value not changed

par(mfrow=c(1,1))
```



**Q19** : *Does there seem to be a seasonal difference in rainfall? Is it consistent year-to-year?* *Jump to A19* •

This series has missing values; for some analyses complete series are needed. The `na.contiguous` function finds the longest contiguous sequence of values:

```
str(tana.ppt)

##  Time-Series [1:9490] from 1981 to 2007: 0 0 0 0 ...

sum(is.na(tana.ppt))

## [1] 127

tana.ppt.c <- na.contiguous(tana.ppt)
str(tana.ppt.c)

##  Time-Series [1:2912] from 1999 to 2007: 0 7.9 0 0 ...
##  - attr(*, "na.action")= 'omit' int [1:6578] 1 2 3 4 5 6 7 8 9 10 ...

frequency(tana.ppt.c)

## [1] 365

head(time(tana.ppt.c))

## [1] 1999.022 1999.025 1999.027 1999.030 1999.033 1999.036

head(cycle(tana.ppt.c))

## [1]  9 10 11 12 13 14

tail(time(tana.ppt.c))

## [1] 2006.984 2006.986 2006.989 2006.992 2006.995 2006.997

tail(cycle(tana.ppt.c))

## [1] 360 361 362 363 364 365

sum(is.na(tana.ppt.c))

## [1] 0

plot(tana.ppt.c, main="Lake Tana rainfall", ylab="mm", sub="continuous record")
```



---

**Q20** : *What is the extent of the contiguous time series?* *Jump to A20* •

Gap filling is discussed in §7.

## 2.3 Answers

---

A1 : *Among the questions might be:*

- *Is the the groundwater level increasing or decreasing over time?*
- *Is any trend consistent over the measurement period?*
- *We expect an annual cycle corresponding to the rainy and dry seasons; is this in fact observed? What is the lag of this cycle compared to the rainfall cycle?*
- *Is the amplitude of the cycle constant over the measurement period or does it change year-to-year?*
- *Are the trends and cycles the same for both wells?*
- *A practical issue: if measurements are missing from one of the time series, can it be "reliably" estimated from (1) other observations in the same series; (2) observations in the other series; (3) some combination of these?*

*Return to Q1* •

---

A2 : *No, it is just a list of numbers. We have to know from the metadata what it represents.* *Return to Q2* •

---

A3 : *The measurements are a 360-element vector.* *Return to Q3* •

---

A4 : `print` *shows the actual value of each measurements; here the depth to groundwater.* `cycle` *shows the position of each measurement in its cycle; here this is the month of the year.* `time` *gives the fractional year.* *Return to Q4* •

---

A5 : `frequency` *is the number of measurements within each cycle; here there are 12 months in one year.* `deltat` *is the interval between successive measurements in terms of a single cycle, so here* `deltat` *is fractional years:* $1/12 = 0.8\bar{3}$. *Return to Q5* •

---

A6 :

1. *Generally increasing trend over time since about 1983 and lasting till 2002; that is, depth is increasing so the groundwater is further from the surface;*
2. *But, significant recharge 1975 – 1978;*
3. *Clear yearly cycle, but this is not seen in 1975–1978;*
4. *Severe draw-down in 1988, with rapid recharge.*

*Return to Q6* •

---

A*7* : *It is annual; the groundwater is closest to the surface in April and then decreases rapidly until August; recharge begins in September.* *Return to Q7* •

---

A*8* : *The series gets shorter; the differences given are for the end date of each lag, so that, for example, the one-year lag can only begin in January 1991 (difference with January 1990).* *Return to Q8* •

---

A*9* : *Differences between months should reflect the annual cycle: negative in the fall and winter (recharge) and positive in the spring summer (extraction). This in fact is observed.* *Return to Q9* •

---

A*10* : *If there is no variability in the recharge or extraction the monthly differences (for example, March to April) should be the same in each year. This is not observed; for example the recharge in 1991 vs. 1990 (January through May) was much higher than the difference from 1992 vs. 1991.* *Return to Q10* •

---

A*11* : *The **first differences** are the year-to-year differences in the same month. For example, from January 1990 to January 1991 the groundwater depth increased by*

*, whereas the difference in the next year (January 1991 to January 1992) was*

*The **second differences** are the change in difference from one year to the next; for example the difference from January 1991 to January 1992, compared to the difference in the previous year (January 1990 to January 1991) is -0.51 . In this case the second year's differences were less than the first, i.e., the groundwater depth didn't increase as much in the second year (January to January), compared to the first.*

*The **third differences** are the change in two-year differences.* *Return to Q11* •

---

A*12* :

1. *The annual cycle is clear: month-to-month differences are positive (increasing groundwater depth, i.e., extraction) in the spring and summer (extraction) and negative (decreasing groundwater depth, i.e., recharge) in the fall and winter;*
2. *The amplitude of the annual cycle increased until about 1999 and seems to be stable since then;*
3. *There is one extreme extraction event, four times as much groundwater lowering than any other.*

*Return to Q12* •

---

A*13* : *No. Not only is the magnitude of extraction and recharge more, the* *recharge does not begin until March 1988, whereas in the other years it begins in August. Note the small recharge in September 1988, which seems to be the beginning of the recharge (fall and winter rains); but then this is interrupted by six months of extraction during the (wet) winter.* Return to Q13 •

---

**A14** : *To explain this there would have to have been continued extraction in the winter of 1987–1988, perhaps because of failure of winter rains; we do not have the rainfall data to see if this is possible. But also March 1988 would have to had extreme rainfall.*

*Another possibility is that the records for winter 1987–1988 were incorrectly entered into the data file, or perhaps measured in a different way, with this being corrected in March 1988. However, they are consistent month-to-month in this period, so it is unlikely to be a data entry error.* Return to Q14 •

---

**A15** :

- *Is there an annual cycle in rainfall amount?*
- *If so, how many "rainy seasons" are there?*
- *How variable is the rainfall year-to-year?*
- *Is there a trend over time in the rainfall amount, either overall or in a specific season?*
- *How consistent is rainfall day-to-day within the dry and rainy seasons? In other words, how long is the auto-correlation period?*
- *What are probable values for missing measurements, and what is the uncertainty of these predictions?*

Return to Q15 •

---

**A16** : `YEAR` *is the year of measurement; however it is only entered for the first day of each year (upper-left of the sheet).*

`DATE` *is the day of the month (all months), from* `1 .. 31`

`JAN` ...`DEC` *are the rainfall amounts for the given day of the named month.* Return to Q16 •

---

**A17** : `""` *(empty string) means there is no data at all, i.e., no day in the month (here, 29 – 31 February);* `0` *means no rainfall, as does* `0.0`*, and* `"TR"` *means trace rainfall.* Return to Q17 •

---

**A18** : *Only one decimal place is given; also the minimum measurement for February was* `0.1`*, so the precision is* `0.1`*. January does have a* `0.01` *value, which seems to be an inconsistent attempt to record the trace amount.* Return to Q18 •

---

**A19** : *There is a definite seasonality: rains from about May to about August*

*and the other months dry. Rains can begin early (1993) or there can be some sporadic rains before the true start of the rainy season (1992, less in 1990).*
*Return to Q19* •

---

A20 : *From the 9th day of 1999 through the end of 2006, i.e., almost eight years.*
*Return to Q20* •

# 3 Analysis of a single time series

## 3.1 Summaries

Summary descriptive statistics and graphics show the overall behaviour of a time series.

### 3.1.1 Numerical summaries

---

**Task 18** : Summarize the groundwater levels of the first well, for the whole series. •

The `summary` function gives the overall distribution:

```
summary(gw)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   29.90   34.90   41.63   41.58   46.80   57.68
```

More useful is a summary grouped by some attribute of the time series, typically cycle or position in the cycle.

---

**Task 19** : Summarize the groundwater levels of the first well, for each year separately. •

The time-series data structure (one long vector with attributes) is not suitable for grouping by year or position in cycle. We create a data frame, one column being the time series and the other two factors giving the year and cycle. Recall the `time` function returns the time of observation (here, as fractional years); these are converted to year number with the `floor` function. Similarly, the `cycle` function returns the position in the cycle, from `1 ...frequency(series)`. Both of these are converted from time-series to numbers with the `as.numeric` function. We also keep the fractional time axis, as field `time`.

```
gw.f <- data.frame(gw, year=as.numeric(floor(time(gw))),
                   cycle=as.numeric(cycle(gw)), time=time(gw))
str(gw.f)

## 'data.frame': 360 obs. of  4 variables:
##  $ gw   : Time-Series  from 1975 to 2005: 34.4 34.5 34.7 34.8 34.9 ...
##  $ year : num  1975 1975 1975 1975 1975 ...
##  $ cycle: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ time : Time-Series  from 1975 to 2005: 1975 1975 1975 1975 1975 ...
```

Now the year can be used as a grouping factor; the `summary` function (specified with the `FUN` argument) is applied to each year with the `by` function, with the `IND` "index" argument being the grouping factor:

```
head(by(gw.f$gw, IND=gw.f$year, FUN=summary))

## $`1975`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34.36   34.77   35.22   35.18   35.62   35.86
##
## $`1976`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   33.73   34.34   34.76   34.72   35.20   35.51
##
## $`1977`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   30.61   31.47   31.61   31.81   32.16   33.40
##
## $`1978`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   29.90   30.07   30.34   30.41   30.75   30.97
##
## $`1979`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   30.42   30.53   31.15   31.50   32.45   32.96
##
## $`1980`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   31.60   31.91   32.28   32.37   32.74   33.33
```

Note that the function applied could be `max` (to get the deepest level), `min` (the shallowest), `median`, `quantile` etc., in fact anything that summarizes a numeric vector.

A single year can be extracted with the `[[]]` list extraction operator:

```
by(gw.f$gw, IND=gw.f$year, FUN=summary)[["1978"]]

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   29.90   30.07   30.34   30.41   30.75   30.97

by(gw.f$gw, IND=gw.f$year, FUN=max)[["1978"]]

## [1] 30.97

by(gw.f$gw, IND=gw.f$year, FUN=min)[["1978"]]

## [1] 29.9

by(gw.f$gw, IND=gw.f$year, FUN=quantile, probs=.9)[["1978"]]

## [1] 30.839
```

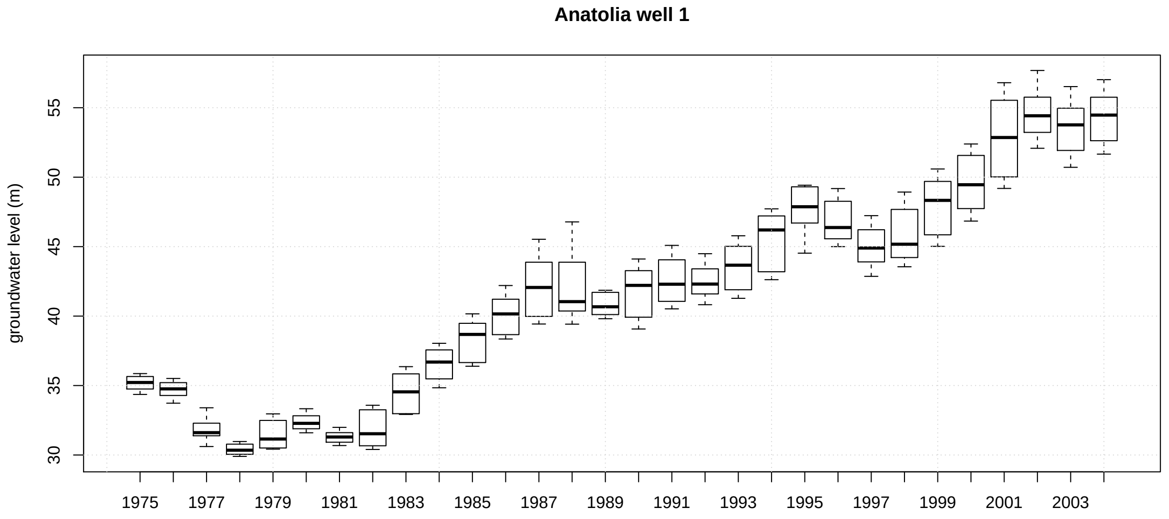
### 3.1.2 Graphical summaries

Boxplots, often grouped by cycle number or position in the cycle, reveal the average behaviour of time series.

**Task 20 :** Display boxplots of the Anatolia well levels, grouped by year. •

This is produced by the `boxplot` function, with the **formula** operator ~ showing the dependence of the left-hand side of the formula (here, groundwater level) on the right-hand side (here, year):

```
boxplot(gw.f$gw  ~ gw.f$year, main="Anatolia well 1",
        ylab="groundwater level (m)")
grid()
```



This gives a clear impression of the per-year variability.

**Q21** : *Is there a trend in the per-year amplitude of the annual cycle of groundwater levels?* *Jump to A21* •

**Task 21** : Display boxplots of the Anatolia well levels, grouped by position in the cycle (month), after correcting for the overall trend. •

We add a field to the data frame that is the difference of each observation from its annual mean, which is computed with the `aggregate` function (see below, §3.2.1 for details of this function):

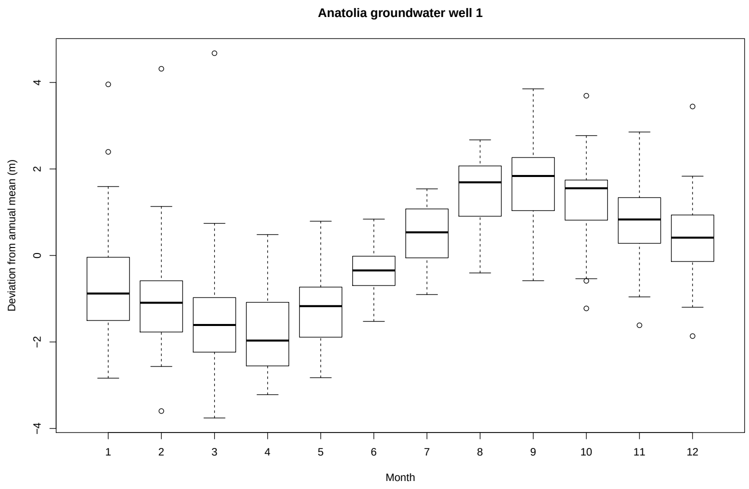
```
(ann.mean <- aggregate(gw, nfrequency=1, FUN=mean))

## Time Series:
## Start = 1975
## End = 2004
## Frequency = 1
##  [1] 35.17750 34.71833 31.80750 30.41083 31.50333 32.36917 31.28667
##  [8] 31.89667 34.50333 36.51417 38.27750 40.09167 42.08583 42.10417
## [15] 40.78417 41.66000 42.56833 42.46167 43.51583 45.49667 47.74917
## [22] 46.78500 44.99417 45.89000 47.85417 49.62000 52.94750 54.59083
## [29] 53.53667 54.21333

time(ann.mean)

## Time Series:
## Start = 1975
## End = 2004
## Frequency = 1
##  [1] 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
## [14] 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
## [27] 2001 2002 2003 2004
```

We subtract the correct annual mean from each observation, using the `match` function to find the position in the vector of annual means that corresponds to the year of the observation:

```
gw.f$in.yr <- as.numeric(gw - ann.mean[match(gw.f$year, time(ann.mean))])
str(gw.f)

## 'data.frame': 360 obs. of  5 variables:
##  $ gw   : Time-Series  from 1975 to 2005: 34.4 34.5 34.7 34.8 34.9 ...
##  $ year : num  1975 1975 1975 1975 1975 ...
##  $ cycle: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ time : Time-Series  from 1975 to 2005: 1975 1975 1975 1975 1975 ...
##  $ in.yr: num  -0.818 -0.727 -0.477 -0.378 -0.297 ...
```

Now we can display the grouped boxplot:

```
boxplot(gw.f$in.yr ~ gw.f$cycle, xlab="Month",
        ylab="Deviation from annual mean (m)",
        main="Anatolia groundwater well 1")
```



**Q22** : *Is there an annual cycle? Are all the months equally variable? Are there exceptions to the pattern?* *Jump to A22* •

## 3.2 Smoothing

Often we are interested in the behaviour of a time series with short-term variations removed; this is called **smoothing** or **low-pass filtering**. A major use of filtering is to more easily detect an overall trend, or deviations from the overall trend that are of longer duration than one time-series interval.

### 3.2.1 Aggregation

Sometimes we want to view the series at coarser intervals than presented. The `aggregate` function changes the frequency of the series. This has arguments:

1. `nfrequency` (by default 1), the new number of observations per unit of time; this must be a divisor of the original frequency. For example, a quarterly summary of an annual time series would specify `nfrequency=4`.

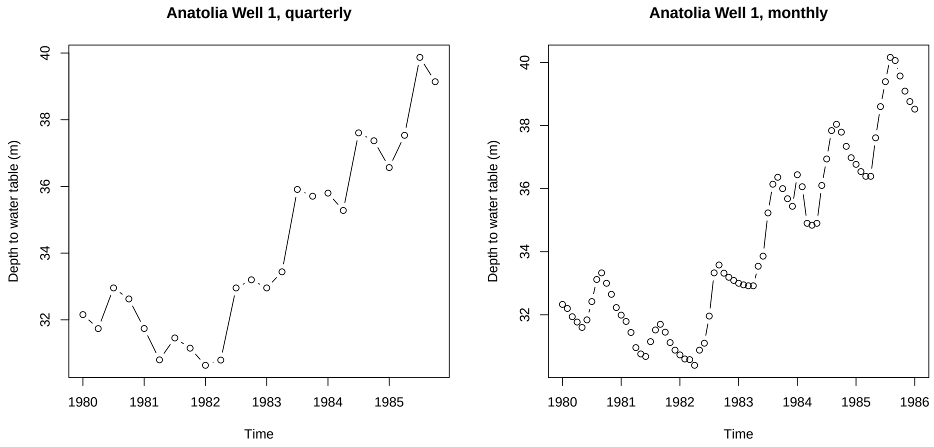
2. `FUN` (by default `sum`), the function to apply when aggregating.

---

**Task 22 :** Convert the monthly time series of well levels from 1980 through 1985 into a quarterly series of mean well levels for the quarter. •

The aggregation function here must could be `mean`; the default `sum` has no physical meaning. Other reasonable choices, depending on objective, would be `max`, `min`, or `median`.

```
gw.q <- aggregate(window(gw, 1980, 1986), nfrequency=4, FUN=mean)
str(gw.q)

##  Time-Series [1:24] from 1980 to 1986: 32.2 31.7 33 32.6 31.7 ...

par(mfrow=c(1,2))
plot(gw.q, ylab="Depth to water table (m)",
     main="Anatolia Well 1, quarterly", type="b")
plot(window(gw, 1980, 1986), ylab="Depth to water table (m)",
     main="Anatolia Well 1, monthly", type="b")
par(mfrow=c(1,1))
```



---

**Q23 :** *What are the differences between the monthly and quarterly series? How much resolution is lost?* *Jump to A23*

•

### 3.2.2 Smoothing by filtering

A simple way to smooth the series is to apply a **linear filter**, using the `filter` function. By default this uses a moving average of values on both sides (before and after) each measurement. The method requires a filter, which is a vector of coefficients to provide the weights, in reverse time order.

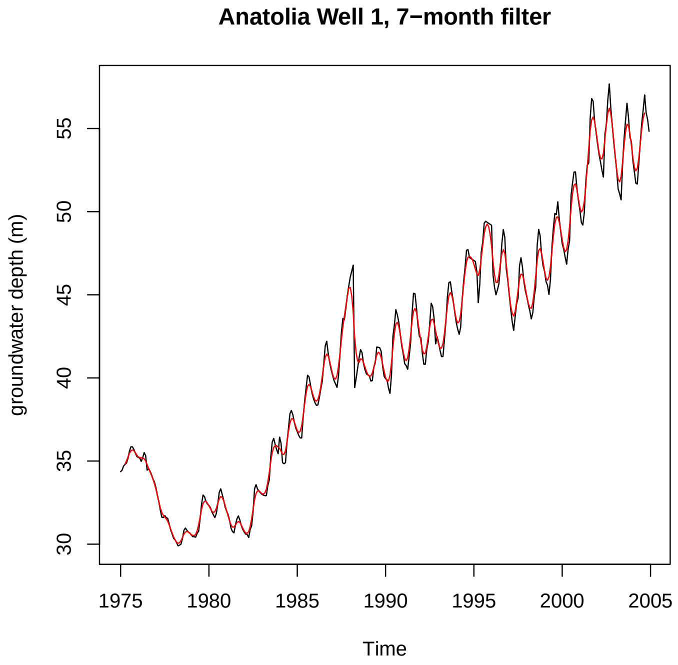
The moving average for a given measurement is a weighted sum of $2p+1$ measurements (the preceding $p$, the 1 original measurement, and the following $p$ measurements); $2p+1$ is called the **order** of the filter. Note that filtering shortens the time series, because it can not be computed at the $p$ first and last measurements:

$$s_t = \sum_{j=-p}^{p} w_j y_{t+j} \; ; \; t = p+1 \ldots n-p \qquad (1)$$

The weights $w_j$ must sum to 1; generally $w_{-j} = w_{+j}$ (symmetry) but this is not required.

**Task 23 :** Filter the series with a symmetric seven-month filter that gives full weight to the measurement month, three-quarters weight to adjacent months, half weight to months two removed, and quarter weight to months three removed. •

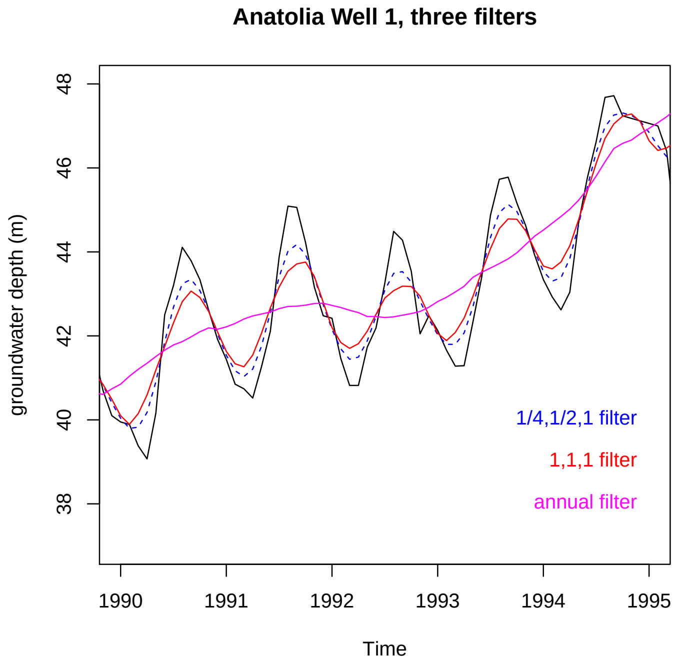
```
k <- c(.25,.5,.75,1,.75,.5,.25)
(k <- k/sum(k))

## [1] 0.0625 0.1250 0.1875 0.2500 0.1875 0.1250 0.0625

fgw <- filter(gw, sides=2, k)
plot.ts(gw, main="Anatolia Well 1, 7-month filter",
        ylab="groundwater depth (m)")
lines(fgw, col="red")
```



**Q24 :** *What is the effect of this filter?* *Jump to A24* •

**Task 24 :** Repeat the seven-month filter with equal weights for each month; also compute an annual filter (12 months equally weighted); plot the three filtered series together for the period 1990–1995. •

```
fgw.2 <- filter(gw, sides=2, rep(1,7)/7)
fgw.3 <- filter(gw, sides=2, rep(1,12)/12)
plot.ts(gw, xlim=c(1990,1995), ylim=c(37,48),
        ylab="groundwater depth (m)")
title(main="Anatolia Well 1, three filters")
lines(fgw, col="blue", lty=2)
lines(fgw.2, col="red")
lines(fgw.3, col="magenta")
text(1995,40,"1/4,1/2,1 filter", col="blue", pos=2)
text(1995,39,"1,1,1 filter", col="red",pos=2)
text(1995,38,"annual filter", col="magenta",pos=2)
```

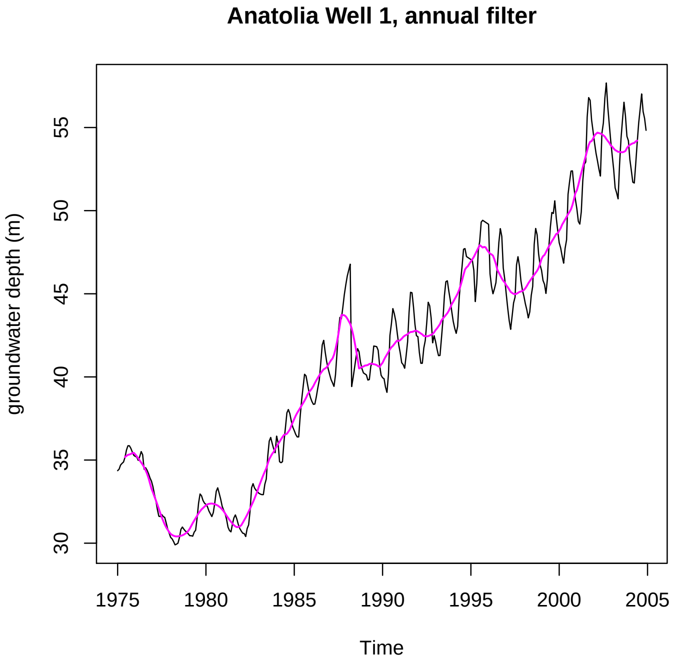


**Q25 :** *What is the effect of giving equal weights to the measurements in the filter?* *Jump to A25* •

**Q26 :** *What is the effect of the annual filter?* *Jump to A26* •

**Task 25 :** Plot the annual filter for the complete series. •

```
plot.ts(gw, main="Anatolia Well 1, annual filter",
        ylab="groundwater depth (m)")
lines(fgw.3, col="magenta", lwd=1.5)
```

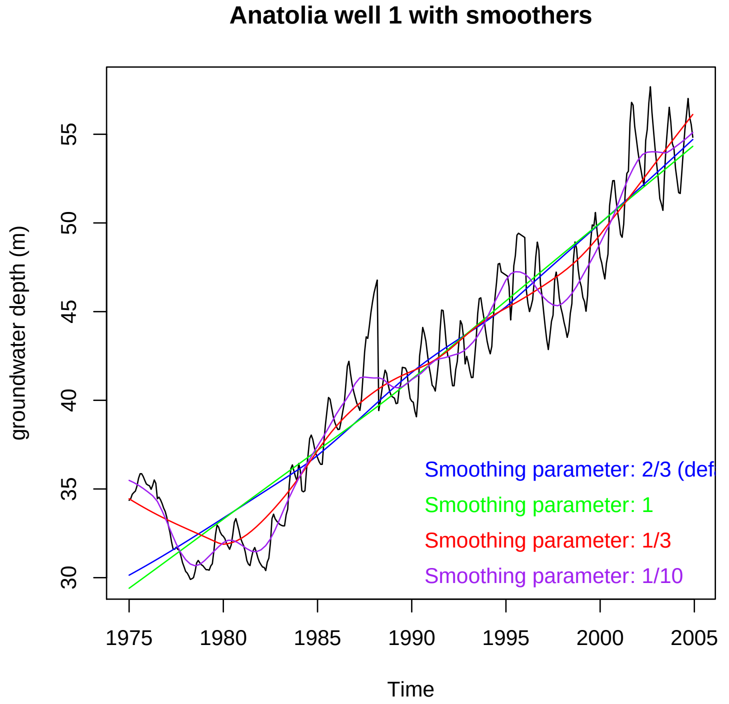


### 3.2.3 Smoothing by local polynomial regression

Another way to visualize the trend is to use the `lowess` "Local Polynomial Regression Fitting" method [6], which fits the data points locally, using nearby (in time) points. These are weighted by their distance (in time) from the point to be smoothed; the degree of smoothing is controlled by the size of the neighbourhood. This results in a smooth curve.

**Task 26** : Display the time series and its smoothed series for the default smoothing parameter (2/3), and three other values of the parameter, one smoother, one finer, and one very fine (little smoothing). •

```
plot.ts(gw, main="Anatolia well 1 with smoothers",
        ylab="groundwater depth (m)")
lines(lowess(gw), col="blue")
lines(lowess(gw, f=1), col="green")
lines(lowess(gw, f=1/3), col="red")
lines(lowess(gw, f=1/10), col="purple")
text(1990, 36, "Smoothing parameter: 2/3 (default)", col="blue", pos=4)
text(1990, 34, "Smoothing parameter: 1", col="green", pos=4)
text(1990, 32, "Smoothing parameter: 1/3", col="red", pos=4)
text(1990, 30, "Smoothing parameter: 1/10", col="purple", pos=4)
```



---

**Q27 :** *What is the effect of the smoothing parameter? Which seems to give the most useful summary of this series?* *Jump to A27* •

## 3.3 Decomposition

Many time series can be decomposed into three parts:

1. A **trend**;
2. A **cycle** after accounting for the trend;
3. A **residual**, also known as **noise**, after accounting for any trend and cycle.

These correspond to three processes:

1. A long-term process that operates over the time spanned by the series;
2. A cyclic process that operates within each cycle;
3. A local process which causes variability between cycles.

Each of these processes is of interest and should be explained by the analyst.

---

**Task 27 :** Decompose the groundwater level time series. •

The workhorse function for decomposition is `stl` "Seasonal Decomposition of Time Series by Loess", i.e., using a similar smooth trend removal as the `lowess` function used above in §3.2.3. This has one required argument, `s.window`, which is the (odd) number of lags for the loess window for seasonal extraction; for series that are already defined to be cyclic (as here), this can be specified as `s.window="periodic"`, in which case the cycle is known from the attributes of the time series, extracted here with the `frequency` function:

```
frequency(gw)

## [1] 12
```

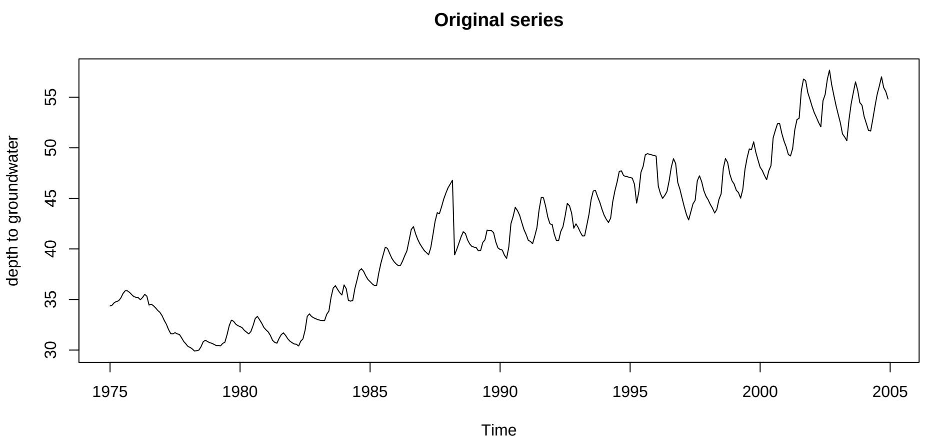
Then, the mean of the cycle at each position is taken:

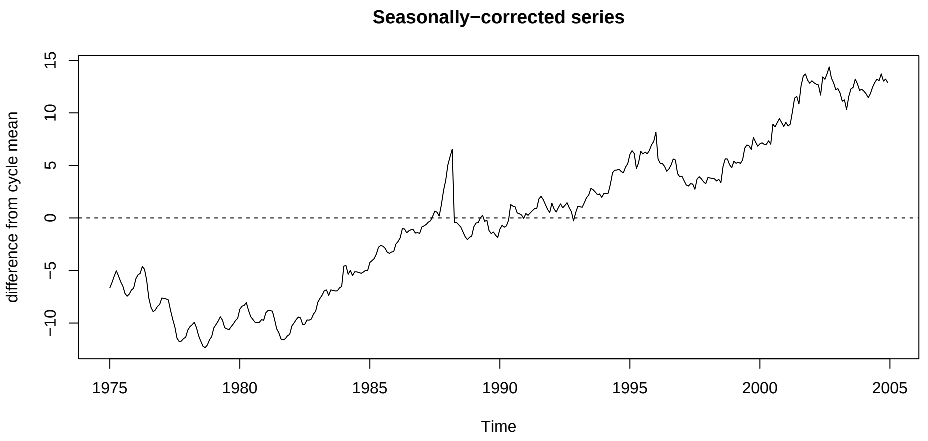
```
tapply(gw, cycle(gw), mean)

##        1        2        3        4        5        6        7
## 41.01700 40.59800 40.25900 39.82733 40.38933 41.22433 42.07233
##        8        9       10       11       12
## 43.03767 43.30633 42.93767 42.33233 41.96433
```

This is subtracted from each value, leaving just the non-seasonal component. Here we show two years' adjustments numerically, and the whole series' adjustment graphically:

```
head(gw, 2*frequency(gw))

##  [1] 34.36 34.45 34.70 34.80 34.88 35.16 35.60 35.86 35.86 35.70 35.48
## [12] 35.28 35.22 35.18 34.98 35.20 35.51 35.32 34.45 34.54 34.39 34.18
## [23] 33.92 33.73

head(gw-rep(tapply(gw, cycle(gw), mean),
            length(gw)/frequency(gw)), 2*frequency(gw))

##         1         2         3         4         5         6         7
## -6.657000 -6.148000 -5.559000 -5.027333 -5.509333 -6.064333 -6.472333
##         8         9        10        11        12         1         2
## -7.177667 -7.446333 -7.237667 -6.852333 -6.684333 -5.797000 -5.418000
##         3         4         5         6         7         8         9
## -5.279000 -4.627333 -4.879333 -5.904333 -7.622333 -8.497667 -8.916333
##        10        11        12
## -8.757667 -8.412333 -8.234333
```

