Predictive Modeling

Series 11

Exercise 11.1

The Swiss Federal Statistical Office provides a huge number of data sets on their web-page. Through the tool *STAT-TAB* it is easy to browse through the data sets and download specific tables that can be composed individually.

a) Visit the web-page
https://www.bfs.admin.ch/bfs/en/home/services/recherche/stat-tab-online-data-search.html
and get accustomed to the data browser.

Hint: You may have a look at the *Guideline for online data search* which can be downloaded at the bottom of the page.

b) The majority of the data sets on this website are infact time series. Try to find the time series containing the number of electric passenger cars (PW) from 1990 - 2016 (for each canton separately). Download the resulting table as a .csv file

c) Load the file into **R** and define a times series for the number of electric cars in Luzern. Plot the time series.

Hint: The files generated by **STAT-TAB** may contain additional information aside to the regular header that displays the variable names. Look at the .csv file in some text editor (e.g. in **Rstudio**) and figure out how many lines of text are preceeding the data (excluding the header). Then you can read the table with the following **R** command (in this example two lines before the header are skipped)

```
PW_data = read.table("./Daten/PW_electrisch.csv",
                        header=T, sep = ";", skip = 2)
PW_LU_ts = ts(as.numeric(PW_data[3, 3:29]),
                start = 1990, frequency = 1)
plot(PW_LU_ts, main="Electric cars in canton Luzern")
```

d) Repeat the procedure in c) for different cantons.

e) What would be a proper way to compare the data between different cantons?

Exercise 11.2

In this section we study the quarterly beer production data.

a) Read the file `AustralianBeer.csv` into`R` and coerce the data into a time series. Plot the data.

   Hint: Have a look at Example 13.2.3. for the syntax to transform the data to a time series in `R`

```
Beer = read.csv("Daten/AustralianBeer.csv", sep=";")
```

b) Plot the aggregated annual series and a boxplot that summarises the observed values for each season, and comment on the plots.

   Hint: Copy and adapt the code in Example 13.3.5. in order to generate the boxplots.

c) Decompose the series into the components trend, seasonal effect, and residuals by means of the `decompose()` command. Comment on the results. Do you think that a data transformation would be necessary before the decomposition?

d) Decompose the series again, this time using `stl()`. Choose a proper value for `s.window`. Compare your results to the one in c).

e) Use the `monthplot()` function to study the change of seasonality in the 4 different quarters.

Exercise 11.3

In this section we study the quarterly electricity production data as presented in the script.

a) Read the file `AustralianElectricity.csv` into `R` and generate a time series. Plot the data.

b) Find an appropriate data transformation such that the variance of the time series is stabilized.

   Hint: Use the `box.cox` function and find an optimal value for `lambda` by visually inspecting the time plot of the transformed series. This is the code that does the trick for `lambda = 1.3`

```r
box.cox <- function(x, lambda) {
  if (lambda==0) log(x) else (x^lambda - 1)/lambda
}

# replace "yourSeries" by the name of your series
yourSeries.tr = box.cox(yourSeries, 1.3)
plot(yourSeries.tr)
```

c) Decompose the series into the components trend, seasonal effect, and residuals by means of the `decompose()` command. Comment on the results.

d) Decompose the series again, this time using `stl()` . Choose a proper value for `s.window`. Compare your results to the one in c).

e) Use the `monthplot()` function to study the change of seasonality in the 4 different quarters.