```
par(mfrow=c(2,1))
plot(gw, ylab="depth to groundwater", main="Original series")
plot(gw-rep(tapply(gw, cycle(gw), mean), length(gw)/frequency(gw)),
     ylab="difference from cycle mean", main="Seasonally-corrected series")
abline(h=0, lty=2)
par(mfrow=c(2,1))
```


Original series


Seasonally−corrected series

---

**Q28** : *What has changed in the numeric and graphical view of the time series, after adjustment for cycle means?* *Jump to A28* •

This series, without the seasonal component, is then smoothed as follows.

The `stl` function has another required argument, but with a default. This is `t.window`, which is the span (in lags,not absolute time) of the loess window for trend extraction; this must also be odd. For the periodic series (`s.window="periodic"`), the default is 1.5 times the cycle, rounded to the next odd integer, so here $12 * 1.5 + 1 = 19$, as is proven by the following example:

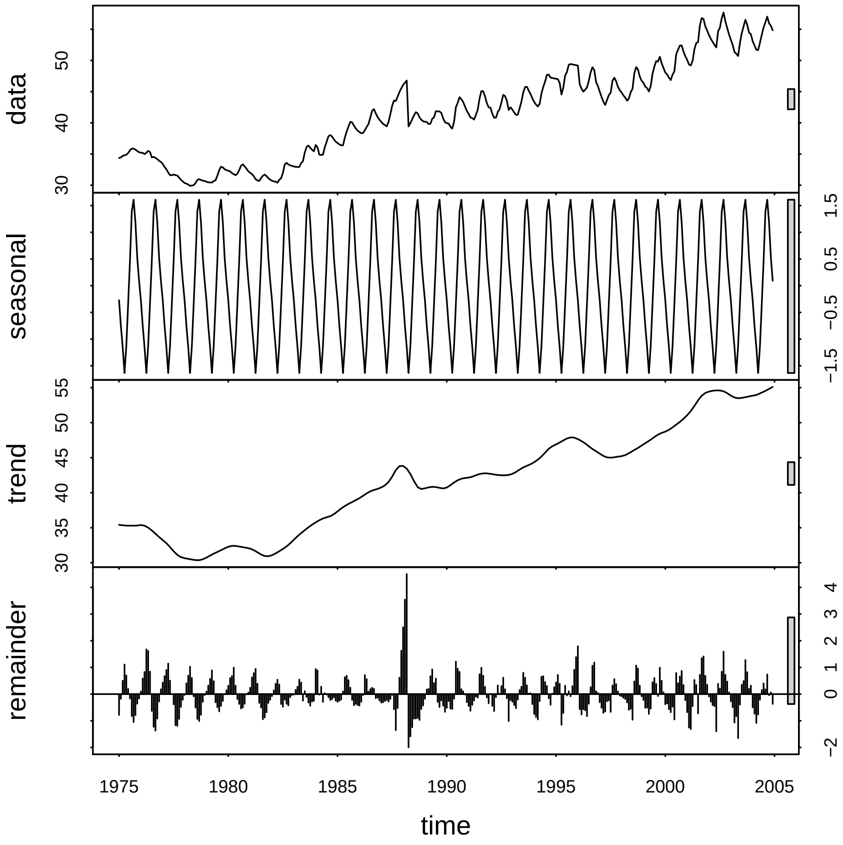
```
gw.stl <- stl(gw, s.window="periodic")
str(gw.stl)

## List of 8
##  $ time.series: Time-Series [1:360, 1:3] from 1975 to 2005: -0.271 -0.75 -1.149 -1.635 -1.
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:3] "seasonal" "trend" "remainder"
##  $ weights    : num [1:360] 1 1 1 1 1 1 1 1 1 1 ...
##  $ call       : language stl(x = gw, s.window = "periodic")
##  $ win        : Named num [1:3] 3601 19 13
##   ..- attr(*, "names")= chr [1:3] "s" "t" "l"
##  $ deg        : Named int [1:3] 0 1 1
##   ..- attr(*, "names")= chr [1:3] "s" "t" "l"
##  $ jump       : Named num [1:3] 361 2 2
##   ..- attr(*, "names")= chr [1:3] "s" "t" "l"
##  $ inner      : int 2
##  $ outer      : int 0
##  - attr(*, "class")= chr "stl"

tmp <- stl(gw, s.window="periodic", t.window=19)
unique(tmp$time.series[,"trend"]-gw.stl$time.series[,"trend"])

## [1] 0

rm(tmp)
```

**Q29** : *What is the structure of the decomposed series object?* *Jump to A29* •

**Task 28** : Display the decomposed series as a graph •

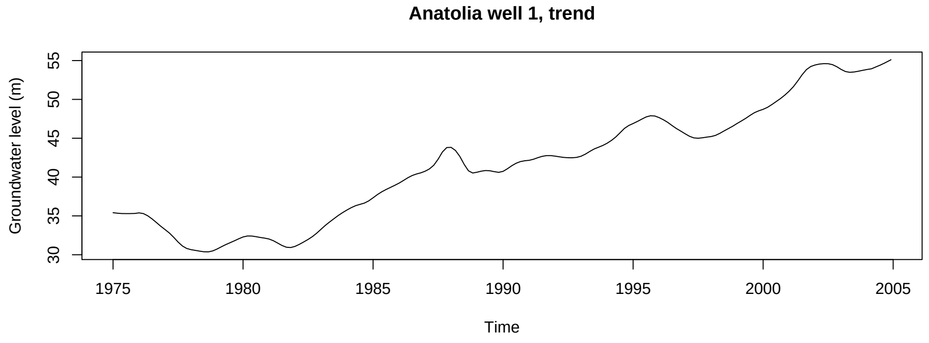
The `plot` function specialises to `plot.stl`, which shows the original series and its decomposition on a single graph.

```
plot(gw.stl)
```



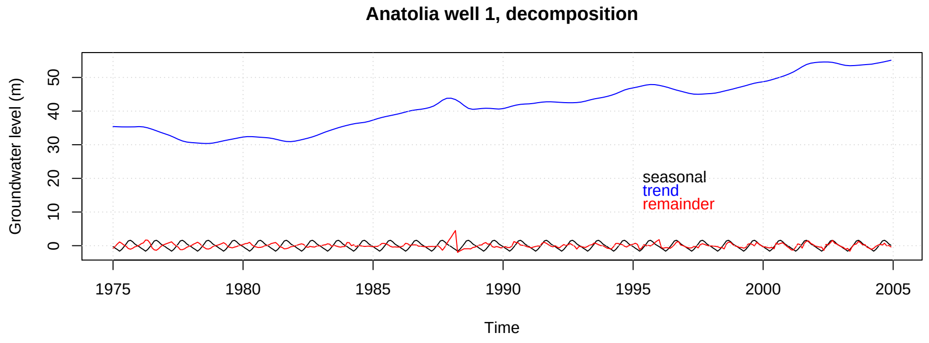
The components of the decomposed series can be extracted; for example to see just the trend:

```
plot(gw.stl$time.series[,"trend"],
     main="Anatolia well 1, trend",
     ylab="Groundwater level (m)")
```



Another way to see the decomposition is with the `ts.plot` function; this shows several time series (here, the components) on the same scale of a single graph, thus visualizing the relative contribution of each component:

```
ts.plot(gw.stl$time.series, col=c("black","blue","red"),
        main="Anatolia well 1, decomposition",
        ylab="Groundwater level (m)")
tmp <-  attributes(gw.stl$time.series)$dimnames[[2]]
for (i in 1:3) {
    text(1995, 24-(i*4), tmp[i], col=c("black","blue","red")[i], pos=4)
    }
grid()
```

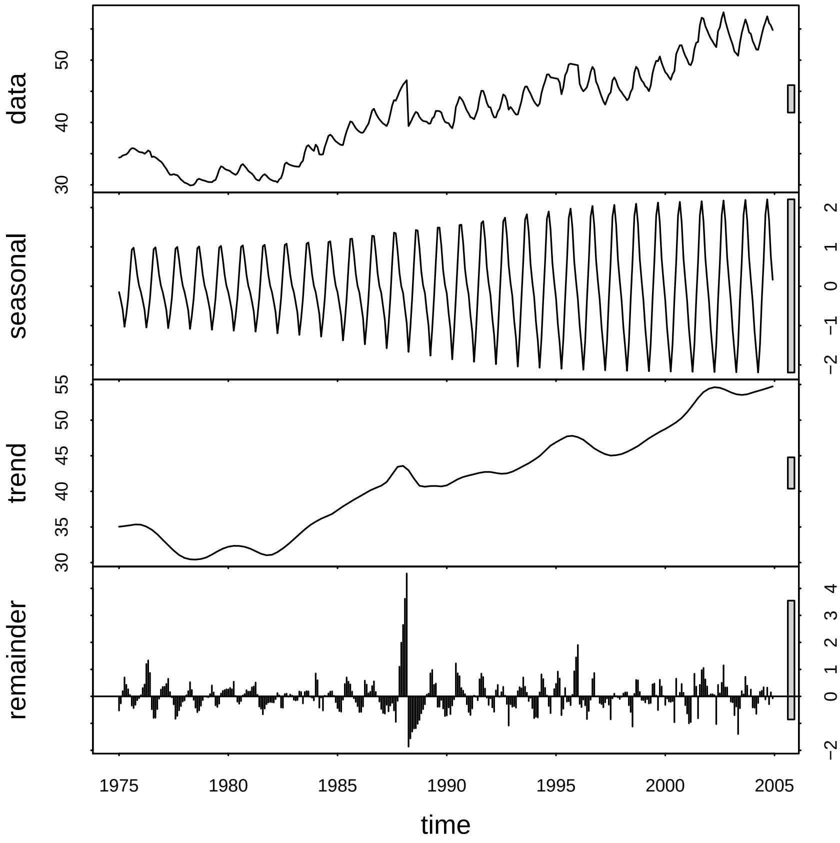


**Q30** : *What are the magnitudes of the three components? Which is contributing most to the observed changes over the time series? Jump to A30* •

The decomposition with `s.window="periodic"` determines the seasonal component with the average of each point in the cycle (e.g., average all Januarys). This can not account for changes in cycle amplitude with time, as is observed here. For this, `s.window` must be set to an odd number near to the cycle or some multiple of it (to average a few years).

**Task 29** : Decompose the groundwater time series, with a two-year window for the seasonal component. •

```
gw.stl <- stl(gw, s.window=2*frequency(gw)+1)
plot(gw.stl)
```



**Q31** : *How does this decomposition differ from the pure periodic decomposition?* *Jump to A31*
•

The smoothness of the lowess fit is controlled with the `t.window` argument; by default this is:

```
nextodd(ceiling((1.5*period) / (1-(1.5/s.window))))
```

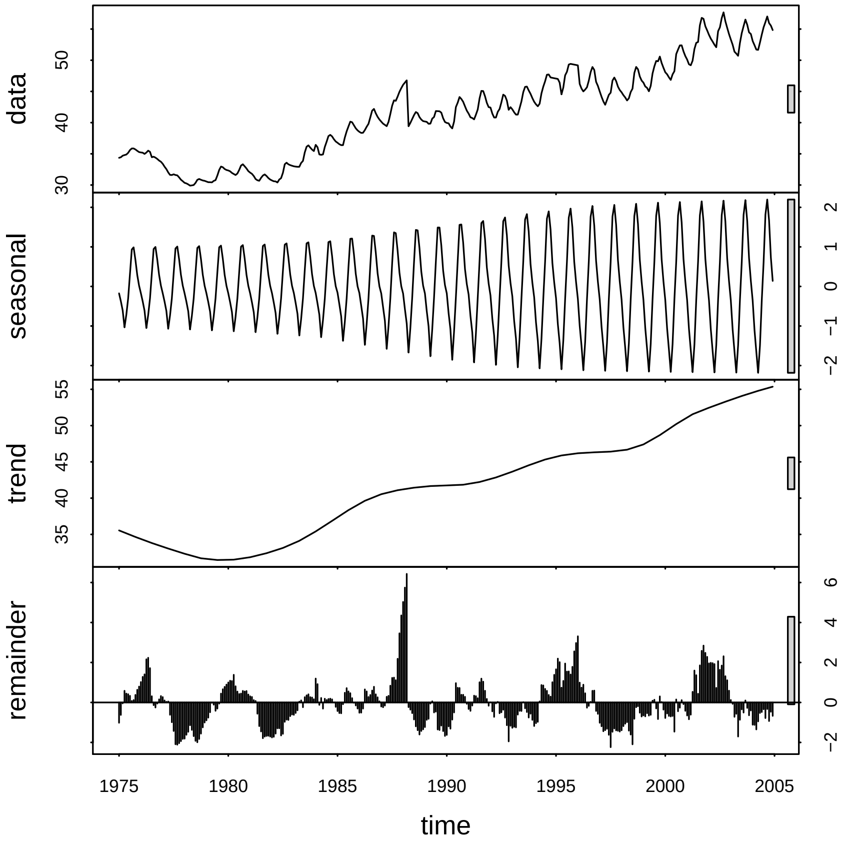
so that for a 25-month seasonal window on a 12-month cycle of this example the trend window is the next higher odd number of $\lceil (1.5 * 12)/(1 - (1.5/25)) \rceil = 20$, i.e., 21.

> **Note:** This is for the case when the period is explicitly given. If the window is specified as `s.window="periodic"` the smoothness parameter is one more than 1.5 times the cycle length, see 3.3.

For a smoother trend this should be increased, for a finer trend decreased. The smoothness of the trend depends on the analyst's knowledge of the process. The previous decomposition has a very rough trend, let's see how it looks with a smoother trend.

**Task 30 :** Recompute and plot the decomposition of the time series with a smoother trend than the default. •

```
gw.stl <- stl(gw, s.window=25, t.window=85)
plot(gw.stl)
```



**Q32 :** *What is the difference between this smooth and the default decompositions? Which best represents the underlying process?* *Jump to A32* •

## 3.4 Serial autocorrelation

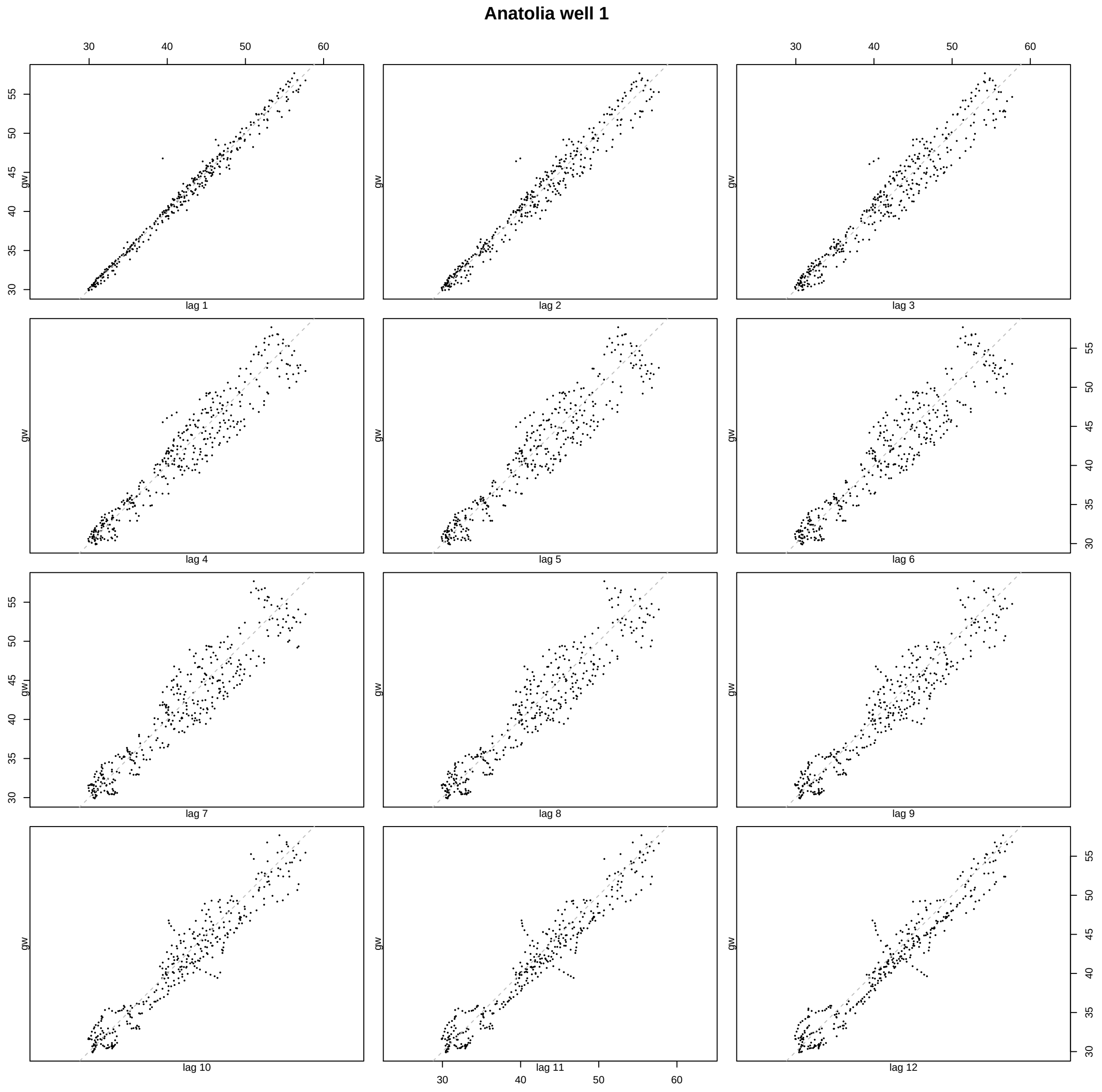
In almost all time series, successive measurements are not independent; rather, they are correlated. Since there is only one variable, this is called

**auto**correlation and is an example of the **second-order** summary[8]

The first way to visualize this is to produce **scatterplots** of measurements compared to others with various **lags**, i.e., time periods **before** the given measurement. The `lag.plot` function produces this.

**Task 31** : Display the auto-correlation scatter plots of groundwater levels for twelve lags (i.e., up to one year of differences). •

```
lag.plot(gw, lags=12, main="Anatolia well 1", cex=0.3, pch=20, lty=1)
```



Note that **positive** values of the `lags` argument refer to lags **before** an

[8] The **first-order** summary is the expected value, either constant mean or trend.

observation. This is used also in the `lag` function, which produces a lagged series with the same indices as the original series, i.e., the series is not shifted.

```
window(gw, 2000, 2001)

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 2000 48.07 47.75 47.26 46.84 47.73 48.24 50.98 51.71 52.38 52.39
## 2001 50.11
##        Nov   Dec
## 2000 51.42 50.67
## 2001

lag(window(gw, 2000, 2001), 1)

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1999
## 2000 47.75 47.26 46.84 47.73 48.24 50.98 51.71 52.38 52.39 51.42
##        Nov   Dec
## 1999       48.07
## 2000 50.67 50.11

lag(window(gw, 2000, 2001), 2)

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1999
## 2000 47.26 46.84 47.73 48.24 50.98 51.71 52.38 52.39 51.42 50.67
##        Nov   Dec
## 1999 48.07 47.75
## 2000 50.11

lag(window(gw, 2000, 2001), -1)

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 2000       48.07 47.75 47.26 46.84 47.73 48.24 50.98 51.71 52.38
## 2001 50.67 50.11
##        Nov   Dec
## 2000 52.39 51.42
## 2001

lag(window(gw, 2000, 2001), -2)

##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 2000             48.07 47.75 47.26 46.84 47.73 48.24 50.98 51.71
## 2001 51.42 50.67 50.11
##        Nov   Dec
## 2000 52.38 52.39
## 2001
```

---

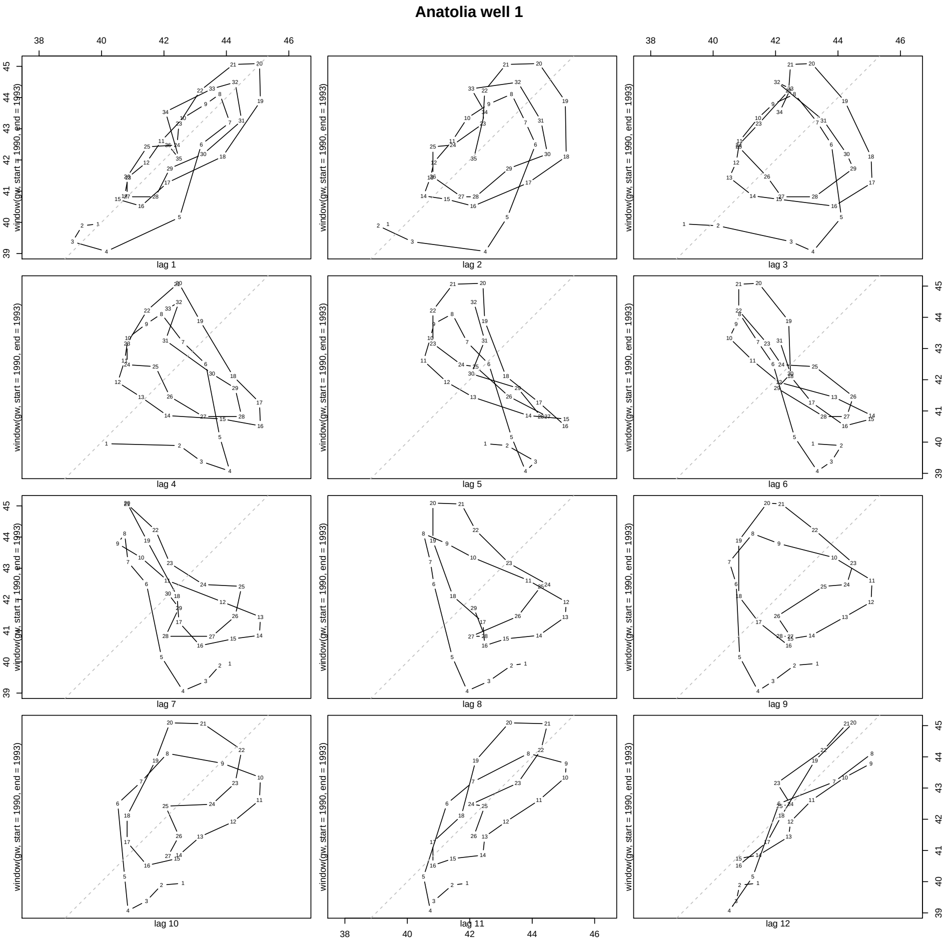
**Q33** : *Describe the evolution of auto-correlation as the lags increase.* *Jump to A33* •

By default if the time series is longer than 150 (as here), individual measurements are not labelled nor joined by lines. For shorter series they are.

---

**Task 32** : Display the auto-correlation scatter plots of groundwater levels for twelve lags (i.e., up to one year of differences) for 1990 through 1992. •

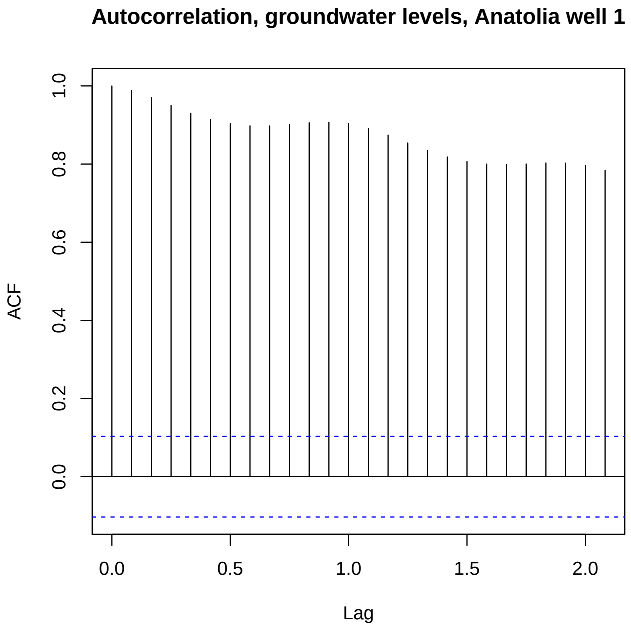
```
lag.plot(window(gw, start=1990, end=1993), lags=12, main="Anatolia well 1")
```



**Q34** : *What additional information is provided by the labelled points and joining lines?* *Jump to A34* •

Autocorrelation ranges from $-1$ (perfect negative correlation) through 0 (no correlation) through $+1$ (perfect positive correlation). It is computed by `acf` "Autocorrelation" function. By default `acf` produces a graph showing the correlation at each lag; to see the actual values the result must be printed. The default number of lags to compute for a single series is $\lceil 10 \cdot \log 10 n \rceil$, where $n$ is the lenth of the series; for the groundwater example this is 26.

```
print(acf(gw, plot=F))

##
## Autocorrelations of series 'gw', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
##  1.000  0.988  0.970  0.950  0.930  0.914  0.903  0.898  0.898  0.901
## 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
##  0.906  0.907  0.903  0.891  0.874  0.854  0.834  0.818  0.807  0.800
## 1.6667 1.7500 1.8333 1.9167 2.0000 2.0833
##  0.799  0.800  0.803  0.803  0.797  0.784

acf(gw, main="Autocorrelation, groundwater levels, Anatolia well 1")
```

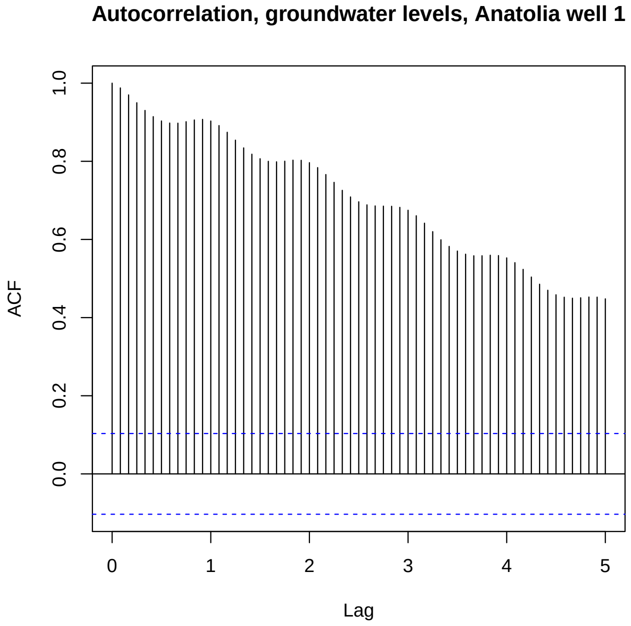


**Q35 :** *Are successive observations correlated? Positively or negatively? How strong is the correlation? How does this change as the lag between observations increases?* Jump to A35 •

Clearly the autocorrelation holds for longer lags.

**Task 33 :** Display the autocorrelation of groundwater levels for five years. •

```
acf(gw, lag.max=60, main="Autocorrelation, groundwater levels, Anatolia well 1")
```



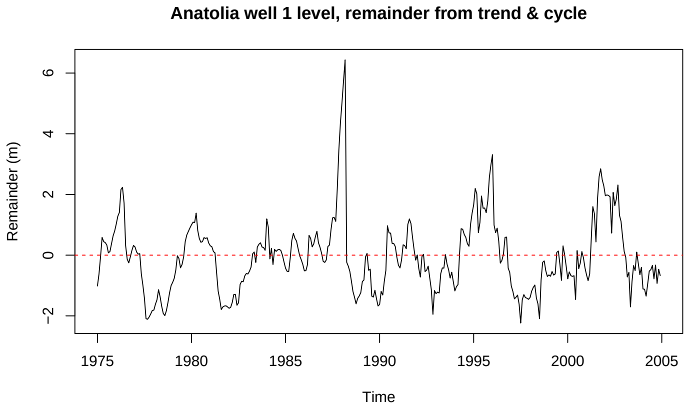
Some of the autocorrelation can be explained by the trend and seasonal components.

---

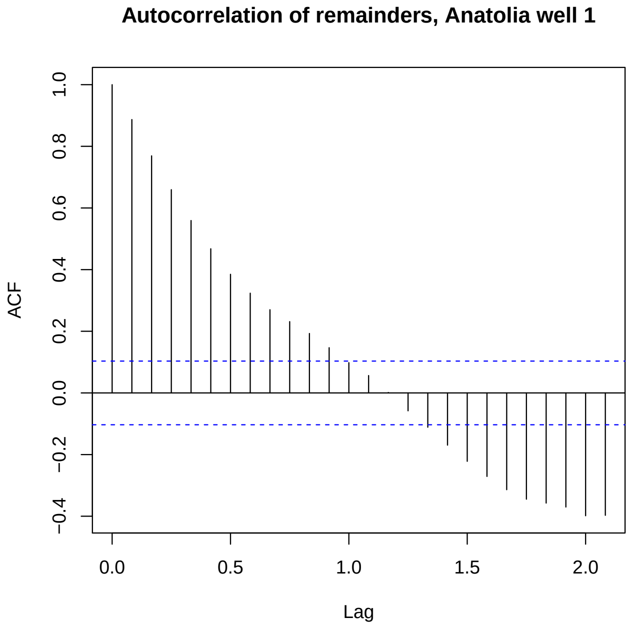
**Task 34 :** Display the autocorrelation of the remainder groundwater levels, after trend and seasonal components have been removed, using the smooth trend removal and a smoothing window of two years. •

Again we use `stl` to decompose the series. The remainder series, i.e., after computing the trend and cycle, is extracted from the result as field `time.series` (using the `$` field extraction operator), column `"remainder"`, and saved as a separate object for convenience:

```
gw.stl <- stl(gw, s.window=2*frequency(gw), t.window=84)
gw.r <- gw.stl$time.series[,"remainder"]
plot(gw.r, ylab="Remainder (m)")
title(main="Anatolia well 1 level, remainder from trend & cycle")
abline(h=0, lty=2, col="red")
```



```
print(acf(gw.r, plot=F))

##
## Autocorrelations of series 'gw.r', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
##  1.000  0.887  0.769  0.659  0.559  0.468  0.385  0.324  0.270  0.231
## 0.8333 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
##  0.193  0.147  0.098  0.056  0.001 -0.058 -0.111 -0.169 -0.222 -0.271
## 1.6667 1.7500 1.8333 1.9167 2.0000 2.0833
## -0.314 -0.345 -0.357 -0.370 -0.398 -0.397

acf(gw.r, main="Autocorrelation of remainders, Anatolia well 1")
```



The blue dashed lines show correlations that are not provably different from zero.

**Q36** : *Describe the autocorrelation of the remainders, and interpret in terms of the processes causing the groundwater level to vary.* *Jump to A36* •

### 3.5 Partial autocorrelation

A more subtle concept is **partial** autocorrelation of a time series. This gives the correlation between measurements and their lagged measurements that is *not* explained by the correlations with a shorter lag. For example, if all autocorrelation can be explained at lag 1, then there is no partial autocorrelation at lags 2, 3, ..., so that the apparent autocorrelation at these lags which we see in the `acf` function can be explained by repeated lag-1 correlations.

For lag $k$, the partial autocorrelation is the autocorrelation between $z_t$ and $z_{t+k}$ with the linear dependence of $z_t$ on $z_{t+1} \ldots z_{t+k-1}$ removed.

The partial and ordinary autocorrelations for lag 1 are the same.

Partial autocorrelations are computed as follows:

$$\mathbf{P_k}\boldsymbol{\phi}_\mathbf{k} = \boldsymbol{\rho}_\mathbf{k} \tag{2}$$

$$\mathbf{P_k} = \begin{bmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{k-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{k-2} \\ . & . & . & \cdots & . \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \cdots & 1 \end{bmatrix} \tag{3}$$

$$\rho_j = \phi_{k,1}\rho_{j-1} + \cdots + \phi_{k,k}\rho_{j-k}, \; j = 1, 2, \ldots k \tag{4}$$

where $\phi_{k,j}$ is coefficient $j$ of an autoregressive process of order $k$.

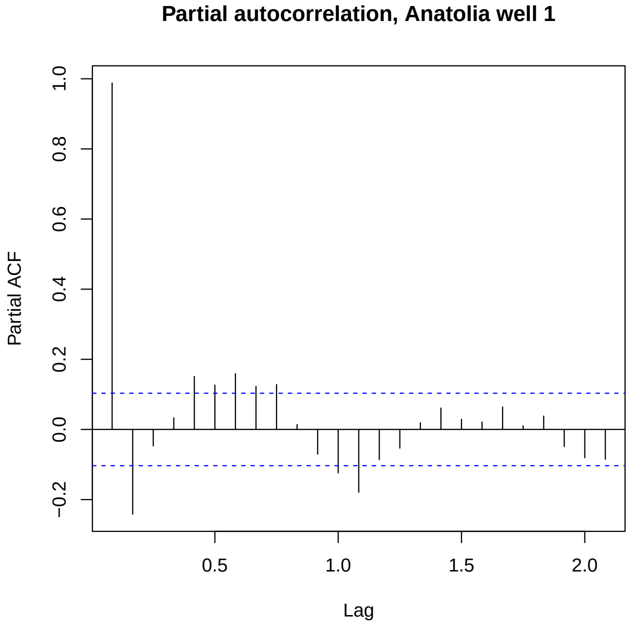
> **Note:** The partial autocorrelation function is used to estimate the order of ARIMA models (§4.4).

---

**Task 35** : Compute and display the partial autocorrelation of the groundwater levels. •

These are produced by the `pacf` "partial autocorrelation" function.

```
print(pacf(gw,plot=F))

##
## Partial autocorrelations of series 'gw', by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333
##  0.988 -0.242 -0.047  0.033  0.151  0.127  0.159  0.123  0.128  0.014
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.070 -0.124 -0.179 -0.086 -0.053  0.019  0.061  0.029  0.021  0.064
## 1.7500 1.8333 1.9167 2.0000 2.0833
##  0.010  0.038 -0.049 -0.081 -0.085

pacf(gw, main="Partial autocorrelation, Anatolia well 1")
```



As with the graph produced by `acf`, the blue dashed lines show correlations that are not provably different from zero.

---

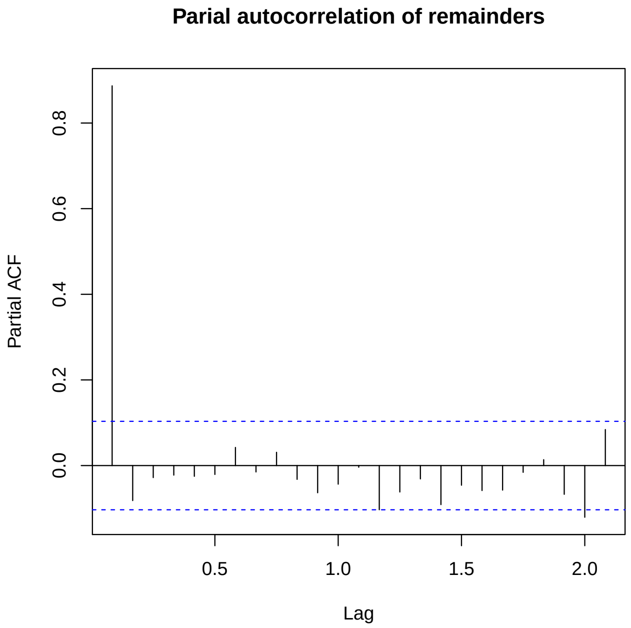
**Q37 :** *What are the partial autocorrelations that are different from zero?* *Jump to A37* •

---

**Task 36 :** Display the autocorrelation of the remainder groundwater levels, after trend and seasonal components have been removed, using the smooth trend removal and a smoothing window of two years. •

The required remainders were computed in the previous setion.

```
print(pacf(gw.r,plot=F))

##
## Partial autocorrelations of series 'gw.r', by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500 0.8333
##  0.887 -0.082 -0.028 -0.022 -0.025 -0.021  0.042 -0.015  0.031 -0.032
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
## -0.064 -0.043 -0.004 -0.103 -0.062 -0.031 -0.091 -0.046 -0.058 -0.057
## 1.7500 1.8333 1.9167 2.0000 2.0833
## -0.016  0.014 -0.067 -0.121  0.084

pacf(gw.r, main="Parial autocorrelation of remainders")
```



---

**Q38 :** *What are the partial autocorrelations that are different from zero? How does this differ from the partial autocorrelations of the original series? What is the interpretation?* *Jump to A38* •

## 3.6 Spectral analysis

Another second-order summary is provided by **spectral analysis**. This is based on the theory, due to Fourier, that any second-order stationary series (i.e., with trend removed and auto-covariance not dependent on position in the series) can be decomposed into sums of sines and cosines with increasing frequencies, each of varying amplitude or "power". In some cases we can guess these already (e.g., annual cycles for the groundwater wells), but for some series we do not know *a priori*. The periodogram also reveals the relative density (roughly, importance of the component) at each frequency..

The **frequency** $\omega$ is defined as the number of divisions of one **cycle**. So for a one-year cycle (as in the groundwater levels), $\omega = 12$ is a monthly frequency. By the Nyquist-Shannon sampling theorem, a function is completely determined by sampling at a rate of $1/2\omega$, so that if we have a time series with $n$ samples per cycle we can estimate spectral densities for $n/2$ frequencies.

The theoretical decomposition of the covariance sequence $\gamma_t$ into **spectral densities** $f$ is:

$$\gamma_t = \frac{1}{2\pi}\int_{-1/2}^{+1/2} e^{2\pi i\omega t} f(2\pi\omega_f)d\omega_f$$

where $\omega_f$ is the frequency expressed as cycles per unit of time.

This can be inverted to give the density at each frequency:

$$f(\omega) = \gamma_0\left[1 + 2\sum_i^{\infty} \rho_t cos(\omega t)\right]$$

where $\gamma_0$ is the overall covariance.

The spectral density is estimated by the **periodogram**, which relates the density to frequency. The periodogram at a given frequency $\omega$ is defined as the squared correlation between the time series $X$ and the sine/cosine waves at that frequency:

$$I(\omega) = \frac{1}{n}\left|\sum_t e^{-i\omega t}X_t\right|^2$$

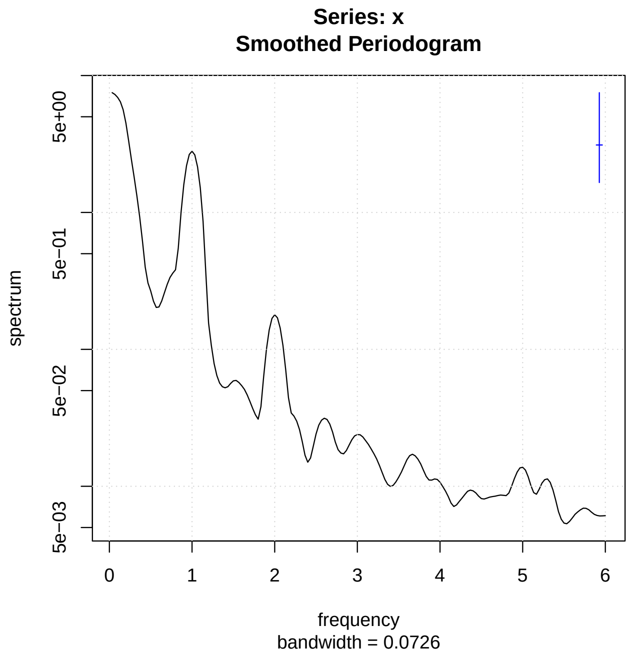
The `spectrum` function computes and graphs the power spectrum. By default it removes any linear trend, it tapers the series at its ends, and it presents the spectrum on a logarithmic scale.

The spectrum is scaled by $1/\omega$, i.e., the inverse of the frequency, so that the spectral density is computed over the range $-\omega/2 \ldots + \omega/2$; since the function is symmetric, it is displayed for $0 \ldots + \omega/2$. In the groundwater example, the frequency is 12 and so the decomposition is from $0 \ldots 6$ periods per cycle. One period per cycle is annual, 6 periods is bi-monthly.

The raw spectrum is too noisy to interpret, so it is usually smoothed with so-called Daniell windows, which give half-weight to end values. The window width is specified with the `spans` optional argument; optional values are trial-and-error, until the main features of the periodogram are revealed.

**Task 37** : Compute and display a smoothed periodogram of the groundwater levels, on the default logarithmic scale. •

```
s <- spectrum(gw, spans=c(5,7)); grid()
str(s)

## List of 16
##  $ freq     : num [1:180] 0.0333 0.0667 0.1 0.1333 0.1667 ...
##  $ spec     : num [1:180] 7.5 7.28 6.92 6.43 5.62 ...
##  $ coh      : NULL
##  $ phase    : NULL
##  $ kernel   :List of 2
##   ..$ coef: num [1:6] 0.1667 0.1562 0.125 0.0833 0.0417 ...
##   ..$ m   : int 5
##   ..- attr(*, "name")= chr "mDaniell(2,3)"
##   ..- attr(*, "class")= chr "tskernel"
##  $ df       : num 14.3
##  $ bandwidth: num 0.0726
##  $ n.used   : int 360
##  $ orig.n   : int 360
##  $ series   : chr "x"
##  $ snames   : NULL
##  $ method   : chr "Smoothed Periodogram"
##  $ taper    : num 0.1
##  $ pad      : num 0
##  $ detrend  : logi TRUE
##  $ demean   : logi FALSE
##  - attr(*, "class")= chr "spec"

head(s$spec, n=12)

##  [1] 7.4972852 7.2767232 6.9218806 6.4268515 5.6236966 4.4911276
##  [7] 3.3260429 2.4309815 1.8019706 1.3133122 0.9278068 0.6206158
```



Note that the spectrum is displayed on logarithmic scale in the plot. The blue vertical line (upper right corner) gives the 95% confidence interval that the reported density is different from zero.

The x-axis of the spectrum shows the **period**, i.e., the inverse of the declared frequency. For example “1” is a full period (one cycle), “2” is half a cycle, “3” is one-third of a cycle, etc.; these are the **harmonics**. So in this example with $\omega = 12$, the spectral density at $x = 1$ is for one cycle (12 months, one year) and the density at $x = 2$ is for a half-cycle (6 months, half-year).

The resolution of the decomposition is determined by the length of the time series (here, 360 observations); the resulting spectral decomposition is half this length (here, $360/2 = 180$), and this is estimated by the number of total cycles in the series (here, $360/12 = 30$ annual cycles), so the finest period is $\omega/2$, here $12/2 = 6$, each period divided into total cycles:

```
frequency(gw)

## [1] 12

length(gw)/frequency(gw)

## [1] 30

head(s$freq,n=length(gw)/frequency(gw))

##  [1] 0.03333333 0.06666667 0.10000000 0.13333333 0.16666667 0.20000000
##  [7] 0.23333333 0.26666667 0.30000000 0.33333333 0.36666667 0.40000000
## [13] 0.43333333 0.46666667 0.50000000 0.53333333 0.56666667 0.60000000
## [19] 0.63333333 0.66666667 0.70000000 0.73333333 0.76666667 0.80000000
## [25] 0.83333333 0.86666667 0.90000000 0.93333333 0.96666667 1.00000000
```

**Q39** : *At what frequencies are the largest periodic components of the Fourier decomposition? What is the interpretation?* *Jump to A39* •

We find the largest components by sorting the spectrum, using the `sort` function with the optional `index.return` argument to save the positions of each sorted element in the original vector, and grouping nearby peaks.

```
ss <- sort(s$spec, decreasing=T, index.return=T)
str(ss)

## List of 2
##  $ x : num [1:180] 7.5 7.28 6.92 6.43 5.62 ...
##  $ ix: int [1:180] 1 2 3 4 5 6 7 30 29 31 ...

hi <- ss$x>.15
which(hi)

##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
## [23] 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

ss$x[hi]

##  [1] 7.4972852 7.2767232 6.9218806 6.4268515 5.6236966 4.4911276
##  [7] 3.3260429 2.8008863 2.6526631 2.6311885 2.4309815 2.1987844
## [13] 2.1528798 1.8019706 1.5936610 1.5100103 1.3133122 0.9948927
## [19] 0.9278068 0.8599072 0.6206158 0.5479999 0.3993403 0.3815691
## [25] 0.3620958 0.3602574 0.3351769 0.3055965 0.2987287 0.2668042
## [31] 0.2600752 0.2258021 0.2248440 0.2041092 0.2024376 0.1789095
## [37] 0.1699968 0.1681654 0.1557876

ss$ix[hi]

##  [1]  1  2  3  4  5  6  7 30 29 31  8 28 32  9 27 33 10 26 11 34 12 25
## [23] 13 24 35 23 22 14 21 15 20 19 16 18 17 60 61 59 36

sort(s$freq[ss$ix[hi]])

##  [1] 0.03333333 0.06666667 0.10000000 0.13333333 0.16666667 0.20000000
##  [7] 0.23333333 0.26666667 0.30000000 0.33333333 0.36666667 0.40000000
## [13] 0.43333333 0.46666667 0.50000000 0.53333333 0.56666667 0.60000000
## [19] 0.63333333 0.66666667 0.70000000 0.73333333 0.76666667 0.80000000
## [25] 0.83333333 0.86666667 0.90000000 0.93333333 0.96666667 1.00000000
## [31] 1.03333333 1.06666667 1.10000000 1.13333333 1.16666667 1.20000000
## [37] 1.96666667 2.00000000 2.03333333
```

The list of indices is in units of inverse frequency; we can see three peaks: near zero (corresponding to no cycles, i.e., the mean), near one (annual), and centred on two (6-month); this harmonic is not provably different from zero.

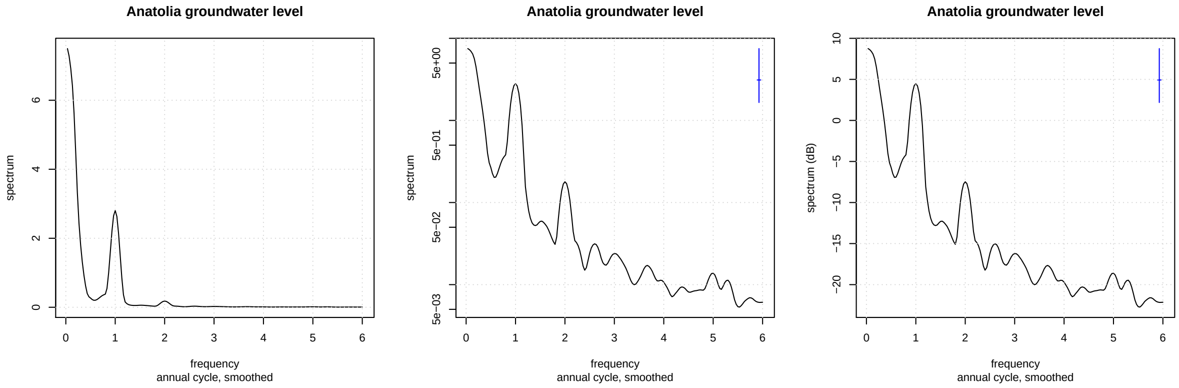
The log scale display for the spectral density is the default in order to more easily show the lower-power components. The raw density is shown by specifying the optional `log="no"` argument; specifying `log="dB"` argument shows the spectrum as decibels[9] as is conventional in signal processing.

---

**Task 38** : Compute and display the periodogram of the groundwater levels, on three scales: linear, log, and decibel. •

[9] $10 \cdot \log 10(I(\omega))$

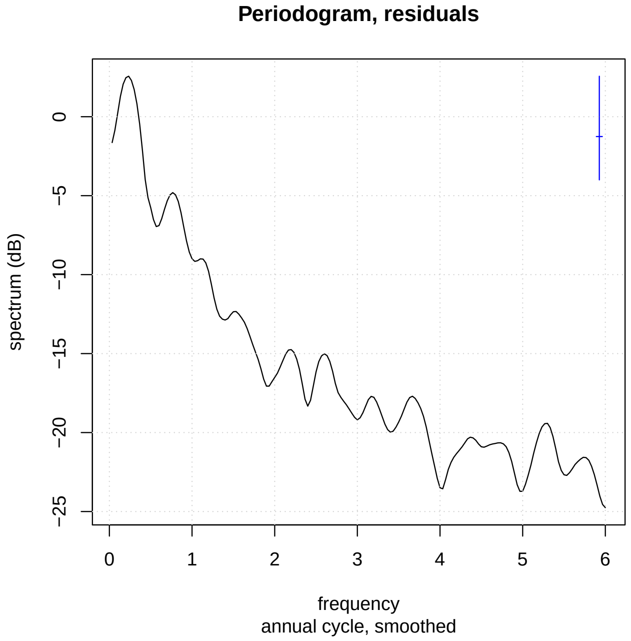
```
par(mfrow=c(1,3))
spectrum(gw, spans=c(5,7), log="no", main="Anatolia groundwater level",
         sub="annual cycle, smoothed")
grid()
spectrum(gw, spans=c(5,7), log="yes", main="Anatolia groundwater level",
         sub="annual cycle, smoothed")
grid()
spectrum(gw, spans=c(5,7), log="dB", main="Anatolia groundwater level",
         sub="annual cycle, smoothed")
grid()
par(mfrow=c(1,1))
```



The annual cycle is well-known; indeed we have already (§3.3) decomposed the time series into trend, cycle and remainder. By removing the dominant cycle we may get a sharper view of other frequencies.

**Task 39** : Compute and display the periodogram of the groundwater levels, after removing trend and annual cycles. •

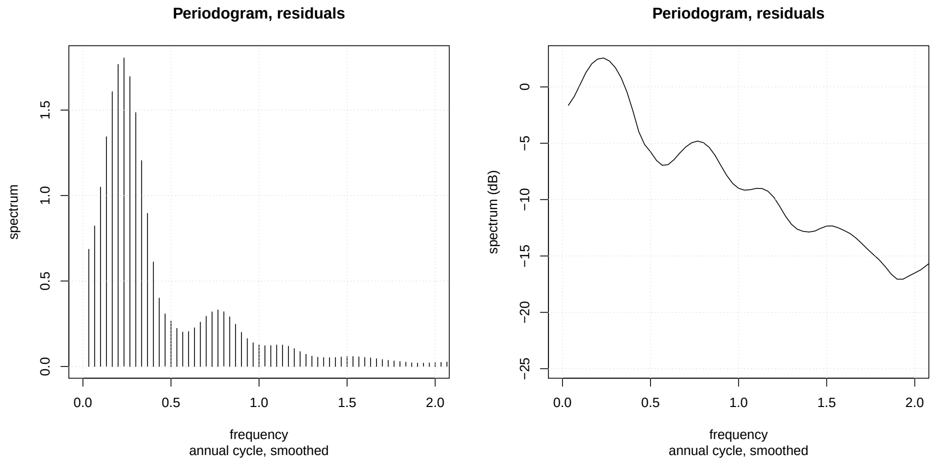
```
sp.gw <- spectrum(gw.r, span=c(5,7), log="dB", main="Periodogram, residuals",
                  sub="annual cycle, smoothed")
grid()
```



Notice that there is no peak corresponding to one year; this has been removed with the annual cycle. There are also no peaks at the harmonics (1/2, 1/3 etc. years).

Zooming in on the first two frequencies, i.e., one and one-half year, with two different views:

```
par(mfrow=c(1,2))
spectrum(gw.r, span=c(5,7), log="no", main="Periodogram, residuals",
         sub="annual cycle, smoothed", xlim=c(0,2), type="h")
grid()
s <- spectrum(gw.r, span=c(5,7), log="dB", main="Periodogram, residuals",
              sub="annual cycle, smoothed", xlim=c(0,2))
grid()
sp.gw$freq[which.max(s$spec)]

## [1] 0.2333333

frequency(gw)/(s$freq[which.max(s$spec)])

## [1] 51.42857

which.max(s$spec[16:30])

## [1] 8

sp.gw$freq[which.max(s$spec[16:30])+15]

## [1] 0.7666667

frequency(gw)/(s$freq[which.max(s$spec[16:30])+15])

## [1] 15.65217

par(mfrow=c(1,1))
```



**Q40** : *What are the periods of the highest spectral densities? How can these be explained physically?* *Jump to A40* •

## 3.7 Answers

**A21** : *Until 1981 the amplitude is small, less than 2 m. It then increases but there are year-to-year differences. In 2002 the amplitude was highest.* *Return to Q21* •

**A22** : *There is a definite annual cycle: September has the deepest levels (at the end of the extractive period) and April the shallowest (at the end of the winter rains). Winter months are a bit more variable than summer. Obvious outliers from the overall pattern are the deep levels in one December – March period.* *Return to Q22* •

**A23** : *The monthly series has thrice the points and thus more detail; however the pattern remains clear and the high/low values for each cycle are not much different, so the lower temporal resolution does not much affect interpretation.* *Return to Q23* •

**A24** : *The peaks and valleys are less extreme, and some month-to-month irregularities are removed.* *Return to Q24* •

**A25** : *The peaks and valleys are further smoothed, month-to-month irregularities are removed.* *Return to Q25* •

**A26** : *The annual cycle is removed, this shows the year-to-year variation.*

*Return to Q26* •

---

A*27* : *The default parameter (2/3) shows the overall trend (increasing groundwater) depth) slightly adjusted for local phenonomena; increasing the parameter to 1 removes almost all variation and results in almost a straight line; decreasing to 1/3 adjusts more closely to trends that go over a few years, for example the initial overall decrease in depth for the first three years, and the unusual extraction in 1988. The paramater value 1/10 adjusts very closely to each year and obscures the overall trend. The parameter value of 1/3 seems most useful here.* *Return to Q27* •

---

A*28* : *Numerically, the mean is zero and the numbers represent deviations from the cycle at each position in it. Thus at the earlier years the groundwater is shallower (negative values), later it is deeper (positive values). For most of the series the seasonal fluctuations have been mostly removed, but prior to 1980 they are amplified. This is because in that period there was not much seasonal fluctuation, so averaging the cycle over all years amplifies these early small fluctuations.* *Return to Q28* •

---

A*29* : *The decomposed series has class* `stl` *and consists of three named time series: "seasonal", "trend", and "remainder", organized as a matrix of class* `mts` *("multiple time series") with three columns (one for each series).* *Return to Q29* •

---

A*30* : *The average seasonal component has amplitude* $\approx \pm 1.5m$ *depth; the trend ranges over* $25m$ *depth and generally increases, after an initial decrease; the remainder ranges from* $\approx -2 \ldots 4.5m$*, but all except 1988 are within a more restricted range,* $\approx \pm 1.5m$*. The remainders show strong auto-correlation within each cycle. Thus the trend is by far the largest componet; the seasonal and remainer are similar orders of magnitude.* *Return to Q30* •

---

A*31* : *The seasonal component can change with time; here the amplitude increases until about 2000 and then stabilizes. The cycles themselves may have different shapes: note the "shoulder" in the decreasing level in 1983-1990, which is absent from later yeard. The amplitude of the remainder has been reduced, because the cycles are more closely fit.* *Return to Q31* •

---

A*32* : *The smoother decomposition has a smoother trend (of course); the seasonal component is the same, so the remainders are larger and their serial auto-correlation extends over a longer span. The smooth trend seems to represent a long-term change in groundwater, probably due to increasing extraction for agriculture. The rough trend has noise that does not seem to be due to a long-term process.* *Return to Q32* •

---

A*33* : *The auto-correlations gets weaker for the first six or seven months, but* *then strengthens; this reflects the annual cycles.* Return to Q33 •

---

A34 : *The detailed scatterplot shows which measurements are more or less correlated to the lagged measurement; also the lines show an evolution of this over time: i.e., whether the correlation is systematically greater or less.* Return to Q34 •

---

A35 : *Groundwater levels are highly positively correlated, decreasing from 0.9878 at lag 1 to 0.7965 at two years (24 months). The correlation increases locally at cycle lengths (one and two years, i.e., 12 and 24 months).*

*Physically, this reflects the fact that groundwater level can not fluctuate rapidly month-to-month; the change in level must be smooth.* Return to Q35 •

---

A36 : *The remainders are positively autocorrelated within one year (up to 11 months); they are then not different from zero (no correlation) until the 16th month, after which the autocorrelation is increasingly negative.*

*The removal of the trend has taken away most of the autocorrelation due to the continuous nature of groundwater level change. Removal of the cycle has taken away any autocorrelation due to seasonality (i.e., extraction and recharge at the same seasons each year), which is reflected in the lack of correlation in the year to year remainders (lag 12).*

*The remainder represents the local effects after accounting for trend and cycle. There is positive autocorrelation within one year, because in a wet (or dry) year the level can not fluctuate rapidly and so tends to stay relatively high (or low). The negative autocorrelation between subsequent years means that relatively wet (dry) years tend to be followed by relatively dry (wet) remainders, i.e., after accounting for trend and cycle.* Return to Q36 •

---

A37 : *Lag 2 has a substantial negative partial autocorrelation: -0.2415. Then months 5 – 9 have slightly positive partial autocorrelations; the negative partial autocorrelation appears again at 12 and 13 months.* Return to Q37 •

---

A38 : *The only partial autocorrelation provably different from zero is the first lag (one month). Once this is accounted for, the other lags have no autocorrelation. The remainders have only a one-month autocorrelation, and thus could be modelling by a first-order autoregressive process (§4.4.1).* Return to Q38 •

---

A39 : *At frequency 1 (one year) there is a large component (-9 dB); i.e., a strong annual cycle. There is also a much weaker component (-16.51 dB) at the six-month frequency.* Return to Q39 •

---

A40 : *The highest spectral density is at $7/30 = 0.2\bar{3}$ cycles per year, i.e., 51.4 months per cycle, or about 4 years 3 months. A much smaller peak is at $23/30 = 0.7\bar{6}$ cycles per year, i.e., 15.7 months, or about 1 year 3 months. These seem to be artefacts of the particular data series.* Return to Q40 •

# 4 Modelling a time series

A **model** of a time series is a mathematical formulation that summarizes the series. This formulation has a model **form** and several **parameters**. A simple example is modelling monthly temperature as a sine wave; this would have a mean, amplitude, and phase (the period would be defined as one year). The twelve monthly temperatures could then be recovered from these three parameters.

There are several aims of modelling a time series:

1. Understand the underlying **process(es)** that produced it;
2. Use the model to **predict** into the future (**forecasting**), at missing data points, or into the past;
3. Use the model to **simulate** similar time series (§8).

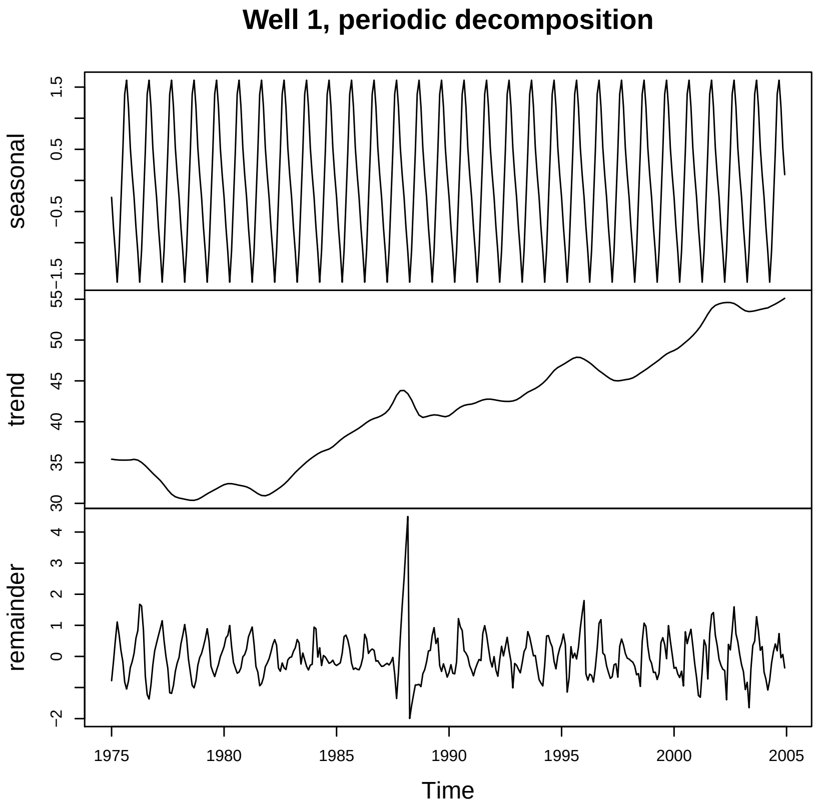
We begin with an examination of some model forms. These arise from different processes in time. A requirement of successful time series modelling is to fit the model form to the actual process. We may have some idea of the process from physical or social principles. Or, we can see which model form best fits the series, and hope that this continues into the future for forecasting.

## 4.1 Modelling by decomposition

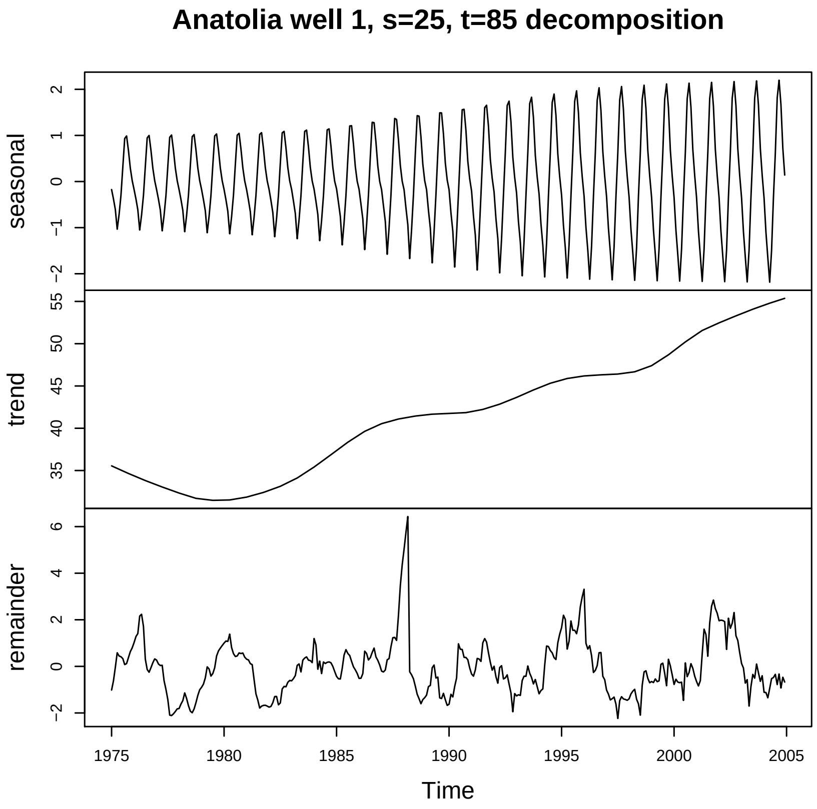
We have already seen (§3.3) that a time series can be decomposed into a trend, cycle, and residual using the `stl` function. The residual is by definition noise in this decomposition, the other two components are a **model**, representing the long-term and cyclic behaviour of the series.

The problem with this approach is that the trend removal is empirical: its smoothness must be adjusted by the analyst. Recall the two decompositions of the groundwater data, with different smoothness:

```r
plot(stl(gw, s.window="periodic")$time.series,
     main="Well 1, periodic decomposition")
```



```
plot(stl(gw, s.window=25, t.window=85)$time.series,
     main="Anatolia well 1, s=25, t=85 decomposition")
```



The first decomposition makes no assumptions other than the annual cycle; the second specifies a (1) `s.window`, the span (number of lags) of the loess window for seasonal extraction; (2) `t.window`, the span for trend extraction. There is no theoretical way to determine these.

The decomposition gives insight into the processes underlying the time series.

**Q41** : *What could be the processes that control groundwater level in this well, as revealed by the decomposition?* *Jump to A41* •

However, there is no way to extrapolate (predict), because there is no model of the underlying process for the trend, nor of the amplitude of the period, only an empirical fit. We now examine some models that do allow prediction.

## 4.2 Modelling by OLS linear regression

The standard techniques of linear modelling can be applied to the time series, considering each observation as a function of time and position in the cycle. This may include interactions between cycle and position. However, cyclic components are better-handled by decomposition (§4.1); regression is more typically used to model a trend over time. Linear models can also be used for interpolation (gap filling) and extrapolation (prediction).

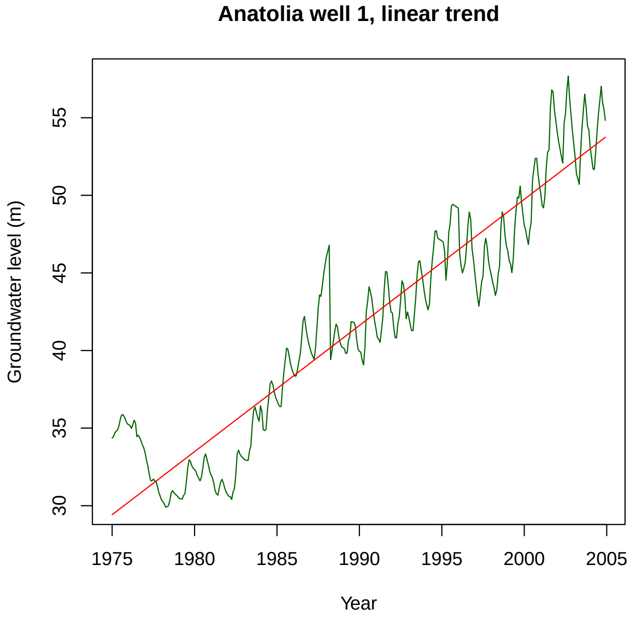
**Task 40** : Build a linear model of the trend in groundwater level, and evaluate its success and suitability. •

The `lm` function computes linear models, using predictors and responses from a dataframe; thus the dataframe `gw.f` must be used instead of the raw time series.