Result Checker

Predictive Modeling

Solutions to Series 11

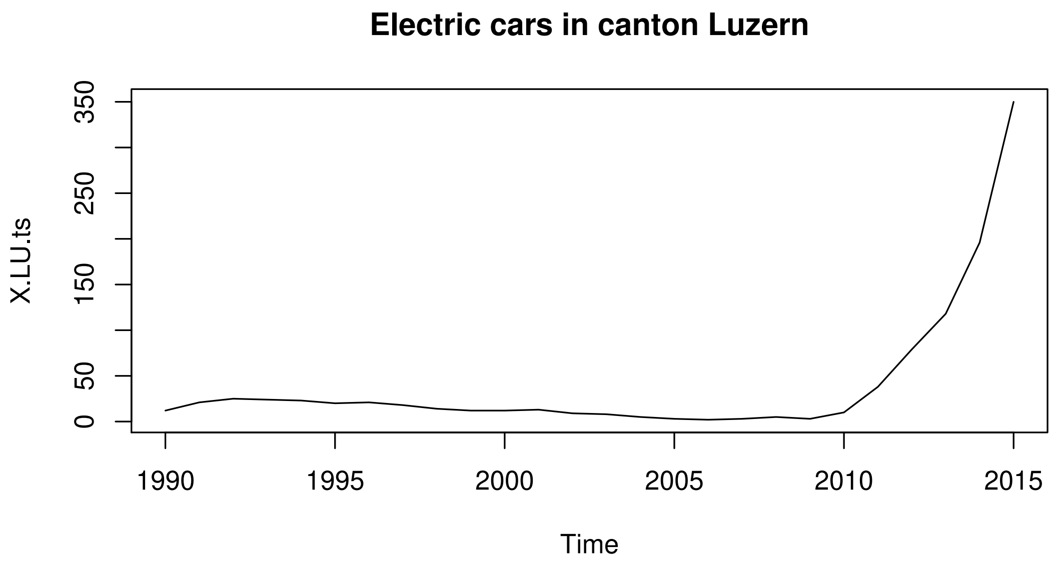
Solution 11.1

a) -

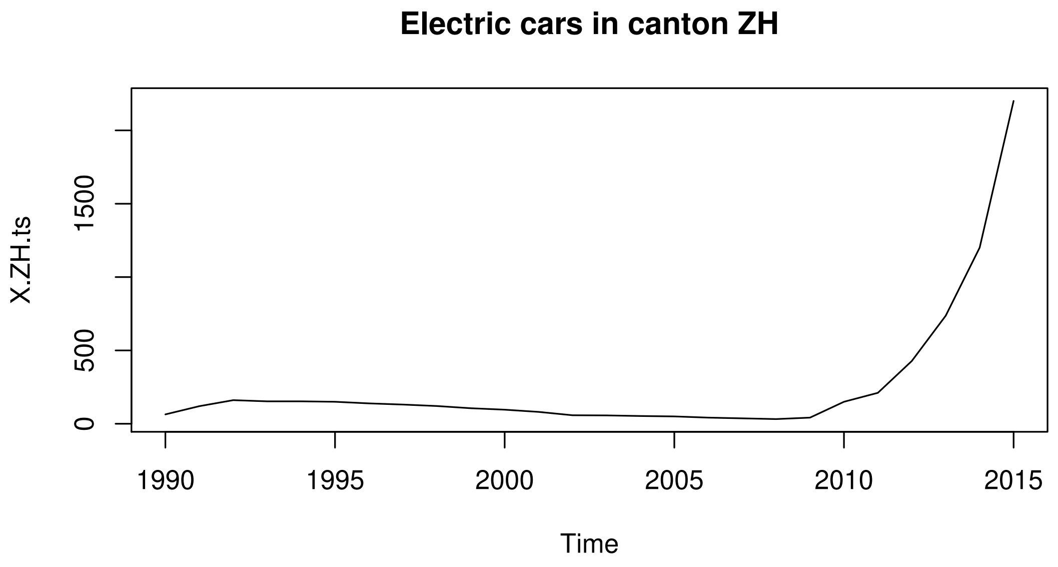
b) -

c) The file with the electricity car data is stored in the file **PW_electric.csv**. We load the file with the **read.table()** function.

```
X = read.table("Daten/PW_electric.csv", sep=";",
                              header=T, skip=2)

# get rid of the Treibstoff variable (it is constant)
X = X[,-2]

# extract the Zurich data (first row)
X.LU = as.numeric(X[3, 2:27])

# generate a time series
X.LU.ts = ts(X.LU, start = 1990, end = 2015, frequency = 1)
plot(X.LU.ts, main="Electric cars in canton Luzern")
```



d)
```
# extract the Zurich data (first row)
X.ZH = as.numeric(X[1, 2:27])

# generate a time series
X.ZH.ts = ts(X.ZH, start = 1990, end = 2015, frequency = 1)
plot(X.ZH.ts, main="Electric cars in canton ZH")
```