```
m.gw <- lm(gw ~ time, data=gw.f)
summary(m.gw)

##
## Call:
## lm(formula = gw ~ time, data = gw.f)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9134 -1.8445 -0.3193  1.4857  6.6562
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.576e+03  3.053e+01  -51.64   <2e-16 ***
## time         8.130e-01  1.534e-02   53.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.521 on 358 degrees of freedom
## Multiple R-squared:  0.887,Adjusted R-squared:  0.8867
## F-statistic:  2809 on 1 and 358 DF,  p-value: < 2.2e-16
```

**Task 41** : Plot the time series, with the fits from the linear model superimposed. •

```
plot(gw.f$time, gw.f$gw, type="l", col="darkgreen",
     ylab="Groundwater level (m)", xlab="Year")
title("Anatolia well 1, linear trend")
lines(x=as.numeric(gw.f$time), y=fitted(m.gw), col="red")
```



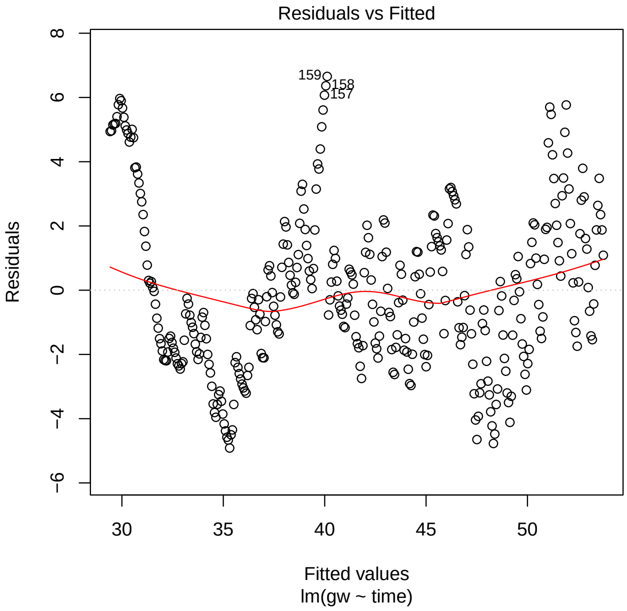
**Q42** : *How much of the variation in groundwater level is explained by this model? How well does it appear to explain the trend? What is the average annual increase in depth?* Jump to A42 •

Note that the adjusted $R^2$ and slope can be extracted from the model object:

```
summary(m.gw)$adj.r.squared

## [1] 0.8866509

coefficients(m.gw)["time"]

##      time
## 0.8130209
```

**Task 42** : Display a diagnostic plot of residuals vs. fitted values. •

```
plot(m.gw, which=1)
```



The diagnostic plot shows that this model violates one assumption of linear modelling: there be pattern to the residuals with respect to fitted values; most of the problem is at low fitted values, i.e., the beginning of the time series. There is also a large discrepency near 40 m, which corresponds to the anomalous year 1988.

## 4.3 Modelling by GLS linear regression

If residuals are correlated in time (as in this case), the OLS regression is not optimal. Instead, the trend should be fit by Generalized Least Squares (GLS).

In OLS the residuals $\varepsilon$ are assumed to be *independently* and *identically* distributed with the same variance $\sigma^2$:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon, \;\; \varepsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) \tag{5}$$

Whereas, in GLS residuals are considered to be a random variable $\eta$ that has a covariance structure:

$$\mathbf{y} = \mathbf{X}\beta + \eta, \;\; \eta \sim \mathcal{N}(0, \mathbf{V}) \tag{6}$$

where $\mathbf{V}$ is a positive-definite variance-covariance matrix of the model residuals. In a time series, the covariances in this matrix (off-diagonals) are naturally based on the time distance between observations, using some model of temporal correlation, e.g. AR(1) (§4.4.1, below).

The solution is:

$$\hat{\beta}_{\text{GLS}} = (\mathbf{X}^T \mathbf{V}^{-1} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{V}^{-1} \mathbf{y} \tag{7}$$

where $\mathbf{V}$ the variance-covariance matrix of the residuals $\mathbf{V} = \sigma^2 C$, where $\sigma^2$ is the variance of the residuals and $\mathbf{C}$ is the correlation matrix.

The computations are performed with the `gls` function of the `nlme` 'Non-linear mixed effects models' package [2].

---

**TASK 43 :** Set up and solve a GLS model, using the covariance structure estimated from the variogram of the OLS residuals. •

The linear model formulation is the same as for `lm`. However:

- It has an additional argument `correlation`, which specifies the correlation structure.
- This is built with various correlation models; we use `corAR1` for AR(1) temporal correlation. This requires two arguments:
  - **value** the value of the lag 1 autocorrelation, which must be between -1 and 1;
  - **form** a one-sided formula specifying the time covariate, if any. In this case there is no covariate, so only an intercept is specified as ~1.

  The value can be changed during optimization and will be reported in the results.

We obtain the initial value from the `acf` function; the second value is the one-lag autocorrelation.

> **Note:** For a list of the predefined model forms see `?corClasses`. Users can also define their own `corStruct` classes.

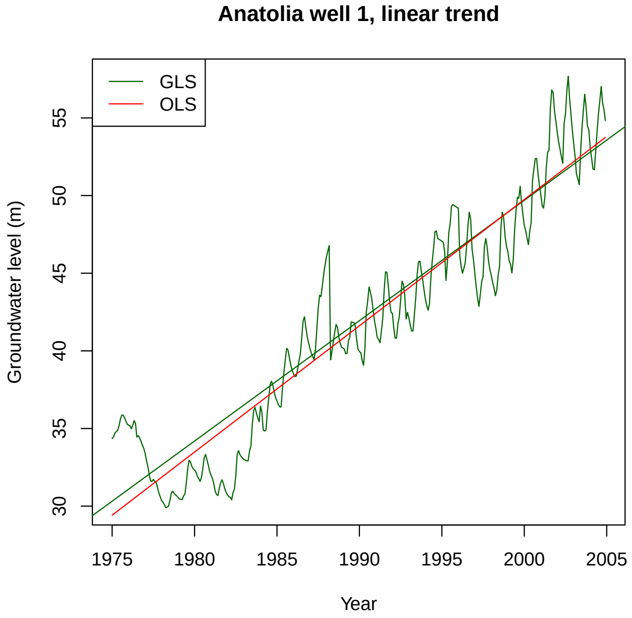
```
library(nlme)
(cor.value <- acf(gw, plot=FALSE)$acf[2])

## [1] 0.9878291

summary(m.gw.gls <- gls(model=gw ~ time, data=gw.f,
             correlation=corAR1(value=cor.value, form = ~1)))

## Generalized least squares fit by REML
##   Model: gw ~ time
##   Data: gw.f
##        AIC      BIC    logLik
##   908.7075 924.2297 -450.3538
##
## Correlation Structure: AR(1)
##  Formula: ~1
##  Parameter estimate(s):
##       Phi
## 0.9571062
##
## Coefficients:
##                  Value Std.Error   t-value p-value
## (Intercept) -1500.4356 200.05311 -7.500186       0
## time            0.7751   0.10053  7.709787       0
##
##  Correlation:
##      (Intr)
## time -1
```

```
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -1.9019852 -0.7551349 -0.2290400  0.4591564  2.1482906
##
## Residual standard error: 2.911948
## Degrees of freedom: 360 total; 358 residual

coef(m.gw)

##   (Intercept)          time
## -1576.2972795     0.8130209

coef(m.gw.gls)

##   (Intercept)          time
## -1500.4355894     0.7750657
```

**Q43 :** *What is the estimate of the AR(1) temporal correlation returned by `gls`? How does this compare with the initial value estimated with `pacf`?* *Jump to A43* •

**Task 44 :** Plot this GLS trend on the time series, with the OLS trend for comparison. •

```
plot(gw.f$time, gw.f$gw, type="l", col="darkgreen",
     ylab="Groundwater level (m)", xlab="Year")
title("Anatolia well 1, linear trend")
abline(m.gw.gls, col="darkgreen")
lines(x=as.numeric(gw.f$time), y=fitted(m.gw), col="red")
legend("topleft", c("GLS","OLS"), lty=1, col=c("darkgreen","red"))
```

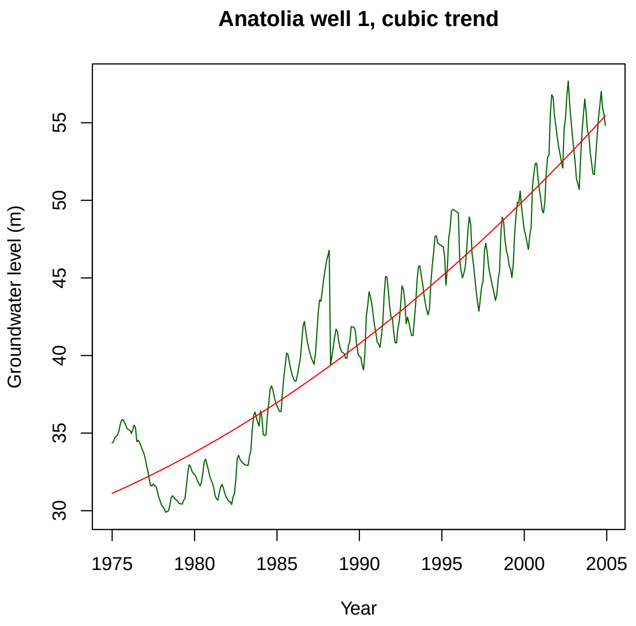


**Q44 :** *How has the slope of the trend changed from the OLS to GLS* *model?* *Jump to A44* •

**Higher-order trend** The trend seems to have two inflection points (around 1980 and 1985), so perhaps fitting a cubic trend might give a better model. The anova function compares two models.

```
m.gw.3 <- lm(gw ~ I(time^3)  + time, data=gw.f)
summary.lm(m.gw.3)

##
## Call:
## lm(formula = gw ~ I(time^3) + time, data = gw.f)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7362 -1.9403 -0.1648  1.6461  7.4775
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.864e+04  4.985e+03   5.746 1.96e-08 ***
## I(time^3)    1.917e-06  3.163e-07   6.062 3.42e-09 ***
## time        -2.197e+01  3.758e+00  -5.845 1.14e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.403 on 357 degrees of freedom
## Multiple R-squared:  0.8975,Adjusted R-squared:  0.8969
## F-statistic:  1563 on 2 and 357 DF,  p-value: < 2.2e-16

anova(m.gw.3, m.gw)

## Analysis of Variance Table
##
## Model 1: gw ~ I(time^3) + time
## Model 2: gw ~ time
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    357 2062.1
## 2    358 2274.4 -1   -212.25 36.746 3.424e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
plot(gw.f$time, gw.f$gw, type="l", col="darkgreen",
     ylab="Groundwater level (m)", xlab="Year")
title("Anatolia well 1, cubic trend")
lines(x=as.numeric(gw.f$time), y=fitted(m.gw.3), col="red")
```



**Q45** : *Is the cubic trend model better than the linear trend?* *Jump to A45* •

**Splitting the series** Clearly the series to about 1978 is different from that after; perhaps the extraction did not begin until then?

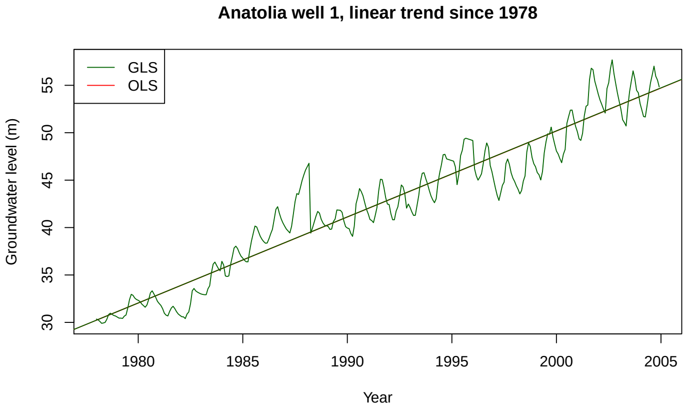
**Task 45** : Model the trend since 1978 with both an OLS and a GLS model. •

The `subset` function is used to limit the time series in the dataframe. We have to re-compute the starting values for the autocorrelation parameter of the GLS model from just this part of the series.

```
gw.f.78 <- subset(gw.f, gw.f$year > 1977)
summary(m.gw.78 <- lm(gw ~ time, data=gw.f.78))

##
## Call:
## lm(formula = gw ~ time, data = gw.f.78)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0422 -1.4399 -0.0702  1.3550  7.3310
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1763.3608    30.4744  -57.86   <2e-16 ***
## time            0.9068     0.0153   59.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.147 on 322 degrees of freedom
## Multiple R-squared:  0.916,Adjusted R-squared:  0.9157
## F-statistic:  3511 on 1 and 322 DF,  p-value: < 2.2e-16
```

```
(cor.value <- acf(gw.f.78, plot=FALSE)$acf[2])

## [1] 0.9841868

summary(m.gw.78.gls <- gls(model=gw ~ time, data=gw.f.78,
             correlation=corAR1(value=cor.value, form = ~1)))

## Generalized least squares fit by REML
##   Model: gw ~ time
##   Data: gw.f.78
##        AIC      BIC   logLik
##   843.0141 858.1123 -417.507
##
## Correlation Structure: AR(1)
##  Formula: ~1
##  Parameter estimate(s):
##       Phi
## 0.9224721
##
## Coefficients:
##                  Value Std.Error   t-value p-value
## (Intercept) -1764.3798  143.9822 -12.25416       0
## time            0.9073    0.0723  12.54906       0
##
##  Correlation:
##      (Intr)
## time -1
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -2.23240755 -0.64109407 -0.03336087  0.59381099  3.23268822
##
## Residual standard error: 2.265006
## Degrees of freedom: 324 total; 322 residual
```

```
plot(gw.f.78$time, gw.f.78$gw, type="l", col="darkgreen",
     ylab="Groundwater level (m)", xlab="Year")
title("Anatolia well 1, linear trend since 1978")
lines(x=as.numeric(gw.f.78$time),
      y=fitted(m.gw.78.gls), col="red")
abline(m.gw.78, col="red")
abline(m.gw.78.gls, col="darkgreen")
legend("topleft", c("GLS","OLS"), lty=1, col=c("darkgreen","red"))
```



For this portion of the time series the GLS and OLS models are almost identical:

```
coef(m.gw.78)

##   (Intercept)          time
## -1763.3607555     0.9067699

coef(m.gw.78.gls)

##  (Intercept)         time
## -1764.379848     0.907287
```

**Task 46** : Compare the GLS model since 1978 with the GLS model for the whole series. •

```
coefficients(m.gw.gls)["time"]

##      time
## 0.7750657

coefficients(m.gw.78.gls)["time"]

##     time
## 0.907287
```

**Q46** : *Is the average annual change different for the model fit on the entire series versus the model fit on the post-1977 section? Which would you use in extrapolating into the future?* *Jump to A46* •

**Task 47** : Predict the groundwater level in January 2010. •

The generic `predict` method specialized to the `predict.lm` function when applied to a linear model object. The confidence interval of the fit is returned if the `interval` argument is provided; the `"prediction"` option returns the upper and lower bounds of the prediction at the specified confidence `level`:

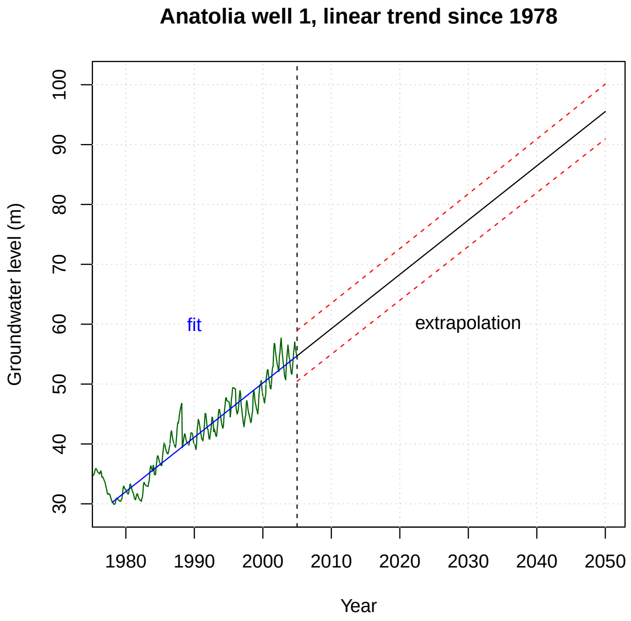
```
predict(m.gw.78, data.frame(time=2010), interval="prediction", level=0.9)

##        fit      lwr      upr
## 1 59.24676 55.66909 62.82443
```

**Task 48** : Predict the groundwater level from 2005 to 2055; graph this with its 95% confidence interval of prediction. •

Again we use the `predict.lm` method, this time with a sequence of times at which to predict.

```
gw.2050 <- predict.lm(m.gw.78, data.frame(time=2005:2050),
                      interval="prediction", level=0.95)
str(gw.2050)

##  num [1:46, 1:3] 54.7 55.6 56.5 57.4 58.3 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:46] "1" "2" "3" "4" ...
##   ..$ : chr [1:3] "fit" "lwr" "upr"

plot(gw.f$time, gw.f$gw, type="l", col="darkgreen",
     ylab="Groundwater level (m)", xlab="Year",
     xlim=c(1978,2050),
     ylim=c(floor(min(gw.f$gw)),ceiling(max(gw.2050[,"upr"])) ))
title("Anatolia well 1, linear trend since 1978")
```

```
grid()
abline(v=2005, lty=2)
lines(x=as.numeric(gw.f$time[gw.f$year > 1977]),
      y=fitted(m.gw.78), col="blue")
lines(2005:2050, gw.2050[,"fit"])
lines(2005:2050, gw.2050[,"upr"], col="red", lty=2)
lines(2005:2050, gw.2050[,"lwr"], col="red", lty=2)
text(1990,60,"fit", col="blue")
text(2030,60,"extrapolation")
```



**Q47** : *Do you expect the groundwater level to be below 80 m by 2040? What factors determine how long the modelled trend may continue in the future?* *Jump to A47* •

### 4.3.1 Modelling a decomposed series

If we are interested in modelling an overall trend, the seasonal component of a series is not interesting. Recall that this component can be identified by decomposition (§3.3); if this is removed the smoothed trend and residual components can be used as the basis of trend modelling. The residual noise quantifies the uncertainty of the parametric fit.

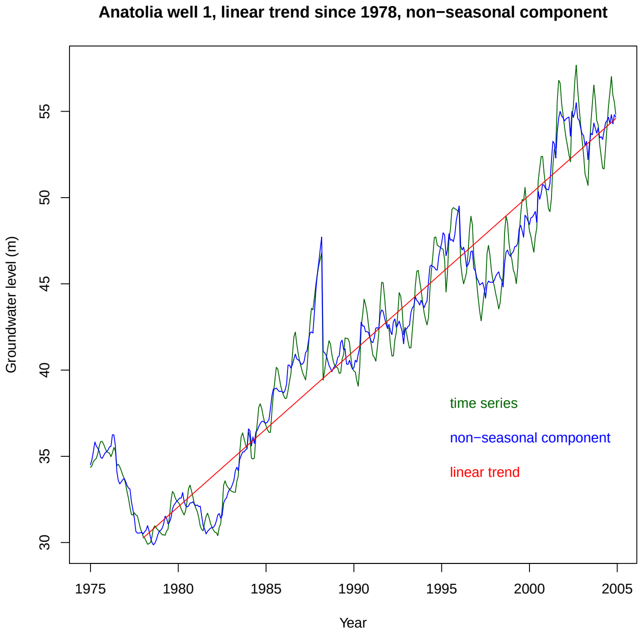
**Task 49** : Fit a linear trend to the non-seasonal component of the groundwater level. •

We first add a field to the time series dataframe and then use it for modelling. Again, the trend since 1978 is modelled.

```
gw.stl <- stl(gw, s.window=2*frequency(gw)+1)
gw.f$nonseas <- gw.stl$time.series[,"trend"] + gw.stl$time.series[,"remainder"]
str(gw.f)
```

```
## 'data.frame': 360 obs. of  6 variables:
##  $ gw     : Time-Series  from 1975 to 2005: 34.4 34.5 34.7 34.8 34.9 ...
##  $ year   : num  1975 1975 1975 1975 1975 ...
##  $ cycle  : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ time   : Time-Series  from 1975 to 2005: 1975 1975 1975 1975 1975 ...
##  $ in.yr  : num  -0.818 -0.727 -0.477 -0.378 -0.297 ...
##  $ nonseas: Time-Series  from 1975 to 2005: 34.5 34.8 35.3 35.8 35.6 ...

m.gw.nonseas <- lm(nonseas ~ time, data=subset(gw.f, gw.f$year > 1977))
summary(m.gw.nonseas)

##
## Call:
## lm(formula = nonseas ~ time, data = subset(gw.f, gw.f$year >
##     1977))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9693 -0.9538 -0.1106  0.8762  8.2621
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.757e+03  2.558e+01  -68.70   <2e-16 ***
## time         9.036e-01  1.284e-02   70.36   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.802 on 322 degrees of freedom
## Multiple R-squared:  0.9389,Adjusted R-squared:  0.9387
## F-statistic:  4950 on 1 and 322 DF,  p-value: < 2.2e-16
```

```
plot(gw.f$time, gw.f$gw, type="l", col="darkgreen",
     ylab="Groundwater level (m)", xlab="Year")
title("Anatolia well 1, linear trend since 1978, non-seasonal component")
lines(x=as.numeric(gw.f$time), y=gw.f$nonseas, col="blue")
lines(x=as.numeric(gw.f$time[gw.f$year > 1977]),
      y=fitted(m.gw.nonseas), col="red")
text(1995, 38, col="darkgreen", pos=4, "time series")
text(1995, 36, col="blue", pos=4, "non-seasonal component")
text(1995, 34, col="red", pos=4, "linear trend")
```



**Task 50 :** Compare this linear model (with the seasonal component removed) to the linear model of the series since 1978 computed in the previous subsection. Consider (1) the amount of variability explained; (2) the slope of the trend. •

```
(tmp <- summary(m.gw.nonseas)$adj.r.squared -
                              summary(m.gw.78)$adj.r.squared)

## [1] 0.02299186

tmp/summary(m.gw.78)$adj.r.squared

## [1] 0.02510743

coefficients(m.gw.nonseas)["time"]

##      time
## 0.9035588

coefficients(m.gw.78)["time"]

##      time
## 0.9067699
```

**Q48 :** *By how much does the proportion of variability explained by the model change when the seasonal component is removed before modelling the trend?* *Jump to A48*
•

**Q49 :** *By how much does the slope of the trend change when the seasonal component is removed before modelling the trend?* *Jump to A49*
•

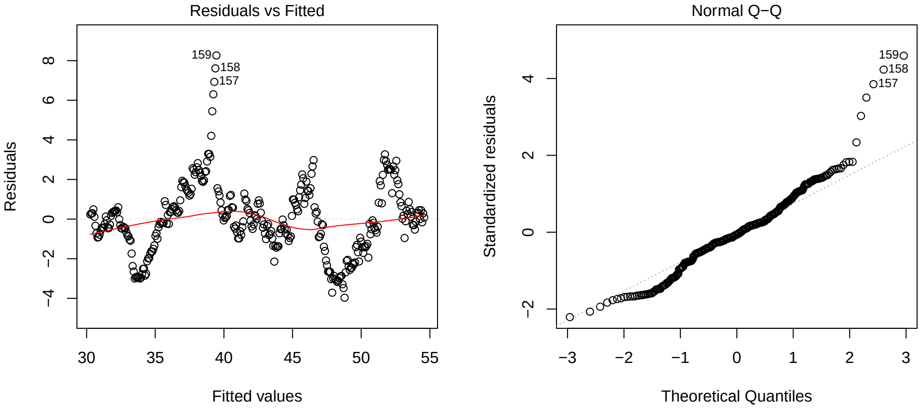
### 4.3.2 Non-parametric tests for trend

A *non-parametric* test is one that does not assume any underlying distribution. In the trend analysis of the previous section (§4.3.1) we assumed that the residuals (after accounting for seasonality and trend) was independently and identically normally-distributed (IIND); this is a requirement for using ordinary least squares (OLS) to fit a linear model.

**Task 51 :** Check that the residuals of the trend analysis are IID. •

There are various formal tests, but we will visualize the regression diagnostics with the `plot` function applied to linear model results, using the `which` argument to select graphs 1 (residuals vs. fitted values) and 2 (normal quantile-quantile plot of residuals).

```
par(mfrow=c(1,2))
plot(m.gw.nonseas, which=1:2)
par(mfrow=c(1,1))
```



**Q50** : *Do the residuals appear to be IIND?* *Jump to A50* •

Since the residuals do not meet the criteria for OLS, the confidence intervals computed for the slope may not be accurate; further the apparent trend may not be real. In this case the trend is quite strong so this is not an issue. Still, we discuss how to detect trend in a series where OLS models are not justified.

One approach is to use a **robust** regression [3] to estimate the trend its confidence limits.

Another approach is to use a non-parametric test. Hirsch et al. [11] present the Mann-Kendall test for trend, which has been included in the `Kendall` package written by Ian McLeod (co-author of [10]).

**Task 52** : Check for a trend in the groundwater levels time series since 1978. •

The `MannKendall` function of the `Kendall` package computes this test for a time series. The `SeasonalMannKendall` function does the same under the alternative hypothesisis that for one or more months the subsample is not distributed identically. In the present case the series is clearly seasonal, so we can either compute the seasonal test for the series or the non-seasonal test for the decomposed series (i.e., we remove the seasonality).These should give similar results.

```
require(Kendall)

## Loading required package:  Kendall

gw.1978 <- window(gw, start=c(1978,1), end=c(2005,12), extend=TRUE)
SeasonalMannKendall(gw.1978)

## tau = 0.892, 2-sided pvalue =< 2.22e-16

gw.nonseas.1978 <- ts(subset(gw.f, gw.f$year > 1977)$nonseas)
MannKendall(gw.nonseas.1978)

## tau = 0.862, 2-sided pvalue =< 2.22e-16
```

The $\tau$-value is a test statistic that is then compared to a theoretical value, resulting in the probability that a $\tau$-value this large could have occurred by chance in a series with no true trend.

---

**Q51** : *What is the probability that there is no trend in this series? Jump to A51* •

Hirsch et al. [11] also propose a non-parametric estimator of slope: compute all differences between the same month in all pairs of years:

$$d_{ijk} = \frac{x_{ij} - x_{ik}}{j - k}, \; i = 1 \ldots 12, \; 1 \le k < j \le n$$

where $n$ is the number of years. Then, estimate the slope $B$ by their **median**. Note that if the number of positive and negative differences are equal, the slope is estimated as zero (no trend). The differences $(x_{ij} - x_{ik})$ are for the twelve months indexed by $i$ and the years indexed by $k < j$ and then normalized by the number of years between each pair of values. So each $d_{ijk}$ is a slope between two years for a given month.

---

**Task 53** : Write a function to compute this non-parametric slope, also displaying a histogram of the individual slope estimates. •

```
MKslope <- function(ts) {
  f <- frequency(ts)
  n <- length(ts)/f
  d <- NULL
  for (j in n:2)
    for (k in (j-1):1)
      for (i in 1:f)
        d <- c(d, (ts[i + (j-1)*f] - ts[i + (k-1)*f])/(j-k));
  hist(d, main="individual slope estimates", xlab="slope")
  print(summary(d))
  return(median(na.omit(d)))
}
```

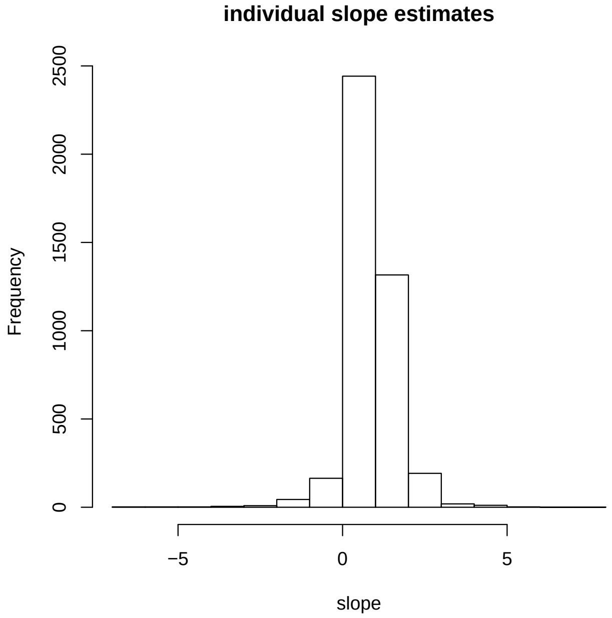
Note the use of `na.omit` to account for the possibility of missing values in the time series, and hence missing differences; this will be used in the following example (§4.3.3).

---

**Task 54** : Estimate the non-parametric slope with this function and compare it to the parametric (OLS) estimate. •

The appropriate slope for comparison is the linear model of the decomposed series (with seasonality removed):

```
print(MKslope(gw.1978))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -6.9700  0.6877  0.9050  0.9132  1.1134  7.1400     324
## [1] 0.905

coefficients(m.gw.nonseas)["time"]

##      time
## 0.9035588
```



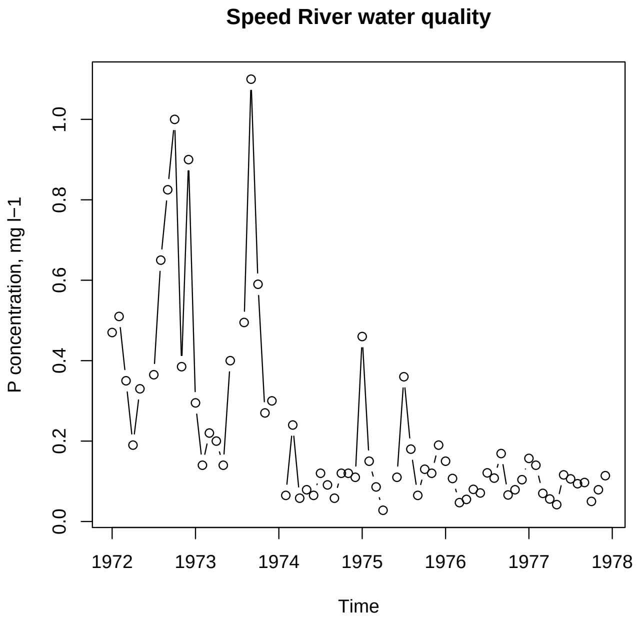
These are very close in this case.

### 4.3.3 A more difficult example

The trend is not always so obvious, and the deviations from IIND residuals much stronger, in some time series. For example, the `Kendall` package includes the sample dataset `GuelphP`, a monthly time series of phosphorous (P) concentrations in mg l-1, Speed River, Guelph, Ontario, January 1972 through January 1977.

**Task 55** : Load this dataset and plot as a time series. •

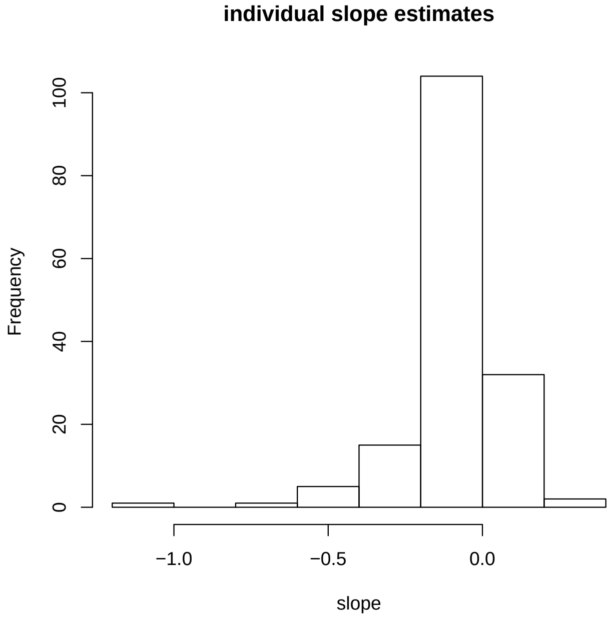
```
data(GuelphP)
plot(GuelphP, type="b", ylab="P concentration, mg l-1",
     main="Speed River water quality")
```



**Q52 :** *Describe this time series qualitatively (in words). Does there seem to be a linear trend?* *Jump to A52* •

**Task 56 :** Test for a trend and, if present, estimate its slope. •

```
SeasonalMannKendall(GuelphP)

## tau = -0.562, 2-sided pvalue =1.758e-07

print(guelphP.b <- MKslope(GuelphP))

##       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's 
## -1.042000 -0.135125 -0.056333 -0.090538 -0.002667  0.275000        20 
## [1] -0.05633333
```



**Q53** : *Is there a trend? If so, what is its slope? Is this slope meaningful?*
*Jump to A53* •

## 4.4 Autoregressive integrated moving-average (ARIMA) models

Box et al. [5] developed an approach to time series analysis known as ARIMA ("autoregressive (AR) integrated (I) moving averages (MA)"), which is especially useful for forecasting. We first examine the "AR" and "MA" aspects, and then add the "I".

### 4.4.1 Autoregressive (AR) models

The simplest model form is the **autoregressive** (AR) model. Here the values in the series are correlated to some number of immediately preceding values. The strength of the correlation, relative to the white noise, gives the **continuity** of the series.

The AR process is defined such that each $Y_t$ in the sequence is computed from some set of previous values $Y_s, s < t$, plus white noise $Z_t$. This white noise is independently and identicall distributed (IID) at each time step.

This model implies an underlying process with no trend, where the the value at one time point is partly retained at the next in an AR(1) process; this can be considered *inertia* in the process. The autocorrelation is not perfect, this allows *white noise*, i.e., completely random processes, to alter the next value(s).

To simplify computations, the series is **centred** by subtracting the overall mean $\mu$ and considering the differences:

$$(Y_t - \mu) = \sum_{l=1}^{p} \alpha_l (Y_{t-l} - \mu) + Z_t \tag{8}$$

where the $Z_t$ form a **white noise** (purely random) sequence $\{Z_t\}$.

The **order** $p$ of the process controls the number of previous values of $Y_t$ considered; the magnitude of $\alpha_l$ is the degree of autocorrelation with the $l$th previous value. The new $(Y_t - \mu)$ are computed from previous values of $(Y_{t-l} - \mu)$ up to the order, plus some new IID white noise $\{Z_t\}$.

**AR(1) models** This model only considers the immediately preceding value, along with white noise:

$$(Y_t - \mu) = \alpha_1 (Y_{t-1} - \mu) + Z_t \tag{9}$$

This has the same form as a linear regression, and the single parameter $\alpha_1$ can be computed in the same way.

This series is sometimes called a **red noise** process, because the white noise represented by the sequence $\{Z_t\}$ has the high-frequency variations ("blue", by analogy with the light spectrum) smoothed out by the autocorrelation. The low-frequency ("red") random variations are preserved.

In §3.5 we saw that the remainders for the groundwater levels of well 1 had no partial autocorrelations, after the first order was accounted for. This indicates an AR(1) model.

Note that the mean of the remainders should be zero, if we've accounted for the trend and cycles; in this case it is very close:

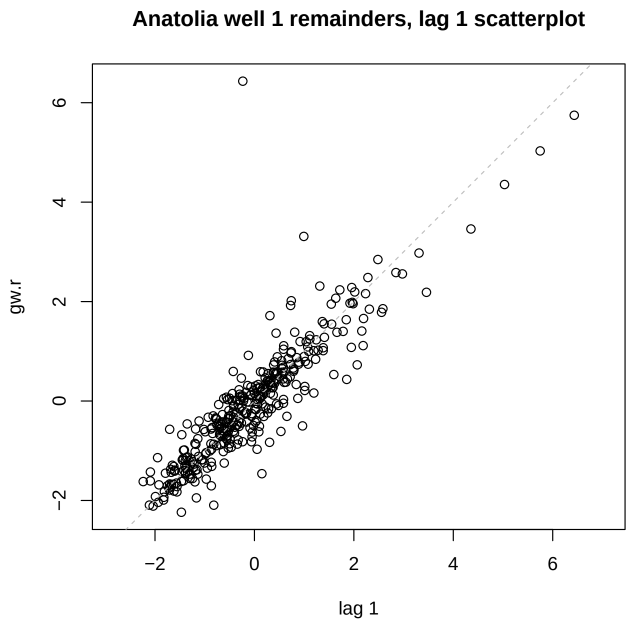
```
mean(gw.r)

## [1] -0.06420781
```

---

**Task 57 :** Estimate the autocorrelation parameter for the remainders for the groundwater levels of well 1. •

First, examine the autocorrelation as a scatterplot; this was already done in §3.4, using the `lag.plot` function:

```
lag.plot(gw.r, lags=1, main="Anatolia well 1 remainders, lag 1 scatterplot")
```



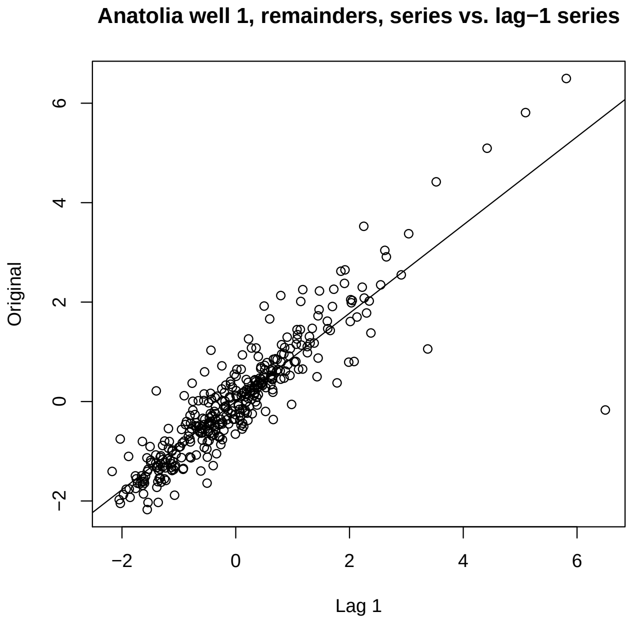
Clearly a linear relation between the series of remainders and its first lagged series is justified. We compute this relation first with the standard `lm` method; note we must relate an observation to the **preceding** observation (time flows in one direction!). We must first produce a time-series offset by one month. We saw in §3.4 that the `lag` function lags the time series but does not shift it; here we need a shift in order to relate the previous month's level to the current month's level. The shift is effected by subscripting, using the `[]` operator.

We first construct the two series, subtracting in each case the mean:

```
gw.r.0 <- gw.r[2:(length(gw.r)-1)] - mean(gw.r)
gw.r.1 <- lag(gw.r,1)[1:(length(gw.r)-2)] - mean(gw.r)
```

```
plot(gw.r.0 ~ gw.r.1, xlab="Lag 1",ylab="Original")
title(main="Anatolia well 1, remainders, series vs. lag-1 series")
m.lag1 <- lm(gw.r.0 ~ gw.r.1)
summary(m.lag1)

##
## Call:
## lm(formula = gw.r.0 ~ gw.r.1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9367 -0.2370 -0.0304  0.2109  1.5253
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.001855   0.029559   0.063     0.95
## gw.r.1      0.887310   0.024360  36.424   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5593 on 356 degrees of freedom
## Multiple R-squared:  0.7884,Adjusted R-squared:  0.7878
## F-statistic:  1327 on 1 and 356 DF,  p-value: < 2.2e-16

abline(m.lag1)
```

```
(alpha.1 <- cor(gw.r.0,gw.r.1))

## [1] 0.8879416
```



---

**Q54** : *How much of the variation in the remainder of groundwater level (after accounting for trend and seasonality) is accounted for by knowledge of the remainder at the previous lag?* *Jump to A54* •

Note that the comparable figure for the uncorrected series is much higher, because of the inherent continuity within the annual cycle:

```
cor(gw[2:(length(gw)-1)] - mean(gw),
    lag(gw,1)[1:(length(gw)-2)] - mean(gw))

## [1] 0.9934396
```

The correlation coefficient should be the first autocorrelation calculated with the `acf` "autocorrelation" function, as shown in §3.4.

```
acf(gw.r, lag.max=1, plot=F)$acf[2,,]

## [1] 0.8868833
```

> **Note:** The slight difference between this estimate 0.8869 and the estimate directly from linear correlation 0.8879 may be due to how the two computations deal with the ends of the series.

Finally, the **innovation variance** is the variance of the white noise of Eq. 9. This is computed as [20, §8.3.1]:

$$\sigma_Z^2 = (1 - \alpha^2)\sigma_Y^2 \qquad (10)$$

where $\sigma_Y^2$ is the variance of the time series. That is, the noise is reduced from the observed noise in the series by the autocorrelation – that much of the noise is accounted for by the model. This illustrates the “red shift” mentioned above. For sampled time series of length $n$, the series variance $\sigma_Y^2$ is estimated from the sample variance $s_Y^2$, the true correlation $\alpha$ is estimated as $\hat{\alpha}$ and the noise must be corrected for bias:

$$s_Z^2 = \frac{n-1}{n-2}(1-\hat{\alpha}^2)s_Y^2 \tag{11}$$

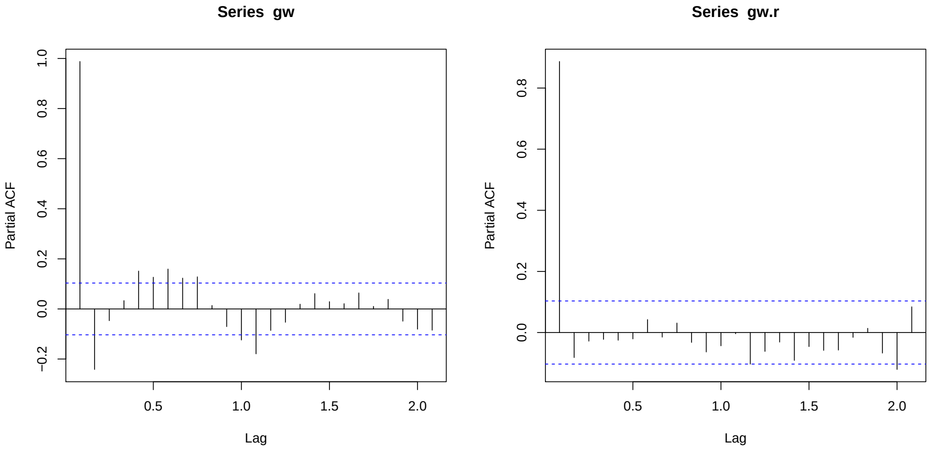
In the current case:

```
var(gw.r)

## [1] 1.469715

(var.ar.1 <- (length(gw.r)-1)/(length(gw.r)-2) * (1 - alpha.1^2) * var(gw.r))

## [1] 0.3118009
```

We will return to this example and simulate an AR(1) series with this these fitted parameters in §8.

**AR(2), AR(3) . . . models** In an AR(1) model the entire prior behaviour of a series is given by the previous value and the one autocorrelation coefficient; this is known as the **first-order Markov** process. For the example series of remainders of groundwater level, we saw from the partial autocorrelations that this is a reasonable assumption. However, it is possible that the current state is also influenced by earlier states, other than the immediately preceding one; this is the case for the original series of groundwater levels:

```
par(mfrow=c(1,2))
pacf(gw)
pacf(gw.r)
par(mfrow=c(1,1))
pacf(gw, lag.max=5, plot=F)$acf

## , , 1
##
##             [,1]
## [1,]  0.98782910
## [2,] -0.24150277
## [3,] -0.04679133
## [4,]  0.03321400
## [5,]  0.15111996

pacf(gw.r, lag.max=5, plot=F)$acf

## , , 1
##
##             [,1]
## [1,]  0.88688330
## [2,] -0.08180894
## [3,] -0.02809817
## [4,] -0.02235775
## [5,] -0.02513076
```



Once the preceding lag is taken into account (high positive correlation, high continuity) we see that the second lag is **negatively** correlated (lack of continuity). Even for the remainders, this is the case but not quite at the level of significance.

**Q55 :** *What is the physical interpretation of this result? Jump to A55* •

For higher-order AR models, the parameters are fit simultaneously; the most common method is with the *Yule-Walker* equations, which relate the parameters to the sample autocorrelations. For the AR(2) model these are:

$$\begin{aligned} r_1 &= \hat{\alpha}_1 + \hat{\alpha}_2 r_1 \\ r_2 &= \hat{\alpha}_1 r_2 + \hat{\alpha}_2 \end{aligned}$$

which can be solved as simultaneous equations. These generalize to any higher order.

The red-noise variance is then estimated as:

$$s_Z^2(2) = (1 - \hat{\alpha}_2^2)\frac{n-1}{n-2}(1 - r_1^2)s_Y^2 \tag{12}$$

where $r_1$ is the sample correlation at lag 1. Thus the original white noise is reduced yet further, as the degree of the AR model increases.

The `ar` function not only solves these equations, but also solves them for all orders from AR(1), AR(2) . . . until the higher-order fit is not better, as judged by the AIC.

**Task 58 :** Fit AR(n) models to the remainder series. •

Here we show the optional `method` argument; the default `"yule-walker"` is used. The red noise variance is printed with the solution; it is also stored with the model object.

```
(ar.gw.r <- ar(gw.r, method="yule-walker"))

##
## Call:
## ar(x = gw.r, method = "yule-walker")
##
## Coefficients:
##       1        2
##  0.9594  -0.0818
##
## Order selected 2  sigma^2 estimated as  0.3133

ar.gw.r$var.pred

## [1] 0.3133392

ar.gw.r$ar

## [1]  0.95943829 -0.08180894
```

For comparison, we re-fit the AR(1) series also, using the `order.max` argument to force `ar` to only fit this order.

```
(ar.gw.r.1 <- ar(gw.r, order.max=1))

##
## Call:
## ar(x = gw.r, order.max = 1)
##
## Coefficients:
##      1
## 0.8869
##
## Order selected 1  sigma^2 estimated as  0.3146

ar.gw.r.1$ar

## [1] 0.8868833
```

---

**Q56** : *What order of AR model is selected by the* `ar` *function? How well is the remainder series modelled by an AR(2) process? How much improvement is this over the AR(1) process?* *Jump to A56* •

> **Note:** An AR(1) model is always stationary, but higher-order models may not be. The parameters must jointly satisfy some constraints. See texts (e.g., Wilks [20, §8.3.2]) for details. The `ar` function reports non-stationarity in the fit.

We will return to this example and simulate an AR(2) series with these fitted parameters in §8.

### 4.4.2 Moving average (MA) models

The MA process is simply a linear filter of some previous white noise, plus the white noise for the current time:

$$Y_t = \sum_{j=1}^{q} \beta_j Z_{t-j} + Z_t \tag{13}$$

The $\beta_j$ weight the relative contributions of the previous values. The time series results from random noise, which (if any $\beta_j \neq 0$) can "drift" into an apparent trend, which in fact is the result of the stochastic process, not a true trend. Thus, MA models are often used to model apparent trends.

### 4.4.3 ARMA models

**Autoregressive moving-average** (ARMA) models combine the AR and MA explained above, so that the observed time series is the result of these two types of random processes.

The theory behind ARMA models is that a time series, written as a sequence of values over time $\{Y_t\}$, can be considered as the contribution of four components:

1. An overall **mean level** $\mu$; this can be subtracted from the series, leaving a series centred on zero;
2. An **autoregressive** (AR) component, where values in the series are correlated to some number of immediately preceding values;
3. A **moving average** (MA), where values in the series are some linear combination of earlier values of white noise, but with no correlation between successive values of this noise;
4. **white noise**, a random component with zero mean and some constant variance, conventionally represented as $\{Z_t\}$.

### 4.4.4 ARIMA models

The "I" in ARIMA stands for "integrated". These are ARMA models with an additional element: the degree of **differencing** applied to the series before ARMA analysis. ARMIMA models are conventionally specified with three components $(p, d, q)$, which are:

1. $p$: the AR order;
2. $d$: the degree of differencing; and
3. $q$: the MA order.

Differencing is applied so that the series is **second-order stationary**, i.e., the expected value and covariance do not depend on the position in the series.

ARIMA models are fit with the `arima` function. This requires at least two arguments: the series and the order. To illustrate, we re-fit the AR(2) model of the well level residuals (§4.4.1) with `arima`. The order is $(2, 0, 0)$:

```
(arima.gw.r <- arima(gw.r, order=c(2,0,0)))

##
## Call:
## arima(x = gw.r, order = c(2, 0, 0))
```

```
##
## Coefficients:
##          ar1      ar2  intercept
##       0.9624  -0.0851    -0.0925
## s.e.  0.0524   0.0525     0.2341
##
## sigma^2 estimated as 0.3078:  log likelihood = -299.48,  aic = 606.95
```

---

**Q57** : *Does the ARIMA(2,0,0) fit give the same coefficients as the AR(2) fit?* *Jump to A57* •

The coefficients for model fit by `ar` are in field `ar`; for a model fit by `arima` in field `coef`:

```
ar.gw.r$ar

## [1]  0.95943829 -0.08180894

arima.gw.r$coef

##         ar1         ar2   intercept
##  0.96236062 -0.08513407 -0.09246881
```

We will examine model fitting in detail just below.

### 4.4.5 Periodic autoregressive (PAR) models

To fit an AR model, we had to establish stationarity. Clearly a cyclic series is not first-order stationary, yet it seems somehow unnatural to remove the cycle, model the series, and add the cycle back in. With **periodic autoregressive** (PAR) models, both are modelled together.

The PAR process is defined like the AR process, with with a fluctuating average $\mu_\tau$ instead of an overall average $\mu$. The values $Y$ are indexed by the cycle number $\eta$ and the position in the cycle $\tau$, i.e., $Y_{\eta,\tau}$. The autoregressive parameter is also indexed by the position in the cycle, as well as the order $l$, i.e., $\alpha_{l,\tau}$. Finally, the white noise depends on the position in the cycle as well: $Z_{\eta,\tau}$. Putting these together, Equation 9 is replaced with:

$$(Y_{\eta,\tau} - \mu_\tau) = \sum_{l=1}^{p} \alpha_{l,\tau}(Y_{\eta,\tau-l} - \mu_\tau) + Z_{\eta,\tau} \tag{14}$$

> **Note:** Note that a PAR model can not have any trend, just the cyclic component. If there is an overall trend it must be removed before modelling.

These models are widely-used for modelling monthly rainfall series or streamflows. They are appropriate when the periodic component is much larger than the white noise.

These are fit with the `arima` function, by specifying the optional `seasonal` argument, which, like the ARIMA `order`, is a list of the three components $(p, d, q)$.

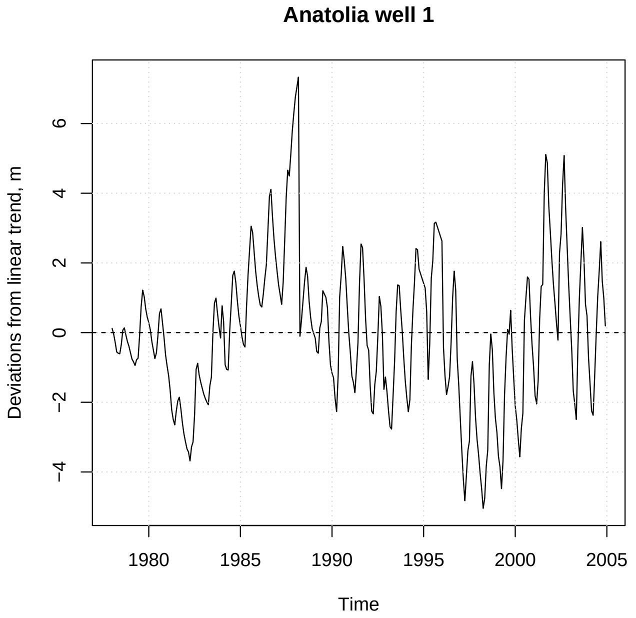
**Task 59** : Fit a PAR model to the de-trended time series of groundwater levels of Anatolia well 1 since 1978. •

Recall that the behaviour before 1978 was qualitatively different than that since; we suspect that extraction began in 1978. In §4.2 a linear trend was established for that period:

```
gw.f.78 <- subset(gw.f, gw.f$year > 1977)
coef(m.gw.78 <- lm(gw ~ time, data=gw.f.78))

##   (Intercept)          time
## -1763.3607555     0.9067699
```

We subtract the fits from this model from each observation (using the `fitted` extractor function on the linear model), to get the de-trended series. We also need to extract the time-series window, using `window`.

```
gw.1978 <- window(gw, c(1978,1), c(2004,12))
str(gw.1978)

##  Time-Series [1:324] from 1978 to 2005: 30.4 30.3 30.1 29.9 29.9 ...

str(fitted(m.gw.78))

##  Named num [1:324] 30.2 30.3 30.4 30.5 30.5 ...
##  - attr(*, "names")= chr [1:324] "37" "38" "39" "40" ...

str(gw.1978.0 <- gw.1978 - fitted(m.gw.78))

##  Time-Series [1:324] from 1978 to 2005: 0.1199 -0.0457 -0.2813 -0.5568 -0.5924 ...
##  - attr(*, "names")= chr [1:324] "37" "38" "39" "40" ...

plot(gw.1978.0, ylab="Deviations from linear trend, m",
     main="Anatolia well 1", pch=20)
grid()
abline(h=0, lty=2)
```



We now fit a PAR model, with AR(2) for the non-seasonal part (as revealed by our previous analysis) and different AR orders for the seasonal part. A seasonal order of $(0, 0, 0)$ corresponds to the same cycle each year. Higher-order AR represent autocorrelation of cycles year-to-year; e.g., a high-amplitude cycle tends to be preceded and followed by a similar amplitude

> **Note:** The frequency can be specified as part of the `seasonal` argument, but defaults to the known frequency of the series, as given by the `frequency` function.

```
(par.gw.1978 <- arima(gw.1978.0, order=c(2,0,0), seasonal=c(0,0,0)))

##
## Call:
## arima(x = gw.1978.0, order = c(2, 0, 0), seasonal = c(0, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       1.3306  -0.4585    -0.0046
## s.e.  0.0492   0.0492     0.3326
##
## sigma^2 estimated as 0.5983:  log likelihood = -377.66,  aic = 763.32

(par.gw.1978 <- arima(gw.1978.0, order=c(2,0,0), seasonal=c(1,0,0)))

##
## Call:
## arima(x = gw.1978.0, order = c(2, 0, 0), seasonal = c(1, 0, 0))
##
## Coefficients:
##          ar1      ar2    sar1  intercept
##       1.1142  -0.2404  0.4272    -0.0026
## s.e.  0.0639   0.0640  0.0593     0.5293
##
## sigma^2 estimated as 0.5159:  log likelihood = -354.63,  aic = 719.27

(par.gw.1978 <- arima(gw.1978.0, order=c(2,0,0), seasonal=c(2,0,0)))

##
## Call:
## arima(x = gw.1978.0, order = c(2, 0, 0), seasonal = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2    sar1    sar2  intercept
##       1.0657  -0.1691  0.3604  0.2144    -0.0118
## s.e.  0.0618   0.0635  0.0562  0.0571     0.8162
##
## sigma^2 estimated as 0.4923:  log likelihood = -347.92,  aic = 707.84
```

The fit of the models, accounting for number of parameters, is given by the AIC; lower is better.

---

**Q58** : *How much does modelling the seasonal component improve the fit? Which degree of autoregression among seasons is indicated?* *Jump to A58* •

## 4.5 Modelling with ARIMA models

ARIMA models were developed primarily for forecasting, so modelling an observed time series with ARIMA models is intended to fit a good empirical model that can be used for this purpose. Interpretation in terms of underlying processes is not straightforward.

Modelling with an ARIMA model has three stages:

1. Model identification;
2. Parameter estimation;
3. Diagnostic checking of model suitability.

These stages are iterated until the model is deemed “suitable”; then the model is ready to use for process interpretation or forecasting.

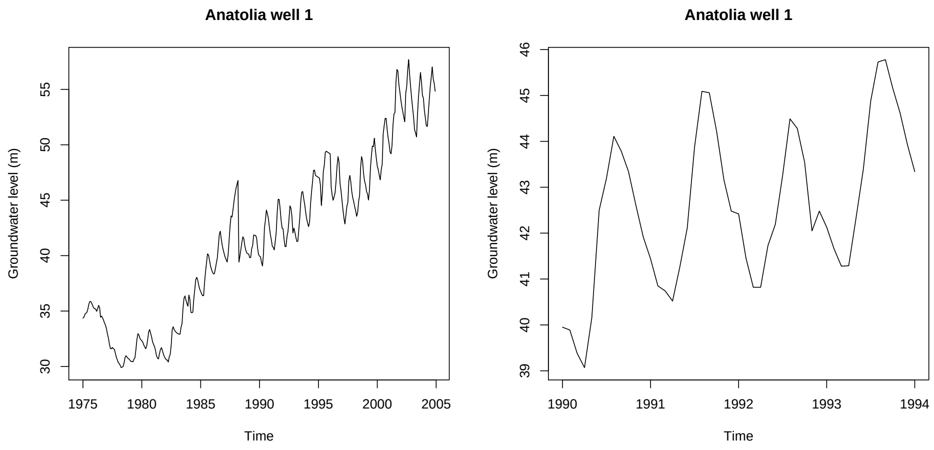
### 4.5.1 Checking for stationarity

The first step in model identification is to determine if any differencing is needed. There are two indications that a series is not stationary:

- A time series that appears to have different overall levels or degrees of autocorrelation in different sections of the series;
- A correlogram (ACF) that does not decay to zero.

**Task 60** : Plot the groundwater time series and evaluate its stationarity; also zoom in on a three-year window to see the fine structure. •

```
par(mfrow=c(1,2))
plot(gw, main="Anatolia well 1",
     ylab="Groundwater level (m)")
plot(window(gw, 1990, 1994), main="Anatolia well 1",
     ylab="Groundwater level (m)")
par(mfrow=c(1,2))
```

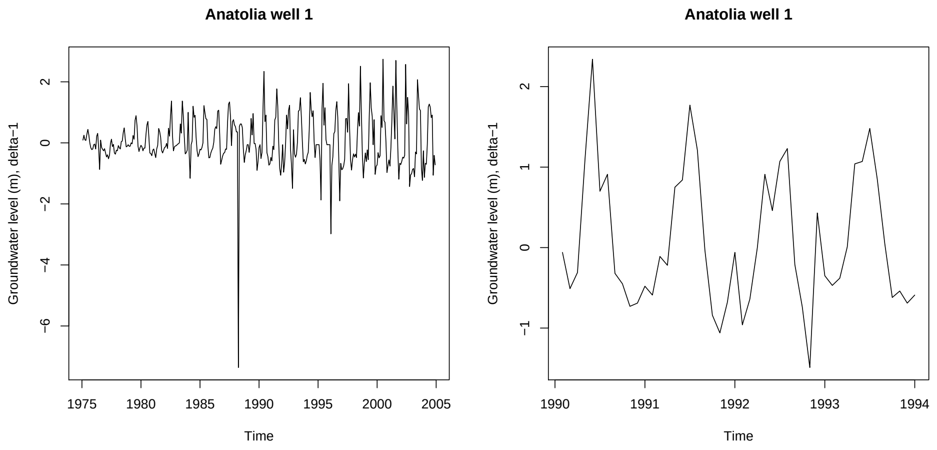


**Q59** : *Does there appear to be a trend and/or cycle (i.e., non-constant expected value?) Does the variance appear to be constant?* *Jump to A59* •

A trend can be removed with one difference.

**Task 61** : Plot the first difference of the groundwater time series and evaluate its stationarity; also zoom in on a three-year window to see the fine structure. •

```
par(mfrow=c(1,2))
plot(diff(gw), main="Anatolia well 1",
     ylab="Groundwater level (m), delta-1")
plot(diff(window(gw, 1990, 1994)), main="Anatolia well 1",
     ylab="Groundwater level (m), delta-1")
par(mfrow=c(1,2))
```

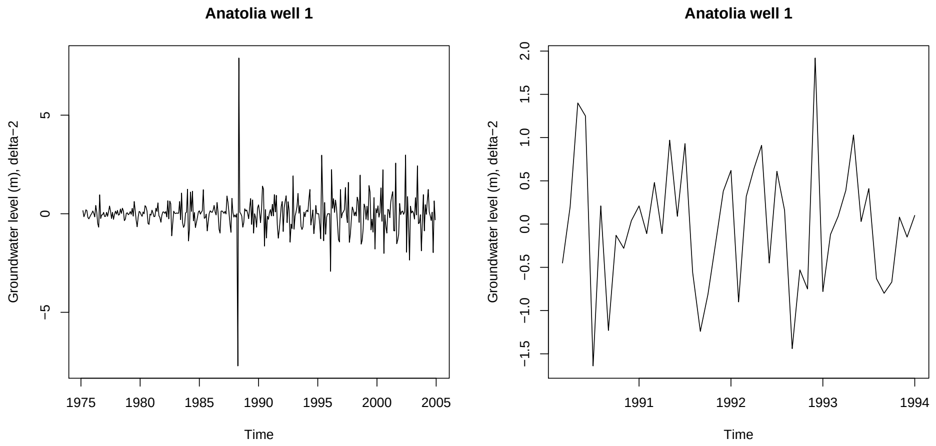


**Q60** : *Does there appear to be a trend and/or cycle (i.e., non-constant expected value?) Does the variance appear to be constant?* *Jump to A60* •

We difference the series once more:

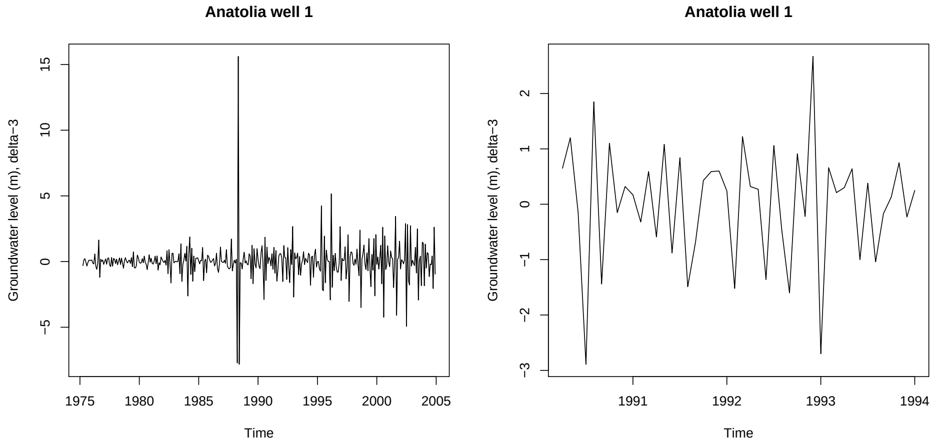
**Task 62** : Plot the second difference of the groundwater time series and evaluate its stationarity. •

```
par(mfrow=c(1,2))
plot(diff(diff(gw)), main="Anatolia well 1",
     ylab="Groundwater level (m), delta-2")
plot(diff(diff(window(gw, 1990, 1994))), main="Anatolia well 1",
     ylab="Groundwater level (m), delta-2")
par(mfrow=c(1,2))
```



**Task 63** : Repeat, with the third differences. •

```
par(mfrow=c(1,2))
plot(diff(diff(diff(gw))), main="Anatolia well 1",
     ylab="Groundwater level (m), delta-3")
plot(diff(diff(diff(window(gw, 1990, 1994)))),
     main="Anatolia well 1",
     ylab="Groundwater level (m), delta-3")
par(mfrow=c(1,2))
```

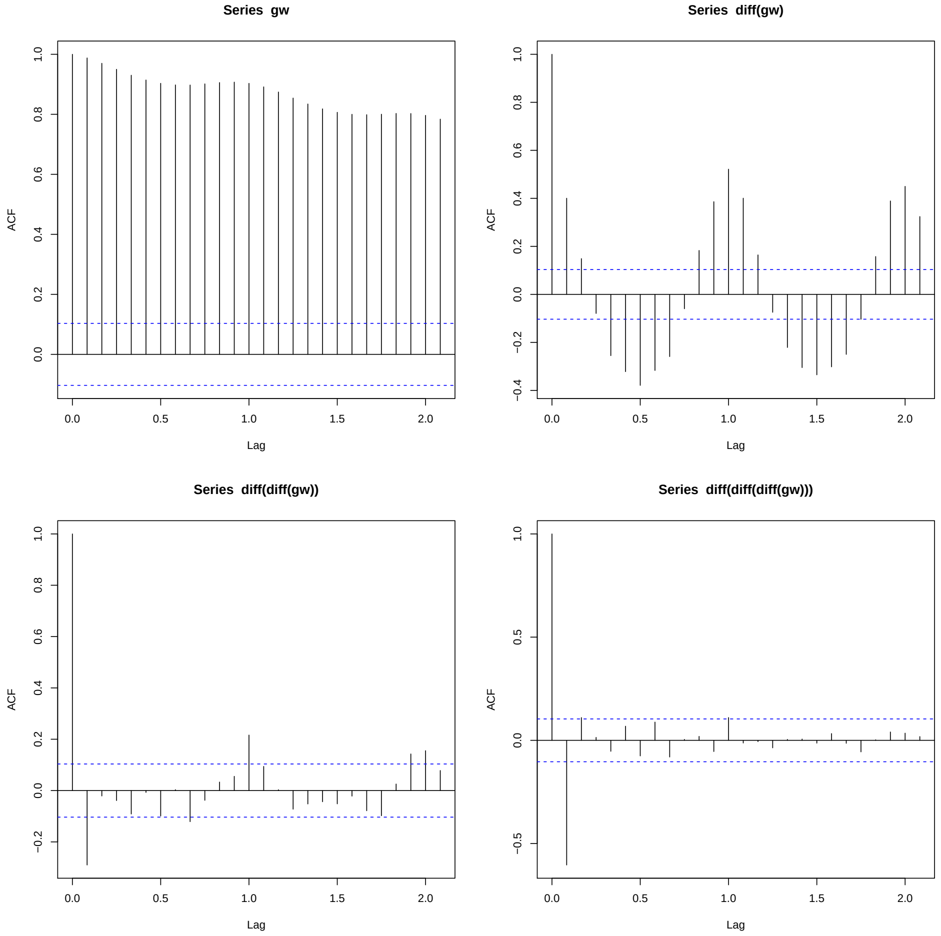


There seems to be little change between the second and third differences.

Another way to look at the stationary is with the autocorrelation function plotted by `acf`.

**Task 64** : Plot the autocorrelation functions for the original time series and the first three differences. •

```
par(mfrow=c(2,2))
acf(gw)
acf(diff(gw))
acf(diff(diff(gw)))
acf(diff(diff(diff(gw))))
par(mfrow=c(1,1))
```



**Q61** : *Describe the behaviour of the ACF with increasing differencing.* *Jump to A61* •

In conclusion, two differencing operations seem to result in a more or less second-order stationary time series.

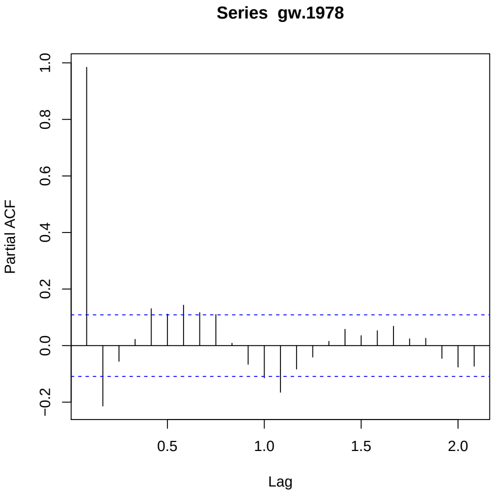
### 4.5.2 Determining the AR degress

The next step in model identification is to examine the autocorrelations and partial autocorrelations using `acf` and `pacf`, respectively.

If the process is AR($p$), the partial autocorrelation is zero at lag $\geq p + 1$. So, the sample partial autocorrelation plot is examined to identify the order: find the lag where the partial autocorrelations for all higher lags are not significantly different from zero.

We can see this nicely for the groundwater data since 1978.

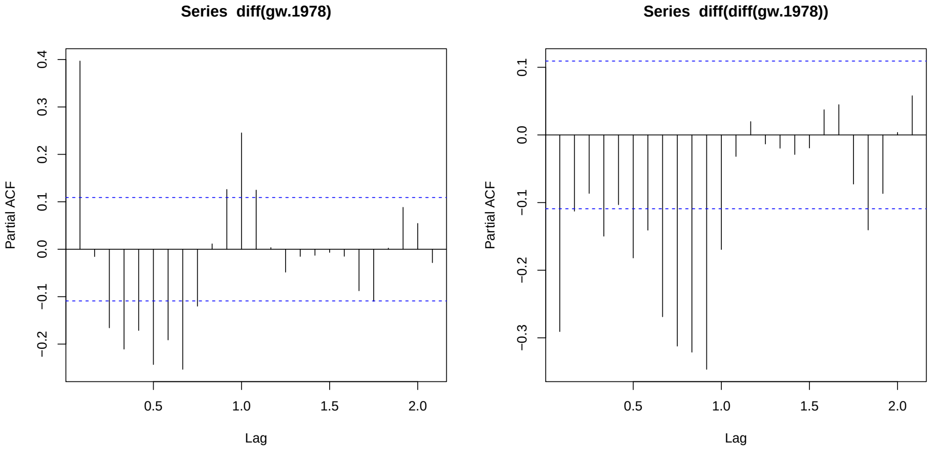
```
pacf(gw.1978)
```



Here the significant partial correlations are at 1, 2, 5 . . . 9, 12 and 13 months (“lag” on this plot refers to the 12-month cycle). So we could try to fit an AR(13) model.

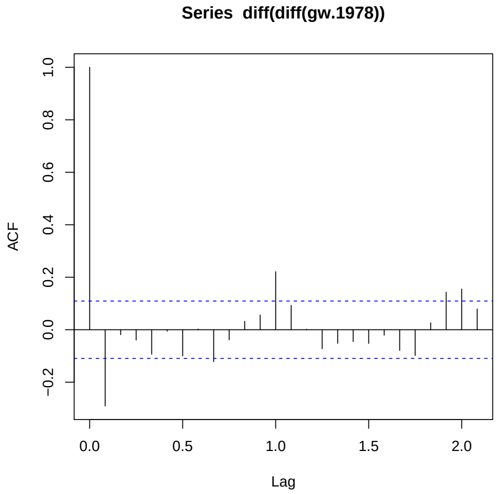
However, since we’ve already determined that two differences are needed for stationarity, we evaluate the PACF of the differenced series:

```
par(mfrow=c(1,2))
pacf(diff(gw.1978))
pacf(diff(diff(gw.1978)))
par(mfrow=c(1,1))
```



These also show partial autocorrelation to 13 months, although the absolute correlation coefficients decrease with increasing differencing.