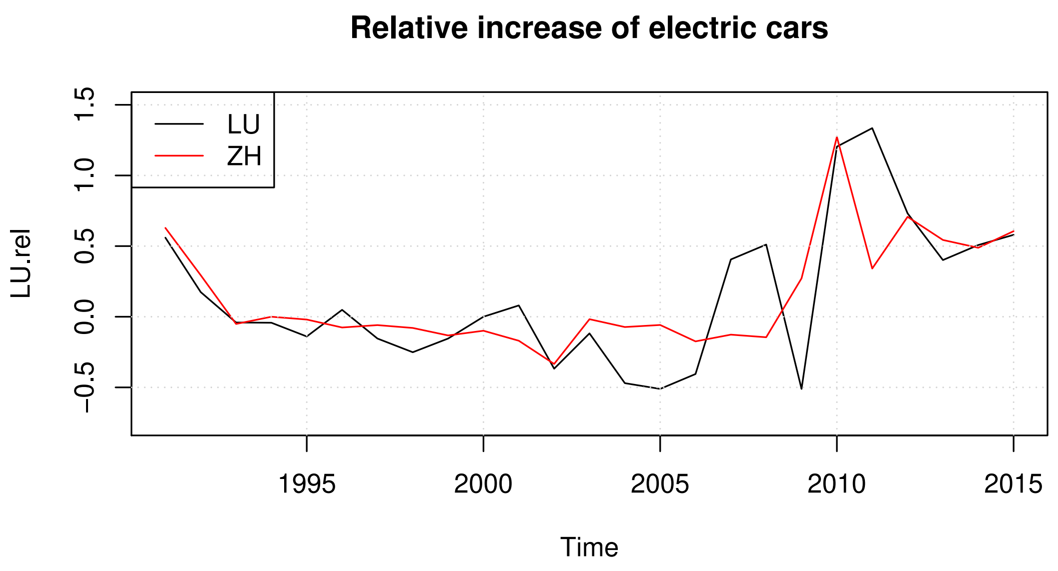


e) A proper way to compare the number of electric cars between two cantons would be to normalize the data by the total number of vehicles. If this additional data is not at hand, one could also compare the *relative increase* of the number of electric vehicles. As we have noted in the lectures, we can estimate the relative increase by the difference of logarithms:

$$\log(X_k) - \log(X_{k-1}).$$

The lagged series can be retrieved in **R** with the **lag()** command.

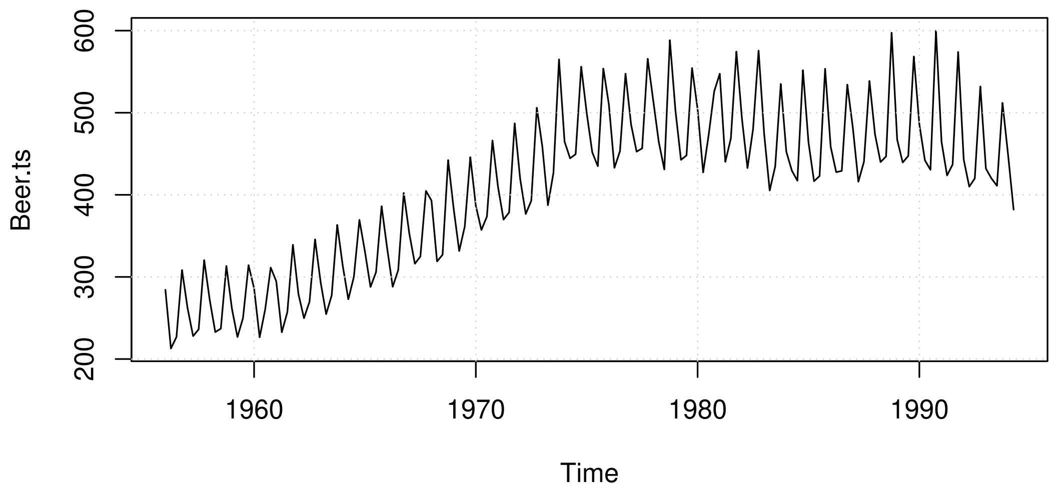
```
LU.rel = log(X.LU.ts) - log(lag(X.LU.ts,-1))
plot(LU.rel, main="Relative increase of electric cars",
     ylim=c(-0.75, 1.5))

ZH.rel = log(X.ZH.ts) - log(lag(X.ZH.ts,-1))
lines(ZH.rel, col="red")
grid()
legend("topleft", legend=c("LU", "ZH"), col = c("black", "red"), lty=
```



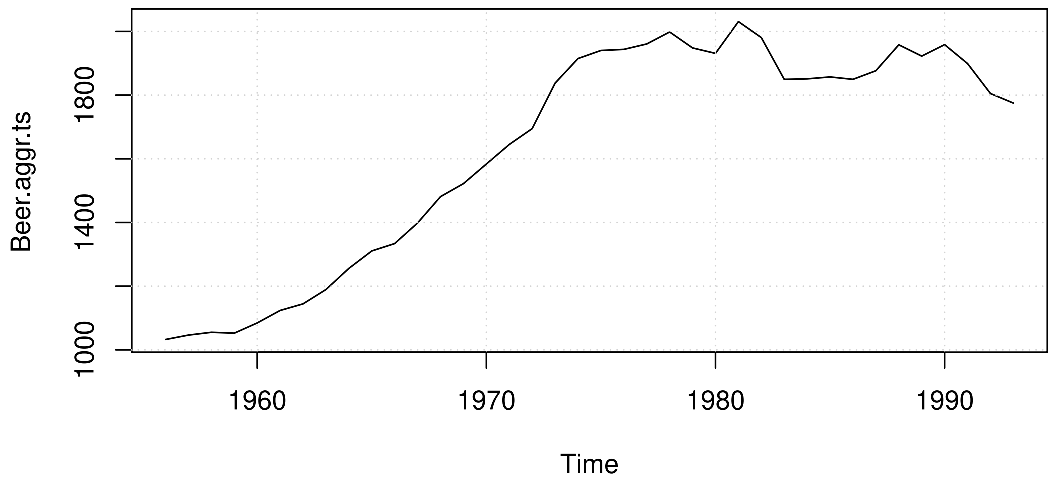
This plot can be interpreted that the main hype for buying electric cars has arrived in the same year. The large relative increase lasted for one more year in Luzern as compared to Zürich.

Solution 11.2

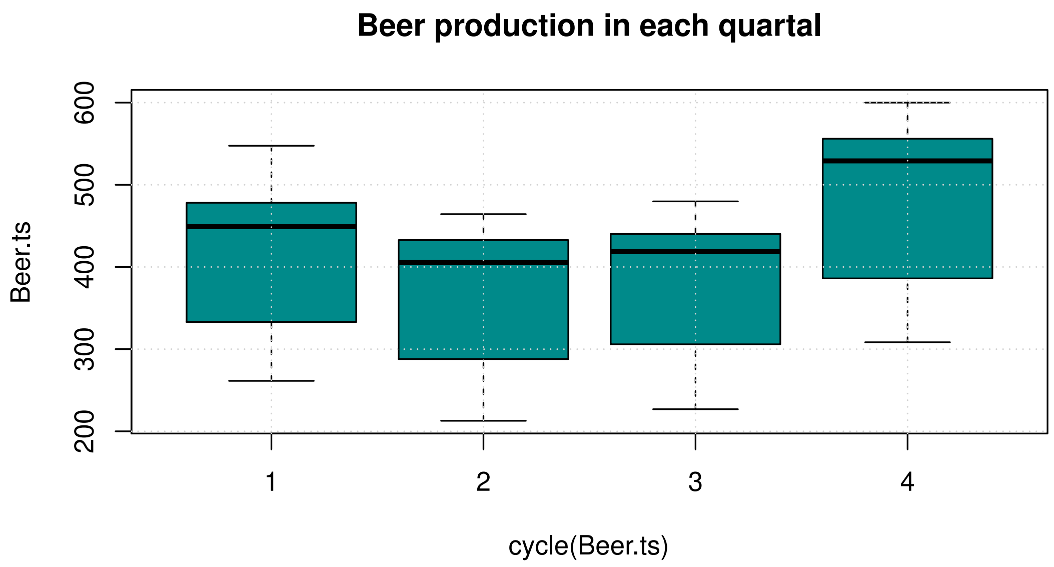
a)
```
Beer = read.csv("Daten/AustralianBeer.csv", sep=";")
Beer.ts = ts(Beer[,2], start = c(1956,1), end = c(1994,2), frequency
plot(Beer.ts)
grid()
```



b)
```r
# aggregate the data
Beer.aggr.ts = aggregate(Beer.ts, nfrequency = 1)
plot(Beer.aggr.ts)
grid()
```