```
acf(diff(diff(gw.1978)))
```



The second step is to estimate the model parameters, using the `arima` function. This must be supplied with three parameters, which specify the model type; these are conventionally known as $p$ (the AR order), $d$ (the degree of differencing), and $q$ (the MA order), as explained above. ARIMA models may also declare a periodic (also called **seasonal**) component, with the same parameters.

**Task 65 :** Calibrate an ARIMA model for the groundwater level. •

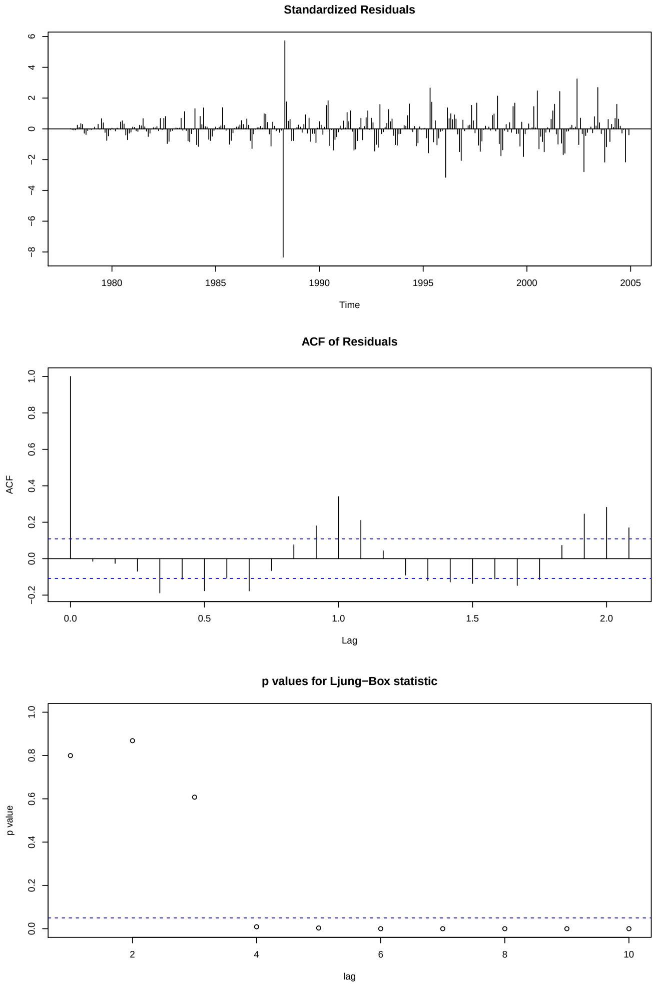
```
(m.ar <- arima(gw.1978, order=c(13,2,0)))

##
## Call:
## arima(x = gw.1978, order = c(13, 2, 0))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7
##       -0.8569  -0.8617  -0.8759  -0.9057  -0.8682  -0.8971  -0.8651
## s.e.   0.0556   0.0719   0.0801   0.0834   0.0839   0.0820   0.0828
##           ar8      ar9     ar10     ar11     ar12     ar13
##       -0.9231  -0.8893  -0.7707  -0.5805  -0.2637  -0.0708
## s.e.   0.0816   0.0835   0.0830   0.0798   0.0715   0.0553
##
## sigma^2 estimated as 0.4767:  log likelihood = -340.38,  aic = 708.76
```

The third step is model checking; the `tsdiag` function produces three diagnostic plots for ARIMA models:

1. standardized residuals (should show no pattern with time);
2. autocorrelation function (ACF) of residuals (should have no significant autocorrelation);
3. the Ljung-Box statistic for the null hypothesis of independence in the time series of residuals.

We can see the effect of a poor model fit by under-fitting the groundwater levels with an AR model:

```
m.ar <- arima(gw.1978, c(3,2,0))
tsdiag(m.ar)
```

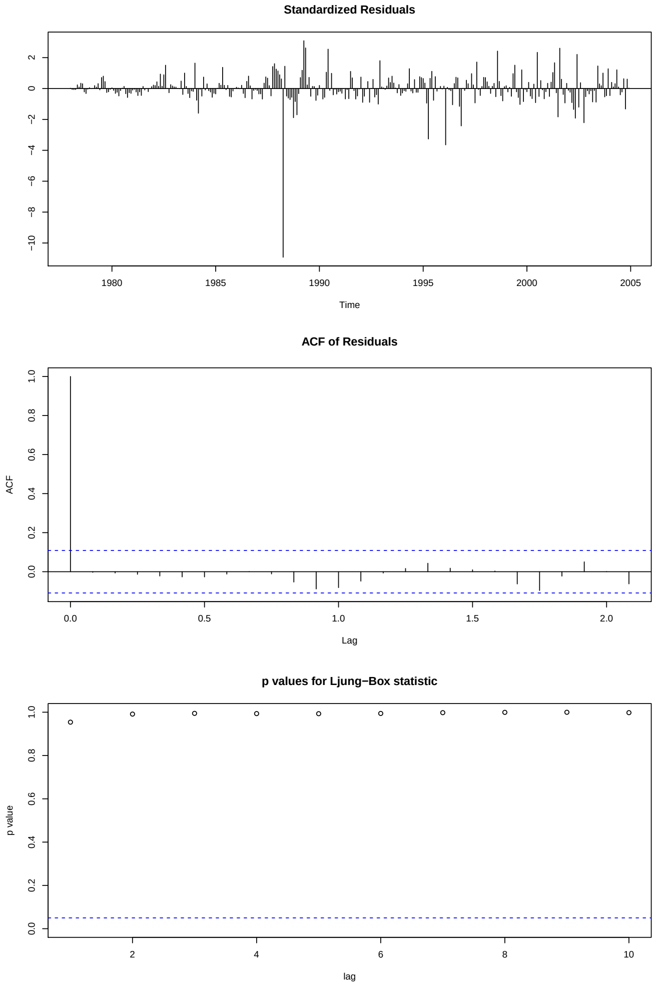

Standardized Residuals

ACF of Residuals

p values for Ljung−Box statistic

Notice the significant autocorrelation of the residuals at 4, 6, 12 and 13 months, and the low p-values of the Ljung-Box statistic after lag 4; this means that we can not reject the null hypothesis of serial dependence at these lags. At a more appropriate order the diagnostics are satisfactory:

```
m.ar <- arima(gw.1978, c(13,2,0))
tsdiag(m.ar)
```



These diagnostics look very good.

## 4.6 Predicting with ARIMA models

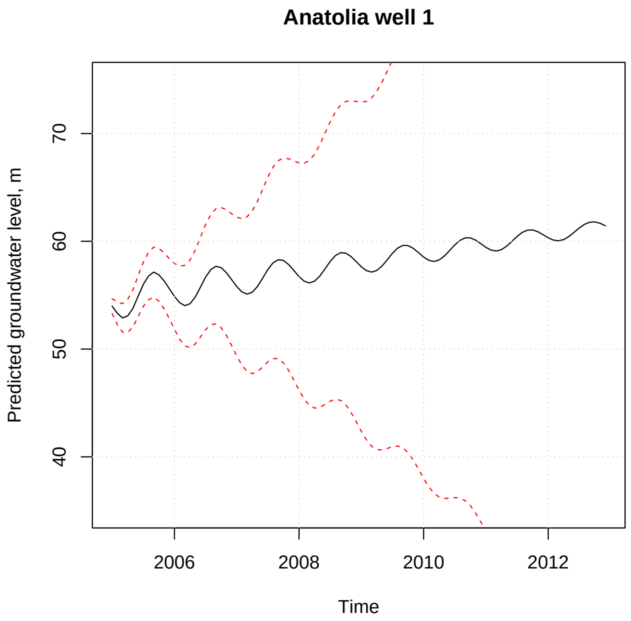
Finally, ARIMA models can be used to predict.

**Task 66 :** Predict groundwater levels from 2005 through 2012, using the AR(13) model just fit. •

For ARIMA models, the generic `predict` method specializes to the `predict.Arima` function; this returns both the predictions and their standard errors. The argument `n.ahead` gives the number of prediction points, here months:

```
p.ar <-predict(m.ar, n.ahead=12*(2013-2005))
str(p.ar)

## List of 2
##  $ pred: Time-Series [1:96] from 2005 to 2013: 54 53.3 52.9 53.1 53.8 ...
##  $ se  : Time-Series [1:96] from 2005 to 2013: 0.69 1.05 1.32 1.54 1.72 ...

plot(p.ar$pred, ylim=c(35, 75), ylab="Predicted groundwater level, m",
     main="Anatolia well 1")
```

```
lines(p.ar$pred+p.ar$se, col="red", lty=2)
lines(p.ar$pred-p.ar$se, col="red", lty=2)
grid()
```



**Q62** : *What happens to the prediction as the time forward from the known series increases? What happens to the confidence intervals? As a groundwater manager, how far ahead would you be willing to use this predicted series?* *Jump to A62* •

## 4.7 Answers

*A41 : (1) There is a long-term **trend**, to a slightly shallower level until 1980 and then steadily to a deeper level. The slope of the trend (groundwater level vs. time interval) varies a bit over the 1980- 2005, which may reflect rainfall differences. The trend is most likely due to increased extraction for irrigation, since the quantity of far exceeds annual rainfall; which we see in . . . (2) There is a seasonal cycle in groundwater level, due to recharge in the winter (rains) and extraction in the summer; however the magnitude of the fluctuation appears to increase from about 1983–1995 and has since stabilized. The increasing fluctuation may be caused by more extraction but is also balanced by more rainfall. (3) The remainder is of similar magnitude to the annual cycle (±2m) and is strongly auto-correlated; the explanation for this is unclear. The very large remainder in 1988 was discussed in §2.1.* *Return to Q41* •

*A42 : The proportion of variation explained is given by the adjusted $R^2$, here 0.89. The overall trend is fairly well-explained by the line. The average annual* *increase is given by the slope coefficient, 0.813* *Return to Q42* •

---

**A43** : *The estimate of the AR(1) temporal correlation returned by* `gls` *is 0.9571, quite close to our original estimate 0.9878.* *Return to Q43* •

---

**A44** : *The slope is somewhat shallower: GLS 0.7751 vs. OLS 0.813.* *Return to Q44* •

---

**A45** : *Yes, the proportion of variation explained has increased to 0.9; the ANOVA shows a highly significant improvement in fit. The cubic adjusts somewhat to the initial part of the series.* *Return to Q45* •

---

**A46** : *The average annual increase has changed considerably, increasing from 0.775 for the whole series to 0.907 for the shorter series. The steeper slope ignores the earliest part of the series, when the process was different, and so seems to better reflect current conditions and is preferable for predicting in the short term.* *Return to Q46* •

---

**A47** : *The trend can not continue forever – for example, at a certain point the groundwater level will be too deep to pump economically and will stabilize at that level. Also, future rainfall patterns and future demand for irrigation water may change.* *Return to Q47* •

---

**A48** : *The variance explained increases 2.51% compared to the fit to the original time series, because there is less overall variability.* *Return to Q48* •

---

**A49** : *The slope of the trend is almost identical; the fit without the seasonal component predicts 3.21 mm less drawdown per year.* *Return to Q49* •

---

**A50** : *There are definite discrepencies. First, there is clear periodicity in the residuals vs. fitted values: high (positive) residuals around fitted values of 32, 48, 46, and 52 m and low (negative) at 34, 44, 48 m. Second, the highest residuals are far too positive (i.e., very strong underprediction); this is the year 1988 anomaly.* *Return to Q50* •

---

**A51** : *Effectively zero. There is certainly a trend.* *Return to Q51* •

---

**A52** : *The series from 1972 – 1974 has large fluctuations and high concentrations; after 1974 these are both much lower, except for some spikes in 1975. There seems to be a seasonal component. Some observations are missing. There is no linear trend, instead there seems to be a discontinuity in 1974,* *from high to low concentrations.* Return to Q52 •

---

**A53** : *The probability that the observed trend, estimated as -0.056 (mg l-1) yr-1, is due to sampling variation is very small, about 1 in* $10^7$. Return to Q53 •

---

**A54** : *Proportionally, 0.788 of the variance is explained. This is a high degree of continuity, even after accounting for trend and cycle.* Return to Q54 •

---

**A55** : *After correcting for immediate continuity (since this is ground water, to be expected) the lag-2 level or remainder introduces a negative correction. That is, going two steps forward with a single AR(1) model would over-state the continuity, compared to an AR(2) model.* Return to Q55 •

---

**A56** : *An AR(2) series is selected; thus the lag-2 component can be modelled. The residual variance decreases slightly, from 0.3146 to 0.3133. Most of the variance reduction was from the original series to the AR(1) series; the AR(2) series is only a small improvement.* Return to Q56 •

---

**A57** : *No, there are slight differences. The AR(2) fit gives the two coefficients as 0.9594 and -0.0818; for the ARIMA(2,0,0) fit these are computed as 0.9624 and -0.0851* Return to Q57 •

---

**A58** : *The standard error of the best seasonal ARIMA fit is 0.4923, compare to 0.3133 for the non-seasonal model; this is somewhat higher, i.e., the fit is worse. But, the seasonal model also includes the seasonal component, which was removed from the time series fit with the non-seasonal model.* Return to Q58 •

---

**A59** : *There seems to be a clear trend to deeper levels since 1982; further the expected values seem to follow an annual cycle. Both are indications that this is not a first-order stationary series. The variance increases over time. So, this is not second-order stationary.* Return to Q59 •

---

**A60** : *There is no apparent trend, but there still seems to be an annual cycle. So, this is not first-order stationary. The variance increases over time. So, this is not second-order stationary.* Return to Q60 •

---

**A61** : *The ACF of the original series does not decay to zero; this indicates a trend. Once this is removed by the first difference, the ACF shows clear cyclic behaviour. This is largely removed by the second difference and completely by the third.* Return to Q61 •

---

**A62** : *The prediction becomes more uniform the further out we predict, i.e.,* *the cyclic behaviour is damped and approaches the overall linear trend. The upper and lower confidence limits become wider and also are damped; but they rapidly become much larger than the annual cycle. The prediction does not seem very useful to the manager; it seem more logical to use the average behaviour of the cycles in the known series and add it to the trend. Return to Q62* •

# 5 Intervention analysis

Time series may arise completely or mostly from natural processes (e.g., rainfall, temperature) but may also be influenced by human activities. These **interventions** may be one-time (e.g., damming a river) or pulsed (e.g., streamflow with controlled releases from dams). Hipel and McLeod [10, Ch. 19] is an excellent explanation of **intervention analysis**, which has two main uses:

1. Determining the effect of a known intervention (e.g., has a new sewage treatment plant improved water quality?);

2. Identifying probable unknown interventions (e.g., is there a new source of pollutants?).

Each of these may affect the series in several ways, known as **transfer functions**:

- An **immediate, single** effect (e.g., one day's streamflow is affected);
- An **immediate, permanent** effect (e.g., a pollutant is at a different level and maintains that level);
- An **asymptotic** effect, reaching a new equilibrium over time;
- An **immediate** effect, then **relaxing** to the original condition;
- Any of these may be **delayed**;
- Any of these may be **multiple**.

## 5.1 A known intervention

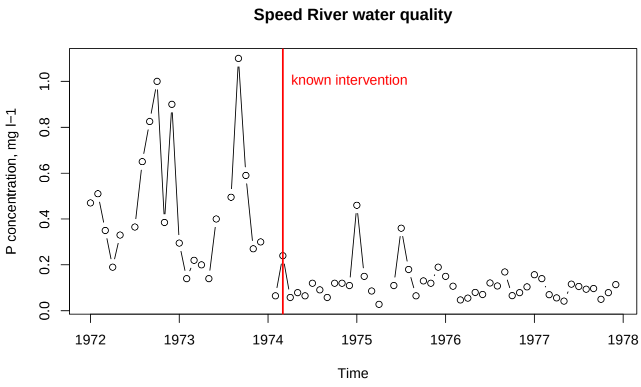
We return to the example of the phosphorous (P) concentrations briefly presented in §4.3.3. This is included in the `Kendall` package as a sample dataset `GuelphP`, a monthly time series of phosphorous (P) concentrations in mg l$^{-1}$, Speed River, Guelph, Ontario, January 1972 through January 1977. It is known [10, p. 655] that in February 1974 a new P treatment was introduced in the sewage treatment plant.

---

Task 67 : Load this dataset and plot as a time series, showing the known intervention. •

```
require(Kendall)
data(GuelphP)
str(GuelphP)
```

```
##  Time-Series [1:72] from 1972 to 1978: 0.47 0.51 0.35 0.19 0.33 NA 0.365 0.65 0.825 1 ...
##  - attr(*, "title")= chr "Phosphorous Data,Speed River,Guelph,1972.1-1977.1"

plot(GuelphP, type="b", ylab="P concentration, mg l-1",
     main="Speed River water quality")
abline(v=1974+2/12, col="red", lwd=2)
text(x=1974+2/12, y=1,"known intervention", col="red", pos=4)
```



Note the use of the `abline` function to add a line to the graph, specifying the `v` argument to specify a **v**ertical line at the indicated date.

**Q63 :** *Describe the effect of the new P treatment.* *Jump to A63* •

How can we reliably quantify this difference? One obvious way is to split the series and compare the means, medians, or variances.

**Task 68 :** Split the series at February 1974 and compare the means, medians, or variances. •

Since these statistics do not involve time series, and there are missing values, we convert to ordinary vectors with `as.vector` and remove the NA's with `na.omit`:

```
(guelph.1 <- na.omit(as.vector(window(GuelphP, start=NULL,
                                      end=1974+1/12))))

##  [1] 0.470 0.510 0.350 0.190 0.330 0.365 0.650 0.825 1.000 0.385 0.900
## [12] 0.295 0.140 0.220 0.200 0.140 0.400 0.495 1.100 0.590 0.270 0.300
## [23] 0.065
## attr(,"na.action")
## [1]  6 19 25
## attr(,"class")
## [1] "omit"

(guelph.2 <- na.omit(as.vector(window(GuelphP, start=1974+2/12,
                                      end=NULL))))

##  [1] 0.240 0.058 0.079 0.065 0.120 0.091 0.058 0.120 0.120 0.110 0.460
## [12] 0.150 0.086 0.028 0.110 0.360 0.180 0.065 0.130 0.120 0.190 0.150
## [23] 0.107 0.047 0.055 0.080 0.071 0.121 0.108 0.169 0.066 0.079 0.104
## [34] 0.157 0.140 0.070 0.056 0.042 0.116 0.106 0.094 0.097 0.050 0.079
## [45] 0.114
## attr(,"na.action")
## [1] 15
## attr(,"class")
## [1] "omit"

mean(guelph.1); mean(guelph.2)

## [1] 0.4430435
## [1] 0.1159556

median(guelph.1); median(guelph.2)

## [1] 0.365
## [1] 0.106

sd(guelph.1); sd(guelph.2)

## [1] 0.2839124
## [1] 0.07789123
```

**Q64 :** *Describe the difference in the two sub-series.* *Jump to A64* •

We would like to state that these are significant differences (not due to chance) but we can't use a *t*-test, because the observations are *not* independent – they are clearly serially and seasonally correlated. So we need to build a time-series model that includes the intervention.

## 5.2 Answers

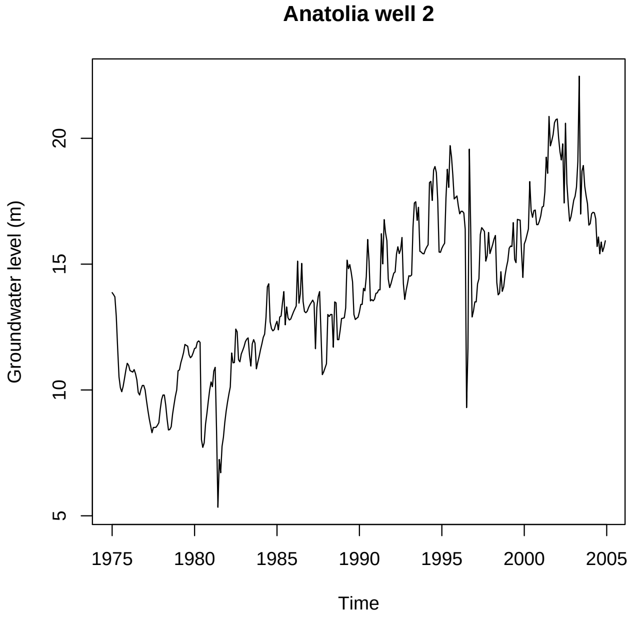
**A63 :** *The mean P concentration decreases dramatically, and the month-to-month variability is much less.* *Return to Q63*
•

**A64 :** *The later series has a much lower mean, median, and especially standard deviation (variability).* *Return to Q64* •

# 6 Comparing two time series

**Task 69 :** Read the second well's time series in R, convert to a time-series object, and plot it. •

```
gw.2 <- ts(scan("./ds_tsa/anatolia_alibe.txt"), start=1975, frequency=12)
plot(gw.2, ylab="Groundwater level (m)", main="Anatolia well 2")
```



**Task 70 :** Create a "multiple time series" object covering the common time period of the two wells. •

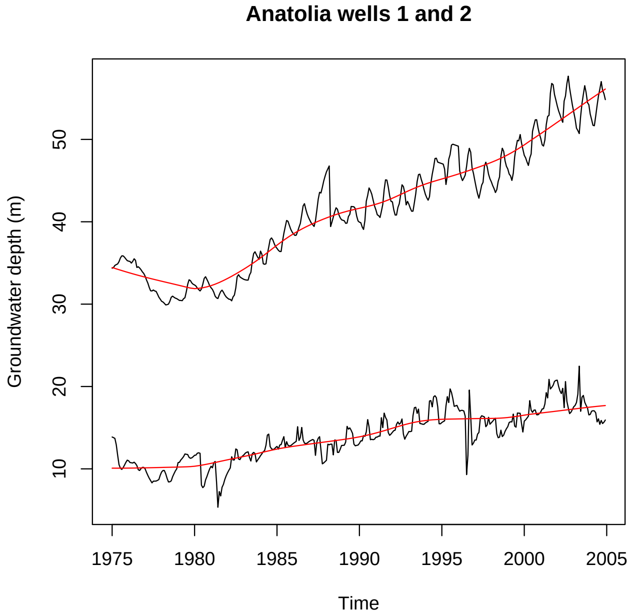
The `plot.ts` function can plot several series together, if they are in a common object. Two (or more) time series can be bound together with the `ts.intersect` and `ts.union` functions; these produce an object of class `mts` "multiple time series", with one column vector for each original series.

These differ in that `ts.union` pads non-overlapping portions of one or more series with NA's, whereas `ts.intersect` limits the multivariate series to their common times. In the present case both have the same period so there is no difference.

```
gw2 <- ts.intersect(gw,gw.2)
class(gw2)

## [1] "mts"    "ts"     "matrix"

str(gw2)

##  Time-Series [1:360, 1:2] from 1975 to 2005: 34.4 34.5 34.7 34.8 34.9 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "gw" "gw.2"
```

**Task 71 :** Plot the two wells' time series on the same graph. •

```
plot(gw2, plot.type="single", main="Anatolia wells 1 and 2",
     ylab="Groundwater depth (m)")
lines(lowess(gw, f=1/3), col="red")
lines(lowess(gw.2, f=1/3), col="red")
```



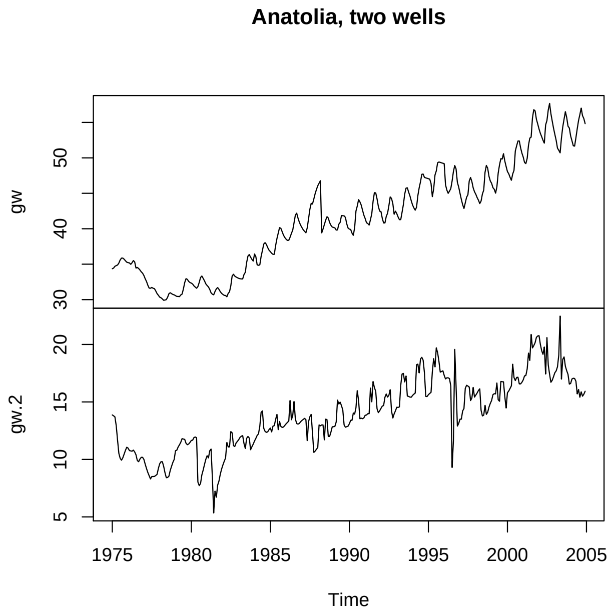
```
              gw  gw.2
Jan 1975 34.36 13.87
Feb 1975 34.45 13.79
...
Nov 2004 55.55 15.67
Dec 2004 54.83 15.93
```

```
summary(gw2)
```

```
##        gw             gw.2
##  Min.   :29.90   Min.   : 5.34
##  1st Qu.:34.90   1st Qu.:11.55
##  Median :41.63   Median :13.74
##  Mean   :41.58   Mean   :13.85
##  3rd Qu.:46.80   3rd Qu.:16.19
##  Max.   :57.68   Max.   :22.47
```

Another way to see the two series is each on their own panel. This has the effect of stretching or compressing the response (here, groundwater level) to the same scale:

```
plot(gw2, plot.type="multiple", main="Anatolia, two wells")
```



**Q65** : *Is there a systematic difference between the wells? Do they appear to have the same temporal pattern?* *Jump to A65* •

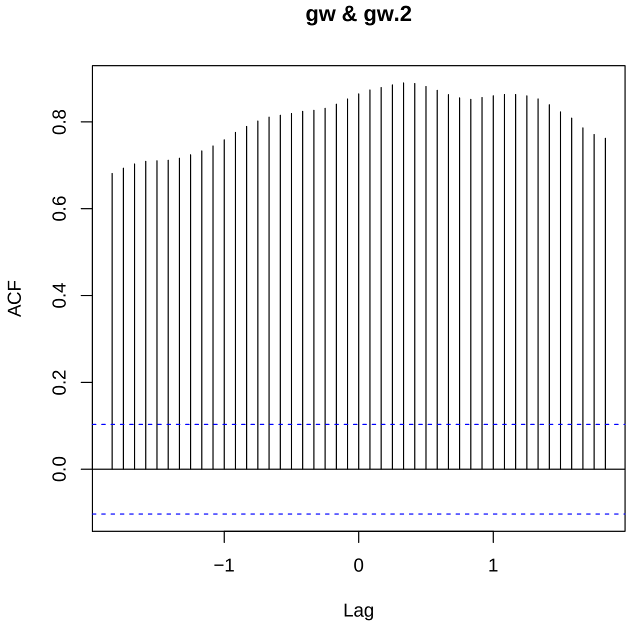
The obvious question is how well are the two series correlated? Function `ccf` computes the **cross-correlation** of two univariate series at a series of lags. Note that the highest correlation between two series might not be at lag 0 (same time), one series may lag ahead or behind the other (for example, stream flow at different distances from the source).

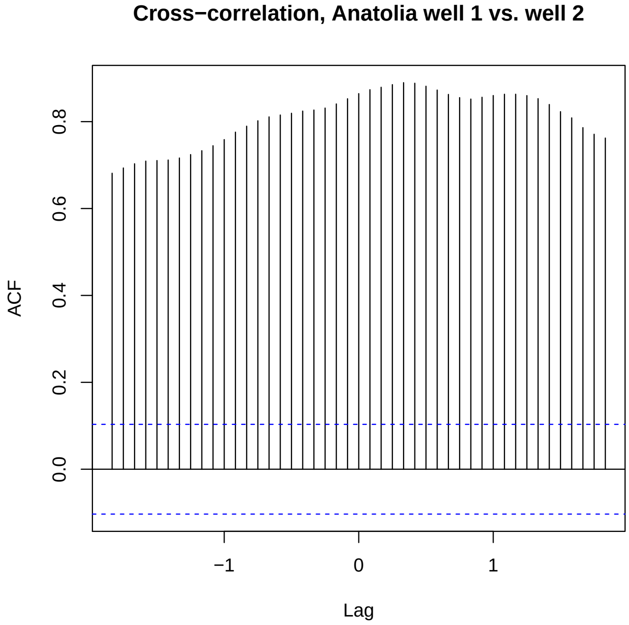
By convention the first series named is moved ahead of the second when computing; so the cross-correlation is between $x_{t+k}$ of the first series and $y_t$ of the second. So a positive lag has the first series ahead of the second, a negative lag the second is ahead of the first.

**Task 72** : Compute and display the cross-correlation between the two wells. •

The `ccf` function has an argument `lag.max` which by default is the integer nearest $10 \cdot \log 10N/m$, here 23, i.e. almost two years. This is computed in both directions.

```
(cc <- ccf(gw,gw.2))

##
## Autocorrelations of series 'X', by lag
##
## -1.8333 -1.7500 -1.6667 -1.5833 -1.5000 -1.4167 -1.3333 -1.2500
##   0.681   0.693   0.703   0.709   0.710   0.712   0.716   0.724
## -1.1667 -1.0833 -1.0000 -0.9167 -0.8333 -0.7500 -0.6667 -0.5833
##   0.733   0.745   0.759   0.776   0.790   0.802   0.811   0.815
## -0.5000 -0.4167 -0.3333 -0.2500 -0.1667 -0.0833  0.0000  0.0833
##   0.819   0.824   0.827   0.831   0.841   0.853   0.864   0.874
##  0.1667  0.2500  0.3333  0.4167  0.5000  0.5833  0.6667  0.7500
##    0.879   0.885   0.890   0.888   0.881   0.873   0.863   0.855
##  0.8333  0.9167  1.0000  1.0833  1.1667  1.2500  1.3333  1.4167
##    0.852   0.856   0.860   0.863   0.863   0.860   0.853   0.839
##  1.5000  1.5833  1.6667  1.7500  1.8333
##    0.823   0.809   0.786   0.771   0.762

plot(cc, main="Cross-correlation, Anatolia well 1 vs. well 2")
```





**Q66 :** *What is the correlation at zero lag, i.e. the same month and year? What is the correlation at one month positive and negative lag?* *Jump to A66* •

**Q67** : *Why is this graph not symmetric about zero-lag?* *Jump to A67* •

**Task 73** : Find the highest correlation and its lag. •

The `max` function finds the maximum in a vector, and `which.max` identifies its index. The structure (using `str`) shows that the `acf` field of the object returned by `ccf` has the correlation, and the `lag` field has the lag, here in years. To get the lag in months, multiply by the cycle length (i.e. 12).

```
str(cc)

## List of 6
##  $ acf   : num [1:45, 1, 1] 0.681 0.693 0.703 0.709 0.71 ...
##  $ type  : chr "correlation"
##  $ n.used: int 360
##  $ lag   : num [1:45, 1, 1] -1.83 -1.75 -1.67 -1.58 -1.5 ...
##  $ series: chr "X"
##  $ snames: chr "gw & gw.2"
##  - attr(*, "class")= chr "acf"

max(abs(cc$acf)) # check highest correlation

## [1] 0.8896892

(i <- which.max(abs(cc$acf)))

## [1] 27

cc$acf[i]

## [1] 0.8896892

cc$acf[i]^2

## [1] 0.7915468

cc$lag[i]   # check maximum lag

## [1] 0.3333333

12*cc$lag[i]   # maximum lag, in months

## [1] 4
```

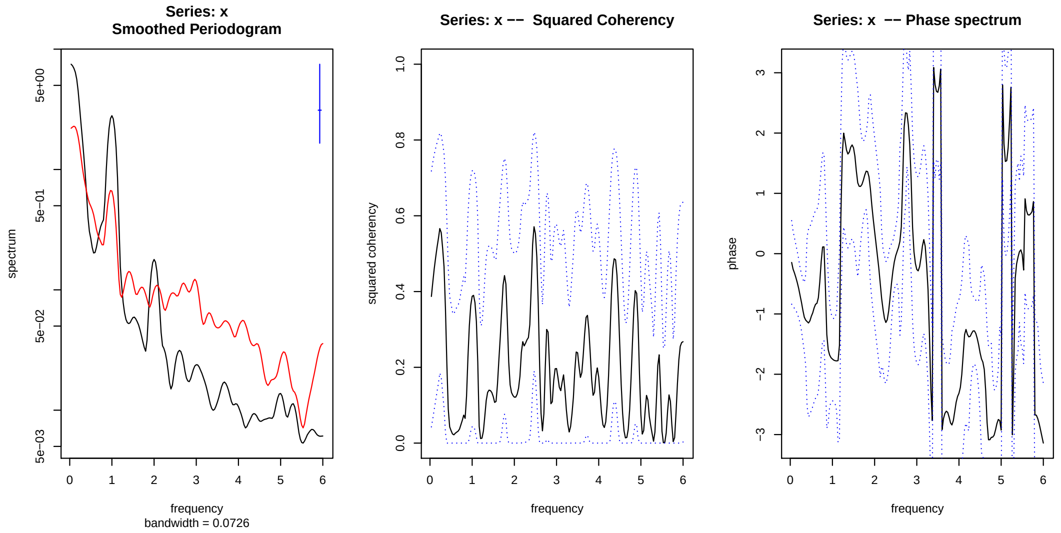
**Q68** : *What is the highest correlation and its lag? Which series leads?* *Jump to A68* •

**Q69** : *Looking at the graphs and the numeric results, how closely correlated are the two series?* *Jump to A69* •

**Task 74** : Compare the spectra of the two series. •

We can also compare the spectral decomposition of the two wells, using the `spectrum` function, as in 3.6. Since there are two series, we can also compute their **coherency**, i.e., how much they agree at each frequency; this is essentially their correlation. Further, it is possible that the series are similar but lagged. An example is monthly rainfall at two stations where a monsoon or other seasonal frontal system reaches one station later than the other. The coherency and phase plots are specified with the `plot.type` optional argument to `spectrum`.

```
par(mfrow=c(1,3))
 spectrum(gw2, spans=c(5,7), lty=1, col=c("black","red"), plot.type="marginal")
 spectrum(gw2, spans=c(5,7), lty=1, col=c("black","red"), plot.type="coherency")
 spectrum(gw2, spans=c(5,7), lty=1, col=c("black","red"), plot.type="phase")
par(mfrow=c(1,1))
```



In the first plot, we can see that the two periodigrams are quite similar, although the second series has somewhat less power at one and two years. There seems to be a slight lag of the second series behind the first, this is most obvious at frequencies 0.5 (two years) and 2 (half-year).

In the second plot, we see the coherency between them at each period. This confirms the impression from the first plot that the same climate forcing applies to both. The lack of coherency near frequencies 0.5 and 2 is also shown here.

The third plot shows the a phase differences. There are clear phase differences at frequencies 0.5, 1, and 2 (series 2 lags) and 1.2–1.8 (series 2 leads). The large swings in phase (±) at 3.4 and 5 cycles must be artefacts of the low power at these frequencies.

## 6.1 Answers

**A65 :** *The second well is much closer to the surface than the first. Both have annual cycles but the trend towards deeper levels is much more pronounced* *in the first well. The second well appears to have more rapid fluctuations. The timing of rapid extractions and recharges is different for the two wells.* Return to Q65 •

---

**A66** : *At lag* 0 *the correlation is 0.864; with the first well ahead by one month it is 0.874; for the first well behind by one month 0.853* Return to Q66 •

---

**A67** : *Shifting one series ahead of the other (i.e., lagging it positively) is not the same as shifting it behind. Think for example of correlating January rainfall of one station with March in another (first station lagged +2) or correlating it with November (lag -2); there is no reason to think these will have the same correlation.* Return to Q67 •

---

**A68** : *The highest correlation is 0.89 at lag 4; the first well's records are moved forward to match the second well's records.* Return to Q68 •

---

**A69** : *Although the two wells are in the same region with similar climate and land use, the highest correlation is not even 0.9; the coefficient of variation (proportion of variation explained, $R^2$) is only 0.792.* Return to Q69 •

## 7 Gap filling

Many uses of time series require complete series, without any gaps. In practice many series contain such gaps because of mechanical failure or observer error; an example is the daily rainfall records from Lake Tana (§2.2).

Gap filling is reviewed by Salas [15, §19.4].

We first list several approaches to gap filling and their areas of application. Several of these are then illustrated with examples.

**Interpolation** If the series is expected to be continuous (no abrupt changes, e.g. groundwater depths) and if the gap is not long (e.g. a single missing well depth), a simple interpolation from nearby values will usually give satisfactory results.

The interpolation can be linear from the two nearest values, or as a weighted average of nearby values; these are both linear filters.

**Functional interpolation** Again if the series is expected to be continuous and if the gap is not long, another way to interpolate from nearby values is to fit a **function** to the series, either global or (more commonly) some local function such as a local polynomial or spline. This function can then be evaluated at the missing value's time. An example is a periodic function fit to an annual cycle.

**Estimatation from other cycles** For a cyclic series (e.g. annual) a missing value may be estimated by some function (typically the mean) of values at the same position in the cycle. If there is a trend the estimation of and from deviation from trend.

For example, missing June rainfall for one year in a 30-year series could be estimated as the mean of the other 29 June rainfalls. This assumes the year is not overall wetter or drier, which may not be a reasonable assumption.

**Estimation from other series** If the series with missing values is well-correlated to one or more other series (e.g. one gauging station among a network, or one weather station in a group), a (multivariate) regression equation can be developed for the time series as a function of the other series.

**Estimation from the autocorrelation structure** If the series is autocorrelated, as revealed by the correlogram and autocorrelation analysis, a gap can be estimated from the observations with which it should be correlated. A typical example is a missing daily weather observation (temperature, dewpoint, etc.). Rainfall can also be estimated, but its autocorrelation structure is usually much weaker.

**Estimation from a model** If the time series is modelled either by structural decomposition or ARIMA, the model can be used to predict at a gap, or even to extend the series.

## 7.1 Simulating short gaps in a time series

To illustrate gap filling, we **simulate** gaps by removing known records from a series; we can then assess the success of the method by comparing the reconstructed series to the known series.

---

**Task 75 :** Remove five months at random from the first Anatolia well.

•

Recall, this is in time series `gw`:

```
str(gw)

##  Time-Series [1:360] from 1975 to 2005: 34.4 34.5 34.7 34.8 34.9 ...
```

First, set up the series with simulated gaps, initially the same as the known series:

```
gwg <- gw
```

Second, pick five separate positions to delete, using the `sample` function. So your results match these, we use `set.seed` to set the random number generator to a known starting position.

```
set.seed(0044)
(ix <- sample(length(gw), size=5))
```

```
## [1] 268 102 182 206  12

sort(ix)

## [1]  12 102 182 206 268
```

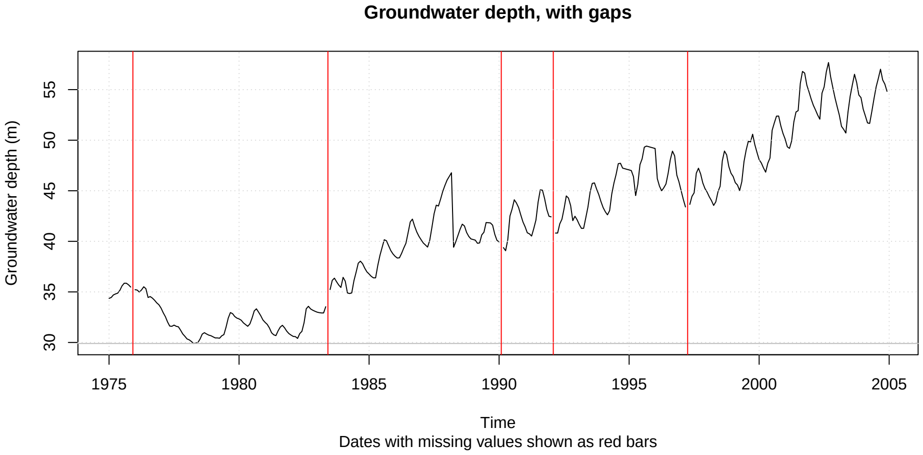
Third, delete the values at these positions, replacing them with the missing value constant NA:

```
summary(gwg)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   29.90   34.90   41.63   41.58   46.80   57.68

gwg[ix] <- NA
summary(gwg)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   29.90   34.89   41.70   41.62   46.92   57.68       5
```

**Q70** : *What is different in the summary of the simulated time series, before and after deletion?* *Jump to A70* •

**Task 76** : Plot the simulated series, showing the dates with missing observations. •

```
plot(gwg, main="Groundwater depth, with gaps",
     ylab="Groundwater depth (m)",
     sub="Dates with missing values shown as red bars")
abline(h = min(gw), col = "gray")
abline(v=time(gw)[ix], , col = "red", lwd=1)
grid()
```



**Task 77** : Try to decompose the time series with gaps into a trend, seasonal component, and residuals. •

We use the best decomposition from §3.3, i.e. with a smoother trend and two-year window on the seasonal amplitude. I know that this will produce an error; to avoid a fatal error I thus enclose the expression in a call to the `try` function, and show the error message with the `geterrmessage` function:

```
try(gwg.stl <- stl(gwg, s.window=25, t.window=85))

## Error in na.fail.default(as.ts(x)) : missing values in object

geterrmessage()

## [1] "Error in na.fail.default(as.ts(x)) : missing values in object\n"
```

The error message tells us that the `stl` function failed because there are missing values ...a fact we know! The usual R approach to missing values is to specify a `na.action`, such as `na.omit` or `na.exclude`; in this case neither help:

```
try(gwg.stl <- stl(gwg, s.window=25, t.window=85, na.action="na.exclude"),
    silent=TRUE)
geterrmessage()

## [1] "Error in stl(gwg, s.window = 25, t.window = 85, na.action = \"na.exclude\") : \n  ser
```

The missing observations have been taken out of the series, so it is no longer periodic; `stl` correctly reports this. Clearly, the gaps have to be filled before analyzing the series.

## 7.2 Gap filling by interpolation

In the previous example we are missing single observations. The first attempt to fill the gaps is to interpolate from neighbouring values.

### 7.2.1 Linear interpolation

The simplest interpolation is as the average of neighbouring (non-missing) values; the `approx` function implements this.

**Task 78 :** Predict the ground water depths at the dates with missing values, using linear interpolation. •

Recall (§7.1) that we know the positions in the original time-series object `gw` for which we deleted the values to make series `gwg`:

```
print(ix)

## [1] 268 102 182 206  12

gw[ix]

## [1] 42.86 33.86 39.89 41.46 35.28

gwg[ix]

## [1] NA NA NA NA NA
```

The dates of these are found with the `time` function on the original series, and the `[]` indexing operator, using the positions of the missing observations as indices:

```
time(gw)[ix]
```

```
## [1] 1997.250 1983.417 1990.083 1992.083 1975.917
```

We now call the `aspline` function, specifying series to be interpolated as the `x` and `y` values, and the missing times as the points to be interpolated (argument `xout`).

```
gw.fill.linear <- approx(x=time(gwg), y=gwg, xout=time(gw)[ix])
str(gw.fill.linear)

## List of 2
##  $ x: num [1:5] 1997 1983 1990 1992 1976
##  $ y: num [1:5] 43.5 34.4 39.7 41.6 35.3

print(gw.fill.linear)

## $x
## [1] 1997.250 1983.417 1990.083 1992.083 1975.917
##
## $y
## [1] 43.525 34.385 39.665 41.620 35.350
```

Compare these to the known values:

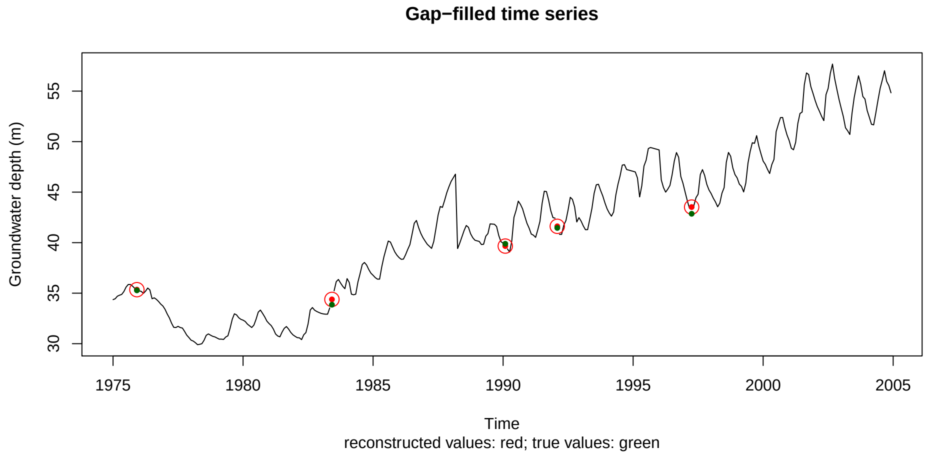
```
time(gw)[ix]

## [1] 1997.250 1983.417 1990.083 1992.083 1975.917

gw.fill.linear$y

## [1] 43.525 34.385 39.665 41.620 35.350

gw[ix]

## [1] 42.86 33.86 39.89 41.46 35.28

summary((gw[ix] - gw.fill.linear$y))

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.665  -0.525  -0.160  -0.239  -0.070   0.225
```

The maximum difference is 0.66 m.

---

**Task 79** : Plot the reconstructed points on the time series with missing values, along with the original series. •

```
plot(gwg, main="Gap-filled time series",
     sub="reconstructed values: red; true values: green",
     ylab="Groundwater depth (m)")
points(gw.fill.linear$x, gw.fill.linear$y, col="red", cex=2)
points(gw.fill.linear$x, gw.fill.linear$y, col="red", pch=20)
points(time(gw)[ix], gw[ix], col="darkgreen", pch=20)
```



**Q71** : *How well did the linear interpolation fill the gaps?* *Jump to A71*
•

### 7.2.2 Spline interpolation

The linear interpolation only takes into account neighbouring values; a higher-order local curve fits to a neighbourhood. One example is a **spline**, which is a smooth function that passes through known points and preserves several derivatives, usually two. A nice implementation of 1D splines is the `aspline` function of the `akima` " Linear or cubic spline interpolation for irregular gridded data" package, based on methods developed by Akima [1].

The base R `stats` has a `spline` function that is similar but less sophisticated; we will compare the Akima and default splines.

```
require(akima)

## Loading required package:  akima
```

**Task 80** : Predict the ground water depths at the dates with missing values, using spline interpolation. •

We now call the `aspline` function, specifying series to be interpolated as the `x` and `y` values, and the missing times as the points to be interpolated (argument `xout`).

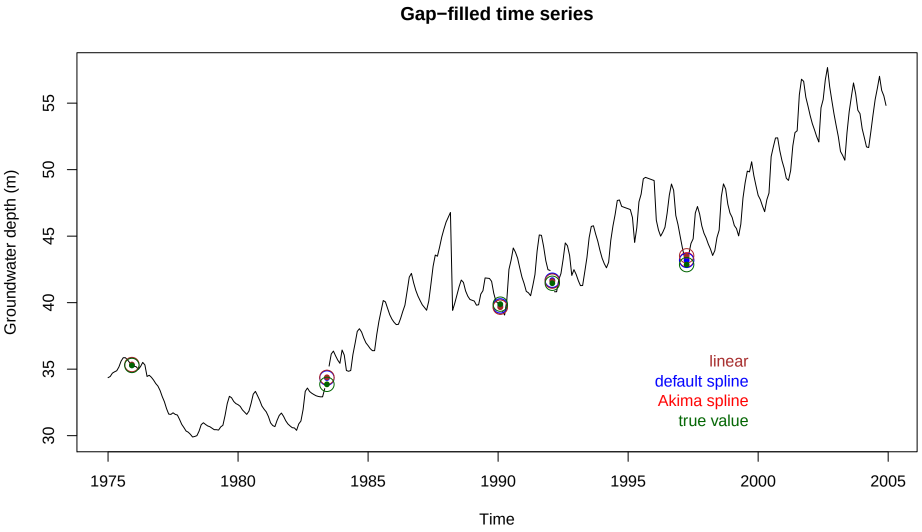
```
gw[ix]

## [1] 42.86 33.86 39.89 41.46 35.28

(gw.fill.aspline <- aspline(x=time(gwg), y=gwg, xout=time(gw)[ix]))$y

## [1] 43.22304 34.37567 39.66733 41.63453 35.32840

(gw.fill.spline <- spline(x=time(gwg), y=gwg, xout=time(gw)[ix]))$y

## [1] 43.17678 34.33328 39.76171 41.69033 35.29984
```

Compare these to each other and the linear interpolator:

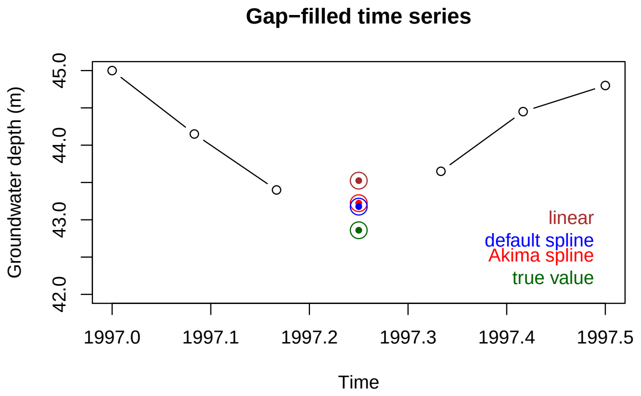
```
summary(gw.fill.aspline$y - gw.fill.spline$y)

##       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.094384 -0.055798  0.028558 -0.006595  0.042386  0.046262

summary(gw.fill.aspline$y - gw.fill.linear$y)

##       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.301963 -0.021600 -0.009333 -0.063208  0.002326  0.014528

summary(gw.fill.spline$y - gw.fill.linear$y)

##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.34822 -0.05172 -0.05016 -0.05661  0.07033  0.09671
```

**Task 81** : Plot the reconstructed points on the time series with missing values, computed by three methods (linear, default spline, Akima spline) along with the original series: (1) for the whole series; (2) for a six-month window centred on March 1997. •

```
plot(gwg, main="Gap-filled time series", type="l",
     ylab="Groundwater depth (m)")
points(gw.fill.aspline$x, gw.fill.aspline$y, col="red", cex=2)
points(gw.fill.aspline$x, gw.fill.aspline$y, col="red", pch=20)
points(gw.fill.spline$x, gw.fill.spline$y, col="blue", cex=2)
points(gw.fill.spline$x, gw.fill.spline$y, col="blue", pch=20)
points(gw.fill.linear$x, gw.fill.linear$y, col="brown", cex=2)
points(gw.fill.linear$x, gw.fill.linear$y, col="brown", pch=20)
points(time(gw)[ix], gw[ix], col="darkgreen", cex=2)
points(time(gw)[ix], gw[ix], col="darkgreen", pch=20)
text(2000, 35.5, "linear", col="brown", pos=2)
text(2000, 34, "default spline", col="blue", pos=2)
text(2000, 32.5, "Akima spline", col="red", pos=2)
text(2000, 31, "true value", col="dark green", pos=2)
```



```
plot(window(gwg, start=1997, end=1997.5), main="Gap-filled time series",
     ylim=c(42,45), type="b", ylab="Groundwater depth (m)")
points(gw.fill.aspline$x, gw.fill.aspline$y, col="red", cex=2)
points(gw.fill.aspline$x, gw.fill.aspline$y, col="red", pch=20)
points(gw.fill.spline$x, gw.fill.spline$y, col="blue", cex=2)
points(gw.fill.spline$x, gw.fill.spline$y, col="blue", pch=20)
points(gw.fill.linear$x, gw.fill.linear$y, col="brown", cex=2)
points(gw.fill.linear$x, gw.fill.linear$y, col="brown", pch=20)
points(time(gw)[ix], gw[ix], col="darkgreen", cex=2)
points(time(gw)[ix], gw[ix], col="darkgreen", pch=20)
text(1997.5, 43, "linear", col="brown", pos=2)
text(1997.5, 42.7, "default spline", col="blue", pos=2)
text(1997.5, 42.5, "Akima spline", col="red", pos=2)
text(1997.5, 42.2, "true value", col="dark green", pos=2)
```



**Q72 :** *How different are the three interpolators?* *Jump to A72* •

**Task 82 :** Make a full time series with the reconstructed values. •

First, copy the series with gaps, then fill in their values:

```
gwg.r <- gwg
sum(is.na(gwg.r))

## [1] 5

gwg.r[ix] <- gw.fill.aspline$y
str(gwg.r)

##  Time-Series [1:360] from 1975 to 2005: 34.4 34.5 34.7 34.8 34.9 ...

sum(is.na(gwg.r))

## [1] 0
```

## 7.3 Simulating longer gaps in time series

The linear and spline interpolators had little problem with single gaps in a smooth series. What happens with longer gaps?

### 7.3.1 Non-systematic gaps

These are like the short gaps; no systematic problem but the result of carelessness or occasional problems with observations.

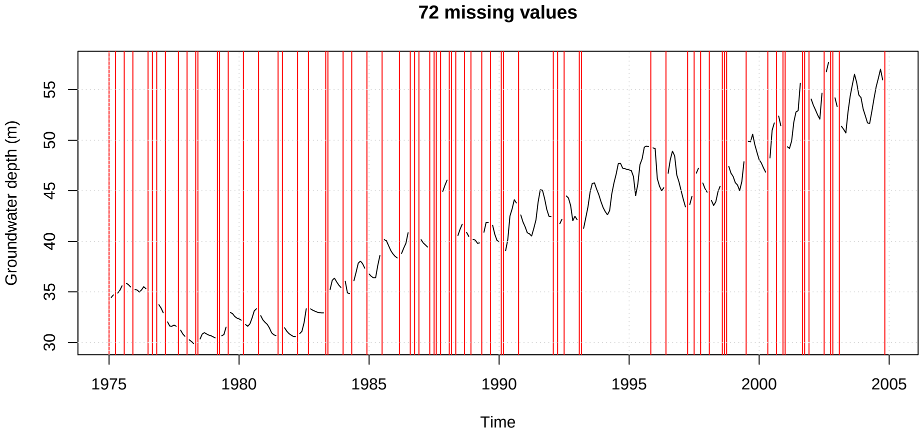
**Task 83 :** Remove one-fifth of the months at random from the first

Anatolia well. •

Again we use `set.seed` so your results will be the same. For convenience in plotting, we also create a sorted version of the missing-observations vector, using the `sort` function.

```
gwg <- gw
set.seed(0044)
(six <- sort(ix <- sample(length(gw), size=length(gw)/5)))

##  [1]   1   4   8  12  19  21  23  27  33  37  41  42  51  52  56  63
## [17]  70  79  81  88  93 101 102 109 113 120 127 135 140 142 144 149
## [33] 151 152 154 158 159 161 165 168 173 177 182 183 190 206 208 211
## [49] 218 219 251 258 268 271 274 278 284 285 286 295 305 309 312 313
## [65] 321 322 324 331 334 335 338 359

time(gw)[six]

##  [1] 1975.000 1975.250 1975.583 1975.917 1976.500 1976.667 1976.833
##  [8] 1977.167 1977.667 1978.000 1978.333 1978.417 1979.167 1979.250
## [15] 1979.583 1980.167 1980.750 1981.500 1981.667 1982.250 1982.667
## [22] 1983.333 1983.417 1984.000 1984.333 1984.917 1985.500 1986.167
## [29] 1986.583 1986.750 1986.917 1987.333 1987.500 1987.583 1987.750
## [36] 1988.083 1988.167 1988.333 1988.667 1988.917 1989.333 1989.667
## [43] 1990.083 1990.167 1990.750 1992.083 1992.250 1992.500 1993.083
## [50] 1993.167 1995.833 1996.417 1997.250 1997.500 1997.750 1998.083
## [57] 1998.583 1998.667 1998.750 1999.500 2000.333 2000.667 2000.917
## [64] 2001.000 2001.667 2001.750 2001.917 2002.500 2002.750 2002.833
## [71] 2003.083 2004.833

gwg[ix] <- NA
summary(gwg)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   29.90   35.17   41.89   41.66   46.88   57.68      72
```

```
plot(gwg, ylab="Groundwater depth (m)", main="72 missing values")
abline(v=time(gw)[six], , col = "red", lwd=1)
grid()
```



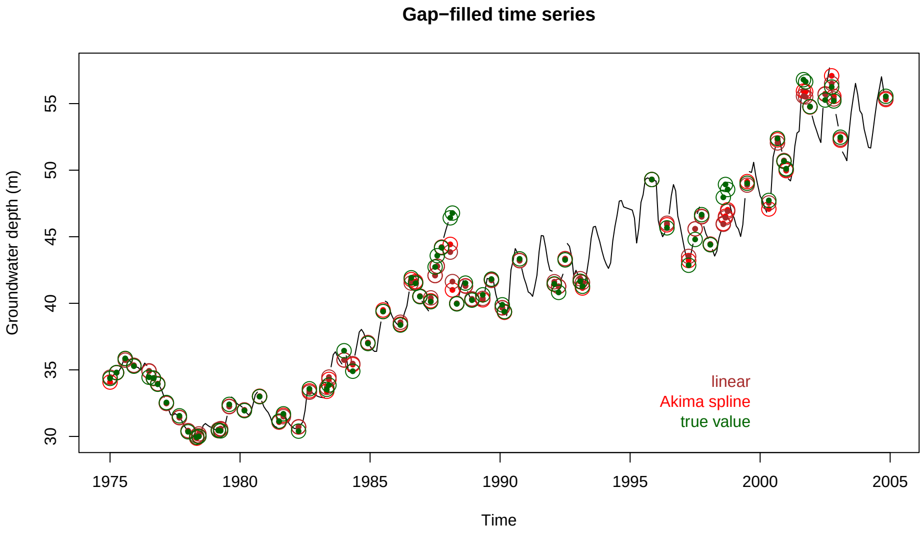
---

**Task 84 :** Fill these with linear interpolation and Akima splines; compare the results. •

Again we use `approx` and `aspline`; for `approx` we must specify the rule to be used at the end of the series (since in this case the first observation was missing); the default `rule=1` returns `NA`; to get a value we specify rule=2:

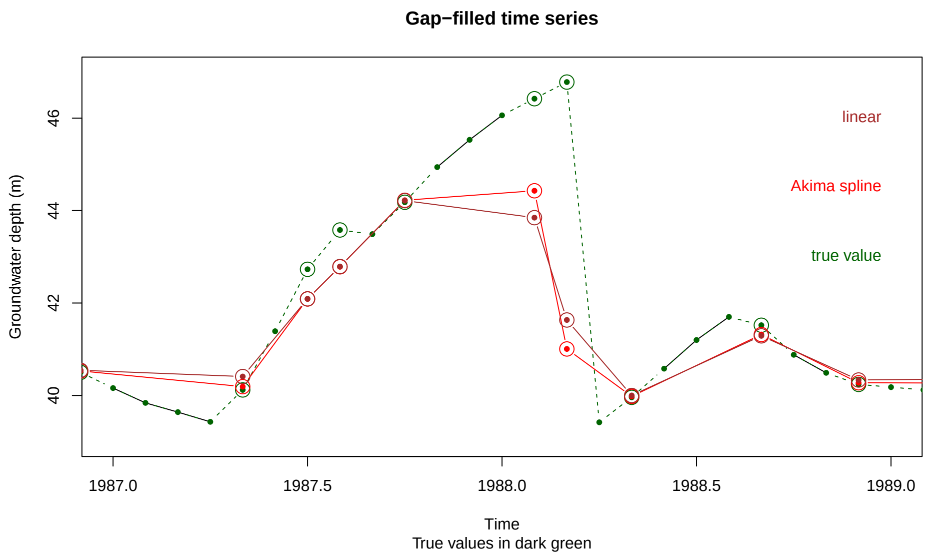
```
gw.fill.linear <- approx(x=time(gwg), y=gwg, xout=time(gw)[six], rule=2)
gw.fill.aspline <- aspline(x=time(gwg), y=gwg, xout=time(gw)[six])
summary(gw.fill.linear$y - gw[six])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -5.1467 -0.1750 -0.0050 -0.1980  0.1508  0.7950

summary(gw.fill.aspline$y - gw[six])

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -5.77350 -0.15037 -0.02987 -0.20920  0.06561  0.83871

summary(gw.fill.aspline$y - gw.fill.linear$y)

##      Min.    1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.626830 -0.077590 -0.003457 -0.011213  0.031560  0.581801
```

```
plot(gwg, main="Gap-filled time series", type="l",
     ylab="Groundwater depth (m)")
points(gw.fill.aspline$x, gw.fill.aspline$y, col="red", cex=2)
points(gw.fill.aspline$x, gw.fill.aspline$y, col="red", pch=20)
points(gw.fill.linear$x, gw.fill.linear$y, col="brown", cex=2)
points(gw.fill.linear$x, gw.fill.linear$y, col="brown", pch=20)
points(time(gw)[ix], gw[ix], col="darkgreen", cex=2)
points(time(gw)[ix], gw[ix], col="darkgreen", pch=20)
text(2000, 34, "linear", col="brown", pos=2)
text(2000, 32.5, "Akima spline", col="red", pos=2)
text(2000, 31, "true value", col="dark green", pos=2)
```



The period from mid-1987 to mid-1988 has, by chance, many missing values, so we plot these in detail:

```
plot(window(gwg, start=1987, end=1989),
     main="Gap-filled time series",
     sub="True values in dark green", type="l",
     ylab="Groundwater depth (m)", ylim=c(39,47))
points(x=time(gw), y=as.numeric(gw), col="darkgreen", lty=2, pch=20, type="b")
points(time(gw)[ix], gw[ix], col="darkgreen", cex=2)
points(gw.fill.aspline$x, gw.fill.aspline$y, col="red", cex=2, type="b", lty=1)
points(gw.fill.aspline$x, gw.fill.aspline$y, col="red", pch=20)
points(gw.fill.linear$x, gw.fill.linear$y, col="brown", cex=2, type="b", lty=1)
points(gw.fill.linear$x, gw.fill.linear$y, col="brown", pch=20)
text(1989, 46, "linear", col="brown", pos=2)
text(1989, 44.5, "Akima spline", col="red", pos=2)
text(1989, 43, "true value", col="dark green", pos=2)
```



**Q73** : *How well did the interpolators fill the longer gaps?* Jump to A73
•

### 7.3.2 Systematic gaps

These are when observations are not made for some block of time, perhaps because of a broken instrument or budget problems. The most interesting case is when an entire cycle is missing.

**Task 85** : Simulate a missing year in the groundwater depth records. •

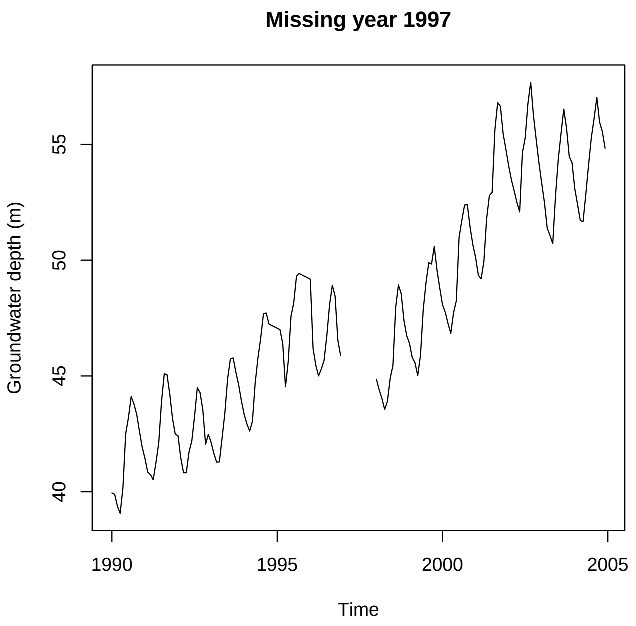
We know the years of the record, extracted from the full date with the `floor` function:

```
unique(floor(time(gw)))

##  [1] 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
## [14] 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
## [27] 2001 2002 2003 2004
```

Select a “typical” year, 1997; remove it; to make the display easier to use just consider the series since 1990, using the `window` function. We use the `which` function to find the array indices for 1997.

```
gww <- window(gw, start=1990)
gwg <- gww
(six <- sort(which(floor(time(gwg))==1997)))

##  [1] 85 86 87 88 89 90 91 92 93 94 95 96

gwg[six] <- NA
summary(gwg)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   39.07   43.92   47.21   47.78   51.71   57.68      12
```

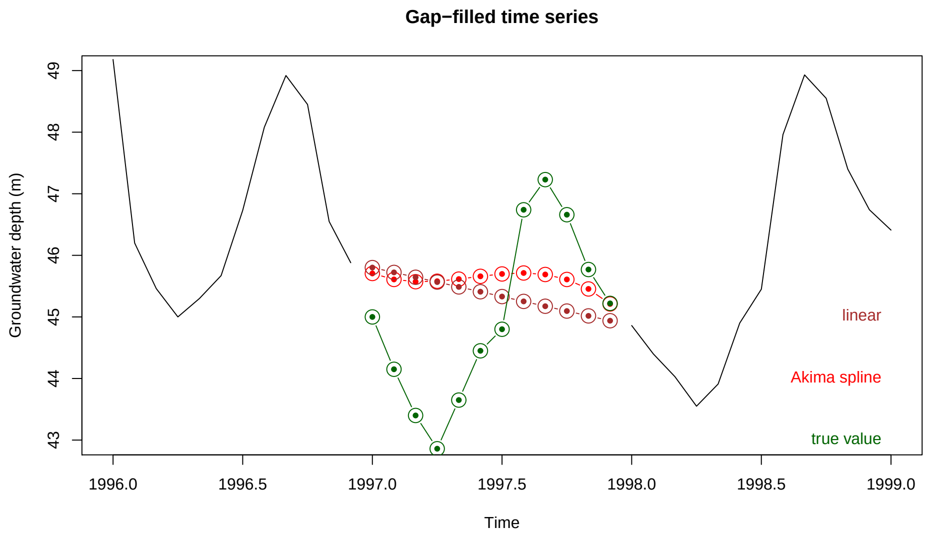
```
plot(gwg, main="Missing year 1997", ylab="Groundwater depth (m)")
```



**TASK 86** : Interpolate the missing values with linear interpolation and Akima splines. •

```
gw.fill.linear <- approx(x=time(gwg), y=gwg, xout=time(gww)[six], rule=2)
gw.fill.aspline <- aspline(x=time(gwg), y=gwg, xout=time(gww)[six])
summary(gw.fill.linear$y - gww[six])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.0562 -0.9367  0.6662  0.3758  1.6392  2.7062

summary(gw.fill.aspline$y - gww[six])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.5406 -0.4931  0.8019  0.5988  1.5852  2.7198

summary(gw.fill.aspline$y - gw.fill.linear$y)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.11433 -0.00781  0.26119  0.22296  0.44363  0.51554
```

```
plot(window(gwg, start=1996, end=1999), main="Gap-filled time series",
     type="l", ylab="Groundwater depth (m)", ylim=c(43,49))
points(gw.fill.aspline$x, gw.fill.aspline$y,
       col="red", cex=2, lty=1, type="b")
points(gw.fill.aspline$x, gw.fill.aspline$y,
       col="red", pch=20)
points(gw.fill.linear$x, gw.fill.linear$y,
       col="brown", cex=2, lty=1, type="b")
points(gw.fill.linear$x, gw.fill.linear$y,
       col="brown", pch=20)
points(time(gww)[six], gww[six],
       col="darkgreen", cex=2, lty=1, type="b")
points(time(gww)[six], gww[six],
       col="darkgreen", pch=20)
text(1999, 45, "linear", col="brown", pos=2)
text(1999, 44, "Akima spline", col="red", pos=2)
text(1999, 43, "true value", col="dark green", pos=2)
```



**Q74** : *Are these satisfactory interpolators?* *Jump to A74* •

Another method is clearly called for.

## 7.4 Estimation from other series

For irregular series like rainfall or stream levels, where there are rapid changes in short times, and often periods with zeroes (rainfall) or low values (baseflow of streams), gap-filling with interpolation will not work. One simple method is to estimate from correlated time series, that do have some observations in common, and have observations at the same times as the missing values.

This is complicated by the fact that correlations over the whole series may not be consistent. For example, correlations of rainfall records in a dry season may be very good (everything is low or zero most days) but poor in a wet season (even if it rains on the same days, the amounts may be considerably different).

To examine this method, we use two additional rainfall stations in the Lake Tana basin, following the procedures from §2.2.

**Task 87** : Read the CSV files for stations `WeteAbay` and `Dangila` into R objects and examine their structures. •

```
tana.2<- read.csv("./ds_tsa/Tana_Daily_WeteAbay.csv", skip=1, header=T,
                  colClasses=c(rep("integer",2), rep("character",12)),
                  blank.lines.skip=T,na.strings=c("N.A","NA"," "))
str(tana.2)

## 'data.frame': 217 obs. of  14 variables:
##  $ Year: int  2000 NA NA NA NA NA NA NA NA NA ...
##  $ Date: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Jan : chr  "0" "0" "0" "0" ...
##  $ Feb : chr  "0" "0" "0" "0" ...
##  $ Mar : chr  "0" "0" "0" "0" ...
##  $ Apr : chr  "0" "0" "0" "12.9" ...
##  $ May : chr  "0" "0" "0" "12.6" ...
##  $ Jun : chr  "0" "12.6" "10.4" "10.3" ...
##  $ Jul : chr  "9" "0.3" "8.4" "46.6" ...
##  $ Aug : chr  "8.4" "4.4" "6.6" "6.4" ...
##  $ Sep : chr  "14" "0" "10.9" "1.4" ...
##  $ Oct : chr  "10.1" "27.8" "0" "20.2" ...
##  $ Nov : chr  "0" "0" "0" "0" ...
##  $ Dec : chr  "0" "0" "0" "0" ...

sum(is.na(tana.2[,3:14]))

## [1] 0

tana.3<- read.csv("./ds_tsa/Tana_Daily_Dangila.csv", skip=1, header=T,
                  colClasses=c(rep("integer",2), rep("character",12)),
                  blank.lines.skip=T,na.strings=c("N.A","NA"," "))
str(tana.3)

## 'data.frame': 620 obs. of  14 variables:
##  $ Year: int  1987 NA NA NA NA NA NA NA NA NA ...
##  $ Date: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Jan : chr  "0" "0" "0" "0" ...
##  $ Feb : chr  "0" "0" "0" "0" ...
##  $ Mar : chr  NA NA NA NA ...
##  $ Apr : chr  NA NA NA NA ...
##  $ May : chr  NA NA NA NA ...
##  $ Jun : chr  NA NA NA "0" ...
##  $ Jul : chr  "3.4" "7.8" "8.6" "13.3" ...
##  $ Aug : chr  "44.6" "22.3" "3.6" "22" ...
##  $ Sep : chr  "5.4" "1" "37.9" "2.2" ...
##  $ Oct : chr  "4.8" "2.2" "8.3" "4.5" ...
##  $ Nov : chr  "0" "0" "0" "0" ...
##  $ Dec : chr  "0" "0" "0" "0" ...

sum(is.na(tana.3[,3:14]))

## [1] 383
```

**Task 88** : Set the trace values and any measurements below 0.1 to zero. •

```
require(car)
for (i in 3:14) {
tana.2[,i] <- recode(tana.2[,i], "c('TR','tr','0.01')='0'")
}
for (i in 3:14) {
tana.3[,i] <- recode(tana.3[,i], "c('TR','tr','0.01')='0'")
}
sum(c(tana.2[,3:14],tana.3[,3:14])=="TR", na.rm=TRUE)

## [1] 0

sum(c(tana.2[,3:14],tana.3[,3:14])=="tr", na.rm=TRUE)

## [1] 0

sum(c(tana.2[,3:14],tana.3[,3:14])=="0.01", na.rm=TRUE)

## [1] 0

sum(c(tana.2[,3:14],tana.3[,3:14])=="0", na.rm=TRUE)

## [1] 0
```

**Task 89 :** Organize the daily values as one long vector of values, as required for time series analysis. •

Which years do we have?

```
sort(unique(tana$YEAR))

##  [1] 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993
## [14] 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006

sort(unique(tana.2$Year))

## [1] 2000 2001 2002 2003 2004 2005 2006

sort(unique(tana.3$Year))

##  [1] 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999
## [14] 2000 2001 2002 2003 2004 2005 2006
```

```
tana.2[tana.2$DATE==29,"FEB"]

## NULL

tana.3[tana.3$DATE==29,"FEB"]

## NULL
```

```
month.days <- c(0,0,31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
tana.2.ppt <- NULL;
for (yr.first.row in seq(from=1, by=32, length=(2006 - 2000 + 1))) {
 for (month.col in 3:14) {
tana.2.ppt <-
    c(tana.2.ppt,
                tana.2[yr.first.row:(yr.first.row + month.days[month.col]-1),
                       month.col])
 }
};
str(tana.2.ppt)

##  chr [1:2555] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" ...

length(tana.2.ppt)/365

## [1] 7

tana.3.ppt <- NULL;
for (yr.first.row in seq(from=1, by=32, length=(2006 - 1987 + 1))) {
 for (month.col in 3:14) {
tana.3.ppt <-
    c(tana.3.ppt,
      tana.3[yr.first.row:(yr.first.row + month.days[month.col]-1),
             month.col])
 }
};
str(tana.3.ppt)

##  chr [1:7300] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" ...

length(tana.3.ppt)/365

## [1] 20

rm(month.days, yr.first.row, month.col)
```

Check that this is an integral number of years:

**Task 90** : Convert this to a time series with the appropriate metadata. •

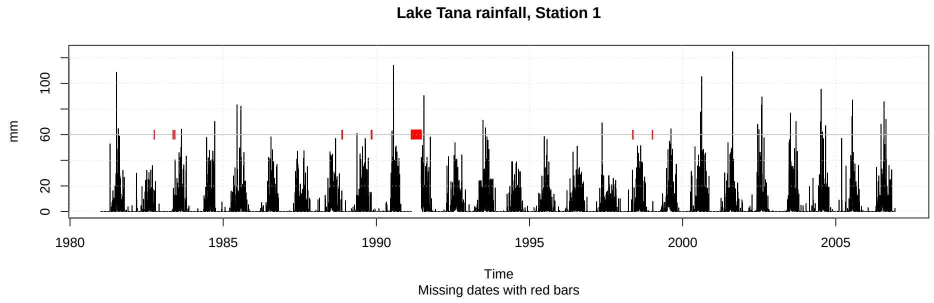
Again, the `ts` function is used to convert the series; the `frequency` argument specifies a cycle of 365 days and the `start` argument specifies the beginning of each series:

```
tana.2.ppt <- ts(tana.2.ppt, start=2000, frequency=365)
str(tana.2.ppt)

##  Time-Series [1:2555] from 2000 to 2007: 0 0 0 0 ...

tana.3.ppt <- ts(tana.3.ppt, start=1987, frequency=365)
str(tana.3.ppt)

##  Time-Series [1:7300] from 1987 to 2007: 0 0 0 0 ...
```

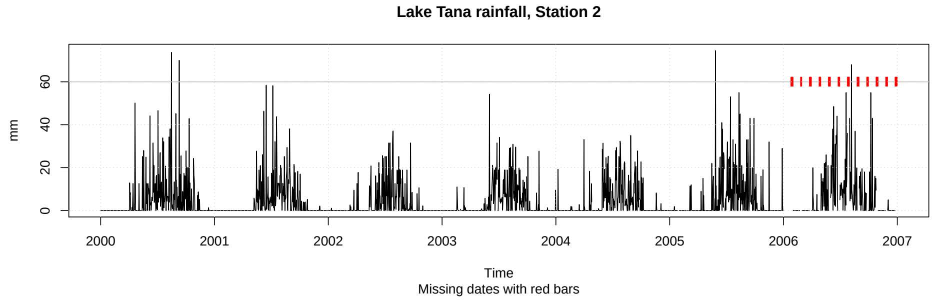
---

**Task 91** : Plot the two time series, and the original station for comparison •

```
plot(tana.ppt, main="Lake Tana rainfall, Station 1",
     ylab="mm", sub="Missing dates with red bars")
abline(h=60, col="gray")
points(xy.coords(x=time(tana.ppt), y=60, recycle=T),
       pch=ifelse(is.na(tana.ppt),"l",""), col="red")
grid()
```

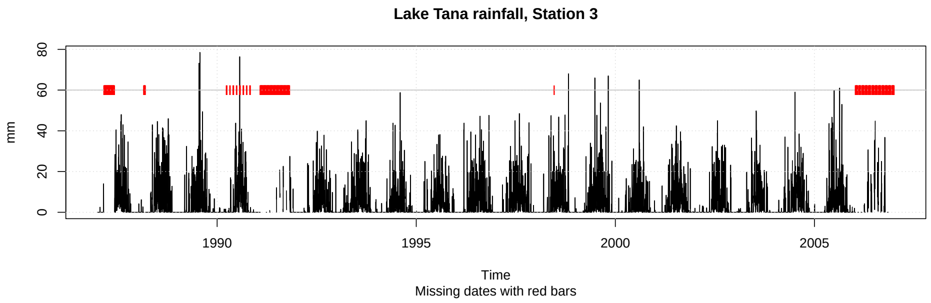


```
plot(tana.2.ppt, main="Lake Tana rainfall, Station 2",
     ylab="mm", sub="Missing dates with red bars")
abline(h=60, col="gray")
points(xy.coords(x=time(tana.2.ppt), y=60, recycle=T),
       pch=ifelse(is.na(tana.2.ppt),"l",""), col="red")
grid()
```



```
plot(tana.3.ppt, main="Lake Tana rainfall, Station 3",
     ylab="mm", sub="Missing dates with red bars")
abline(h=60, col="gray")
points(xy.coords(x=time(tana.3.ppt), y=60, recycle=T),
       pch=ifelse(is.na(tana.3.ppt),"l",""), col="red")
```

```
grid()
```



**Q75** : *Do the three stations cover the same time period? Do they have the same missing dates? Does every date have at least one observation (from at least one of the stations)?* *Jump to A75* •

To answer this, we use the `is.na` logical function to determine whether an observation is a missing value, and the `which` function to identify these positions in the series. We then use these indices to print the times, extracted by the `time` function.

```
(miss.2 <- which(is.na(tana.2.ppt)))

##  [1] 2216 2217 2218 2219 2220 2221 2247 2248 2249 2275 2276 2277 2278
## [14] 2279 2280 2306 2307 2308 2309 2310 2336 2337 2338 2339 2340 2341
## [27] 2367 2368 2369 2370 2371 2397 2398 2399 2400 2401 2402 2428 2429
## [40] 2430 2431 2432 2433 2459 2460 2461 2462 2463 2489 2490 2491 2492
## [53] 2493 2494 2520 2521 2522 2523 2524 2550 2551 2552 2553 2554 2555

(time.miss.2 <- time(tana.2.ppt)[miss.2])

##  [1] 2006.068 2006.071 2006.074 2006.077 2006.079 2006.082 2006.153
##  [8] 2006.156 2006.159 2006.230 2006.233 2006.236 2006.238 2006.241
## [15] 2006.244 2006.315 2006.318 2006.321 2006.323 2006.326 2006.397
## [22] 2006.400 2006.403 2006.405 2006.408 2006.411 2006.482 2006.485
## [29] 2006.488 2006.490 2006.493 2006.564 2006.567 2006.570 2006.573
## [36] 2006.575 2006.578 2006.649 2006.652 2006.655 2006.658 2006.660
## [43] 2006.663 2006.734 2006.737 2006.740 2006.742 2006.745 2006.816
## [50] 2006.819 2006.822 2006.825 2006.827 2006.830 2006.901 2006.904
## [57] 2006.907 2006.910 2006.912 2006.984 2006.986 2006.989 2006.992
## [64] 2006.995 2006.997

miss.3 <- which(is.na(tana.3.ppt))
time.miss.3 <- time(tana.3.ppt)[miss.3]
miss.1 <- which(is.na(tana.ppt))
time.miss.1 <- time(tana.ppt)[miss.1]
```

The `intersect` set operator gives the sets where two time series share the same missing dates.

```
length(miss.12 <- intersect(time.miss.1, time.miss.2))

## [1] 0

length(miss.13 <- intersect(time.miss.1, time.miss.3))

## [1] 96

length(miss.23 <- intersect(time.miss.2, time.miss.3))
```

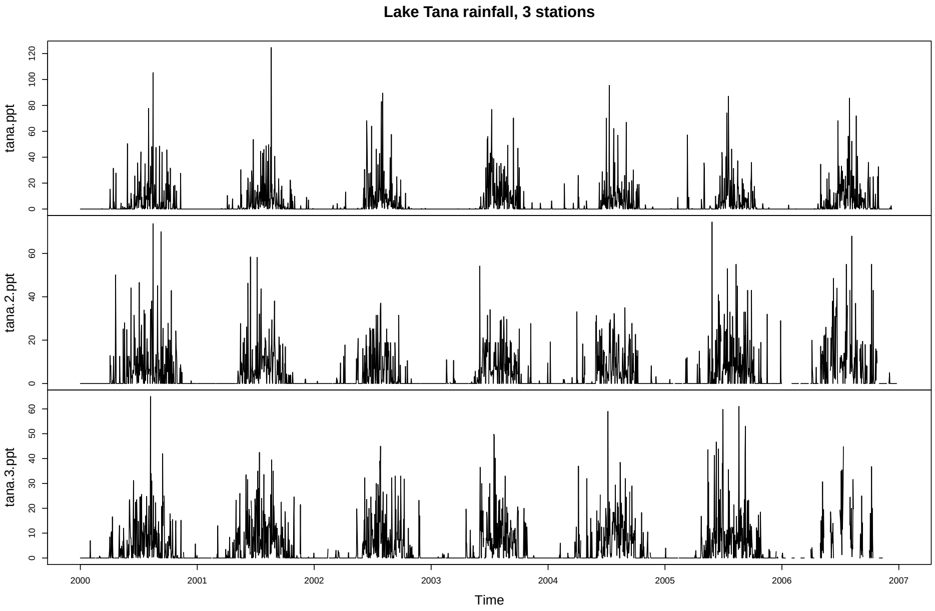
```
## [1] 64
```

```
rm(miss.1, miss.2, miss.3, time.miss.1, time.miss.2, time.miss.3)
```

**Task 92** : Find the common period of the three time series, and plot them on one graph. •

This is the same procedures as in §6, i.e., using the `ts.intersect` function to create a "multiple time series" object of class `mts`:

```
t3 <- ts.intersect(tana.ppt, tana.2.ppt, tana.3.ppt)
str(t3)

##  Time-Series [1:2555, 1:3] from 2000 to 2007: 0 0 0 0 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "tana.ppt" "tana.2.ppt" "tana.3.ppt"

class(t3)

## [1] "mts"    "ts"     "matrix"

plot(t3, main="Lake Tana rainfall, 3 stations")
```



## 7.5 Answers

*A70 : After deletion there are five* NA *(missing) values. The median and mean change slightly because of the deleted values.* *Return to Q70* •

*A71 : Fairly well, especially when the missing value was in a linear section of the graph (steady increase or decrease in depth). However, for March 1997 it missed the high-water stand substantially; this is because that point happens to be a local extremum and not in a linear section of the curve.* *Return to Q71*

•

---

A72 : *The three interpolators are very similar for the gaps in linear sections of the graph; however for the extremum of March 1997 there are substantial differences; both spline interpolators are much closer (about 30 cm) to the true value.* *Return to Q72* •

---

A73 : *For gaps in the linear sections of the graph, both interpolators performed reasonably well. However, with longer gaps around extrema, as in early 1988, both underestimate the extremes. Akima splines performed better than linear interpolation.* *Return to Q73* •

---

A74 : *The interpolators reproduce the overall trend in groundwater depth during the missing year (slightly decreasing); the spline has a very attenuated seasonality as well. But the strong seasonal effect is completely missing.* *Return to Q74* •

---

A75 : *The stations all end in December 2006, but they start in 1981 (Bahir Dar), 1987 (Dangila), and 2000 (Wete Abay). The first and second stations have no missing dates in common, whereas the third station has some of the same missing dates as the other two stations.* *Return to Q75* •

# 8 Simulation

To **simulate** a time series is to generate a series with defined statistical characteristics. This requires a **model** (§4) with a **stochastic** (random) component; the simulation uses random-number generation from various probability distributions. R is particularly strong in this area.

Simulations are used to generate many probable “states of nature”; these are then used as inputs to decision-theoretical models. For example, simulating hundreds of years of rainfall, according to a calibrated model, allows direct evaluation of the probability of an extreme event such as prolonged drought.

Simulation of climatic time series is a key element of synthetic weather generation [19].

## 8.1 AR models

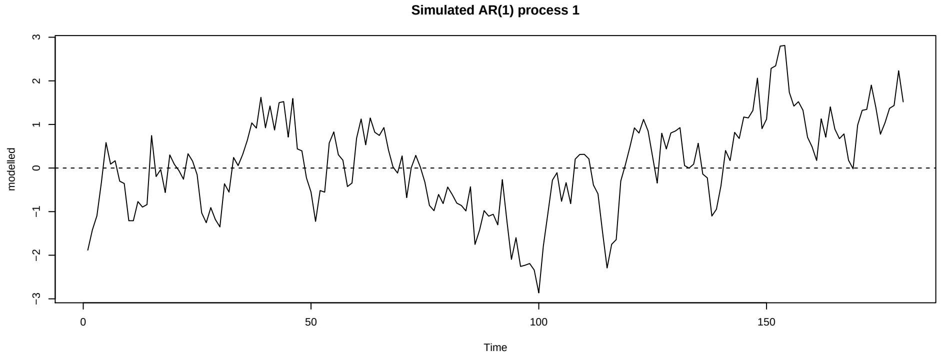
In §4.4.1 we constructed AR(1) and AR(2) models of the groundwater levels, after accounting for trend and seasonality. We can simulate the process with these models, to see how well they reproduce the series. The `arima.sim` function simulates AR, MA, ARMA and ARIMA models. In this case we specify only an AR component.

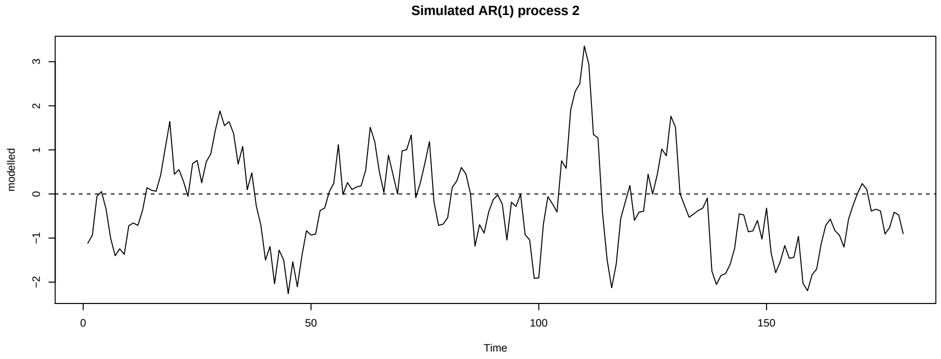
The `arima.sim` function also has an optional `sd` argument to specify the white noise.

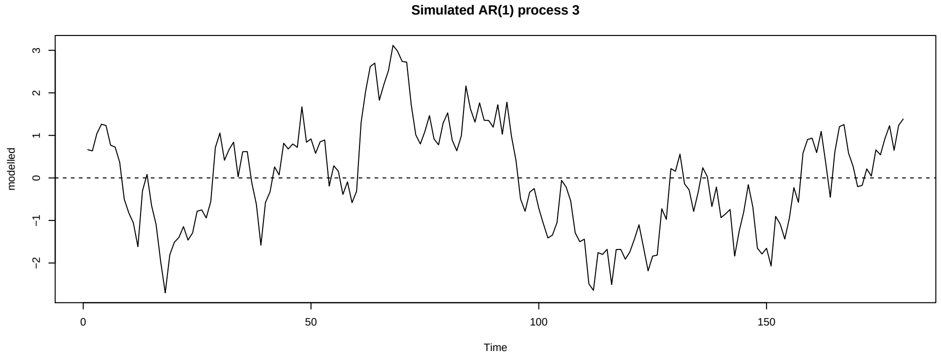
**Task 93 :** Simulate three realizations of fifteen years of groundwater level remainders with the AR(1) model from §4.4.1, and compare with the actual series. •

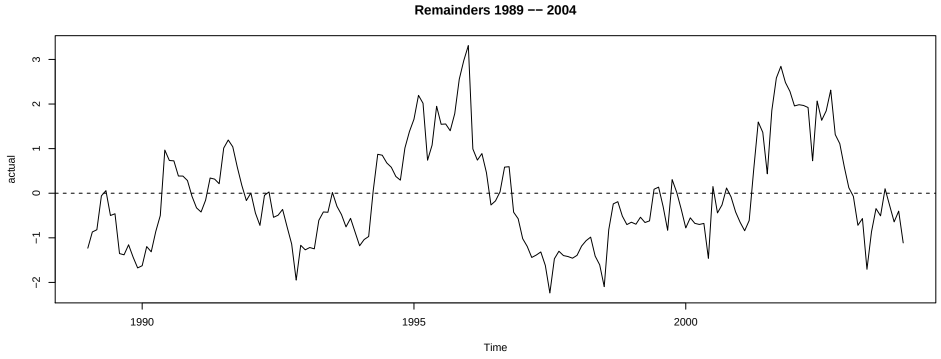
Recall that the fitted AR(1) model is in object `ar.gw.r.1`; the coefficients are in field `ar` of that object, and the red-noise (residual) variance in field `var.pred`:

```
par(mfrow=c(4,1))
for (i in 1:3) {
  plot(arima.sim(model=list(ar=ar.gw.r.1$ar), n=12*15,
                 rand.gen=rnorm, sd=sqrt(ar.gw.r.1$var.pred)),
       main=paste("Simulated AR(1) process",i), ylab="modelled")
  abline(h=0, lty=2)
}
plot(window(gw.r, 1989, 1989+15),
     main="Remainders 1989 -- 2004", ylab="actual")
abline(h=0, lty=2)
par(mfrow=c(1,1))
```







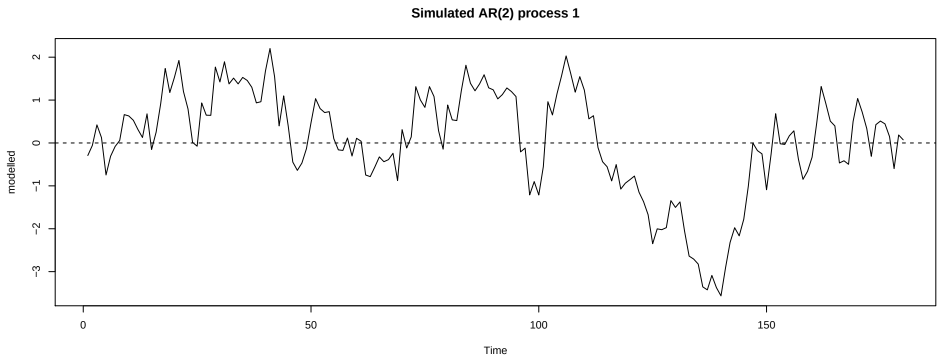


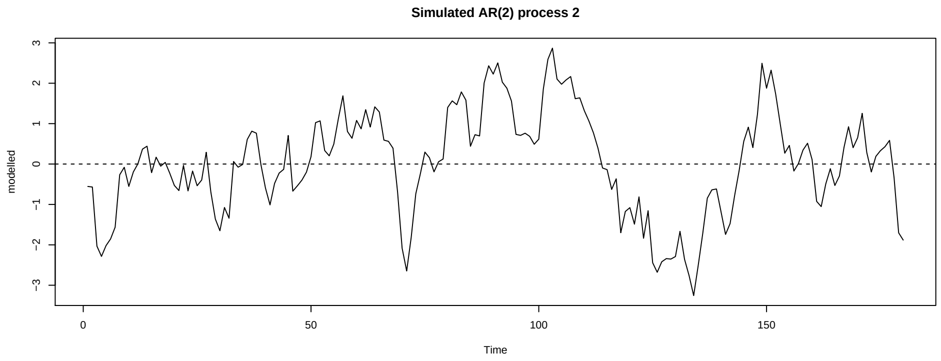
**Q76** : *How well do the simulations reproduce the structure of the actual series?* *Jump to A76* •

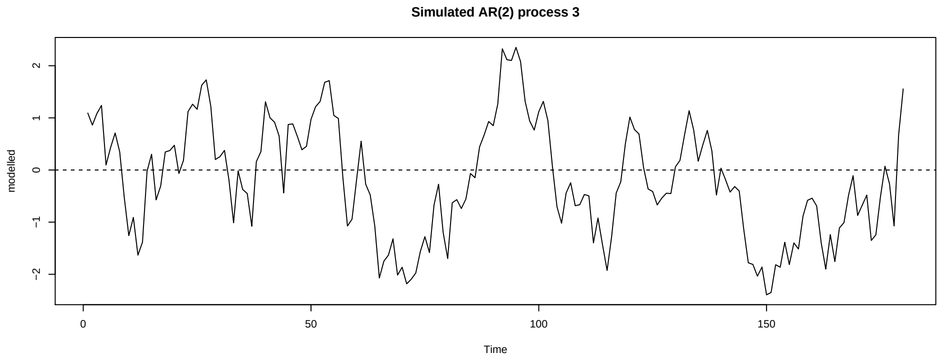
**Task 94 :** Repeat for the fitted AR(2) model. •

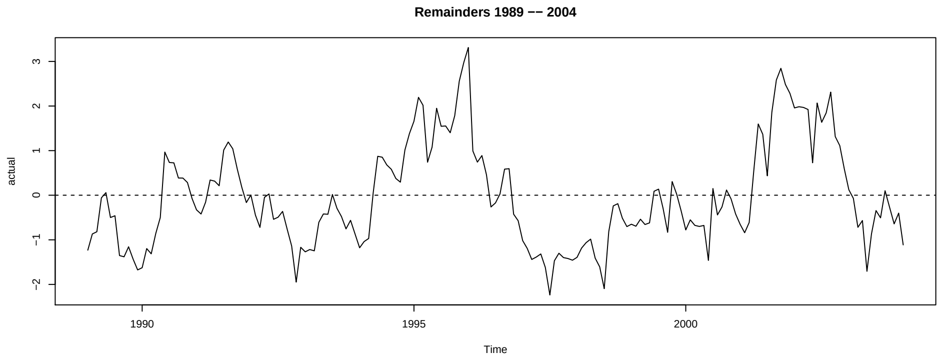
The fitted AR(2) model is in object `ar.gw.r`.

```r
par(mfrow=c(4,1))
for (i in 1:3) {
  plot(arima.sim(model=list(ar=ar.gw.r$ar), n=12*15,
                 rand.gen=rnorm, sd=sqrt(ar.gw.r$var.pred)),
  main=paste("Simulated AR(2) process",i), ylab="modelled")
  abline(h=0, lty=2)
}
plot(window(gw.r, 1989, 1989+15),
     main="Remainders 1989 -- 2004", ylab="actual")
abline(h=0, lty=2)
par(mfrow=c(1,1))
```









**Q77** : *How well do the AR(2) simulations reproduce the structure of the actual series?* *Jump to A77* •

## 8.2 Answers

**A*76*** : *The AR(1) simulations seem somewhat noisier than that actual series. The scale seems correct.* *Return to Q76* •

**A*77*** : *The AR(2) simulations seem to match the actual series better.* *Return to Q77* •

## References

[1] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM*, 17(4):589–602, 1970. 112

[2] D Bates. Fitting linear mixed models in R. *R News*, 5(1):27–30, 2005. 63

[3] David Birkes and Yadolah Dodge. *Alternative methods of regression*. John Wiley & Sons, Inc., New York, 1993. 72

[4] George E. P. Box. *Time series analysis: forecasting and control*. John Wiley & Sons, Inc., fifth edition / edition, 2016. ISBN 978-1-118-67492-5. URL http://ebookcentral.proquest.com/lib/cornell/detail.action?docID=2064681. 1

[5] George E.P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time series analysis: forecasting, and control*. Prentice-Hall, Englewood Cliffs, NJ, 3rd edition, 1994. 76

[6] William S. Cleveland and Susan J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, Sep 1988. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.1988.10478639. 32

[7] JC Davis. *Statistics and data analysis in geology*. John Wiley & Sons, New York, 3rd edition, 2002. 1

[8] Peter J. Diggle. *Time series: a biostatistical introduction*. Oxford Statistical Science Series; 5. Oxford Science Publications, Oxford, 1989. 1

[9] John Fox. *An R and S-Plus Companion to Applied Regression*. Sage Publications, Thousand Oaks, CA, USA, 2002. 15

[10] Keith W. Hipel and A. Ian McLeod. *Time series modelling of water resources and environmental systems*. Number 45 in Developments in Water Science. Elsevier, 1994. ISBN 9780444892706. URL http://www.stats.uwo.ca/faculty/aim/1994Book/. 1, 72, 98

[11] Robert M. Hirsch, James M Slack, and Richard A Smith. Techniques of trend analysis for monthly water quality data. *Water Resources Research*, 18(1):107–121, 1982. 72, 73

[12] R Ihaka and R Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. 1

[13] Andrew V. Metcalfe and Paul S.P. Cowpertwait. *Introductory Time Series with R*. Use R! Springer, 2009. DOI: 10.1007/978-0-387-88698-5. 1

[14] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna,

Austria, 2004. URL http://www.R-project.org. ISBN 3-900051-07-0. 1

[15] Jose D. Salas. Analysis and modeling of hydrologic time series. In David R. Maidment, editor, *Handbook of hydrology*, pages 19.1–19.72. McGraw-Hill, New York, 1993. 1, 107

[16] Robert H. Shumway and David S. Stoffer. *Time series analysis and its applications : with R examples*. Springer, New York, 2nd edition, 2006. 1

[17] R Development Core Team. *R Data Import/Export*. The R Foundation for Statistical Computing, Vienna, version 2.9.0 (2009-04-17) edition, 2009. URL http://cran.r-project.org/doc/manuals/R-data.pdf. 12

[18] WN Venables and BD Ripley. *Modern applied statistics with S*. Springer-Verlag, New York, fourth edition, 2002. 1

[19] D S Wilks and R L Wilby. The weather generation game: a review of stochastic weather models. *Progress in Physical Geography*, 23(3): 329–357, 1999. 125

[20] Daniel S. Wilks. *Statistical methods in the atmospheric sciences*. International Geophysics Series 59. Academic Press, New York, 1995. 1, 79, 82

## Index of R Concepts