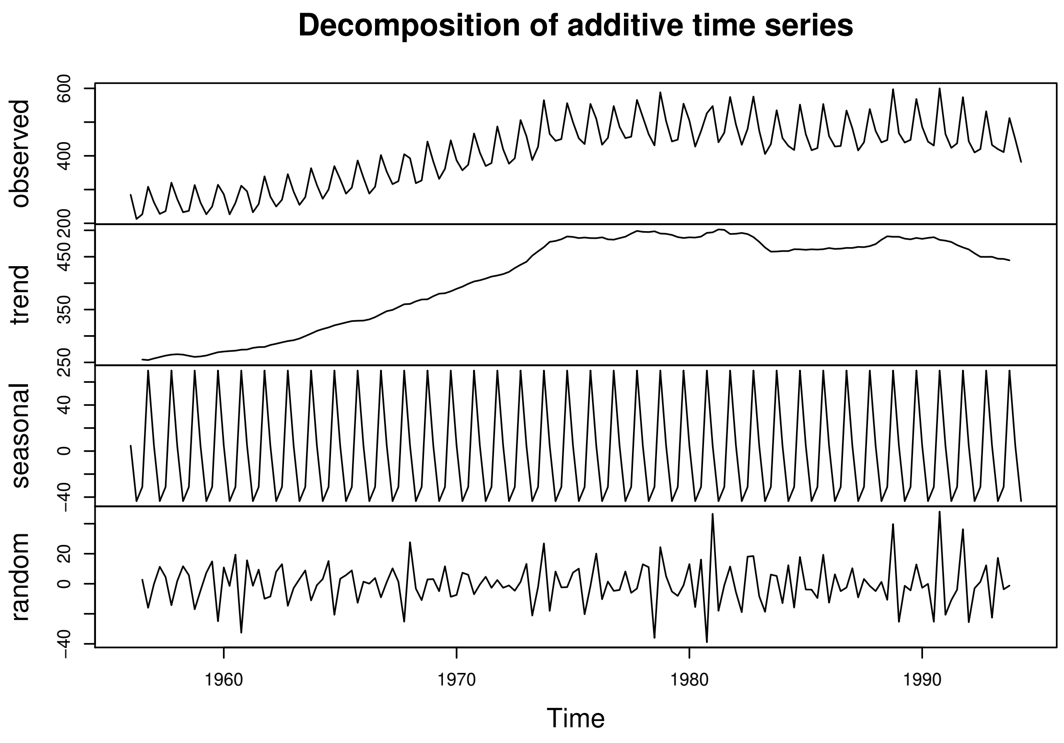


```r
# boxplot the data
boxplot(Beer.ts ~ cycle(Beer.ts), col = "darkcyan",
        main="Beer production in each quartal")
grid()
```



c) The time series plot in a) indicates that the seasonal effect is quite stable over the observation period, i.e. the variance over time is constant. So decomposition without any transformations is reasonable.

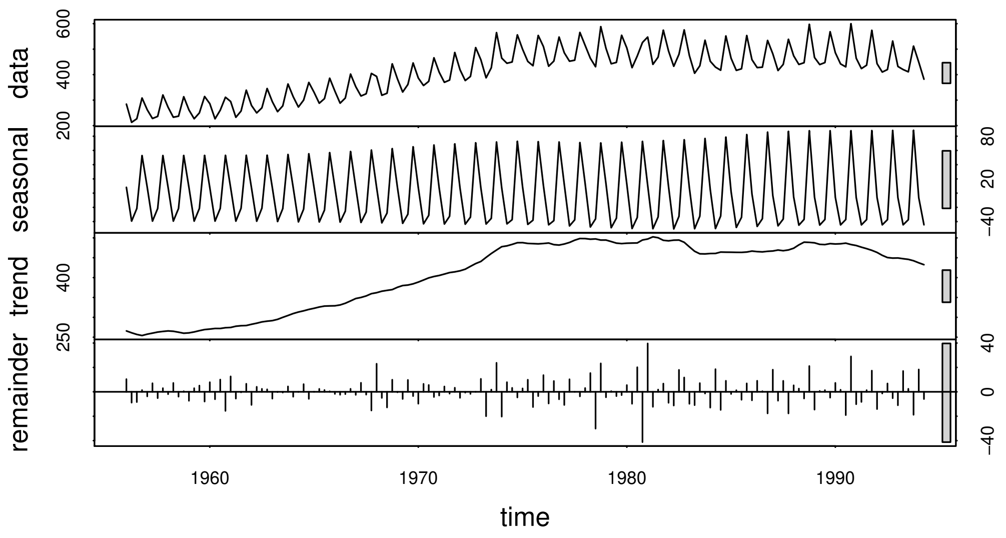
```r
res.dec = decompose(Beer.ts)
plot(res.dec)
```



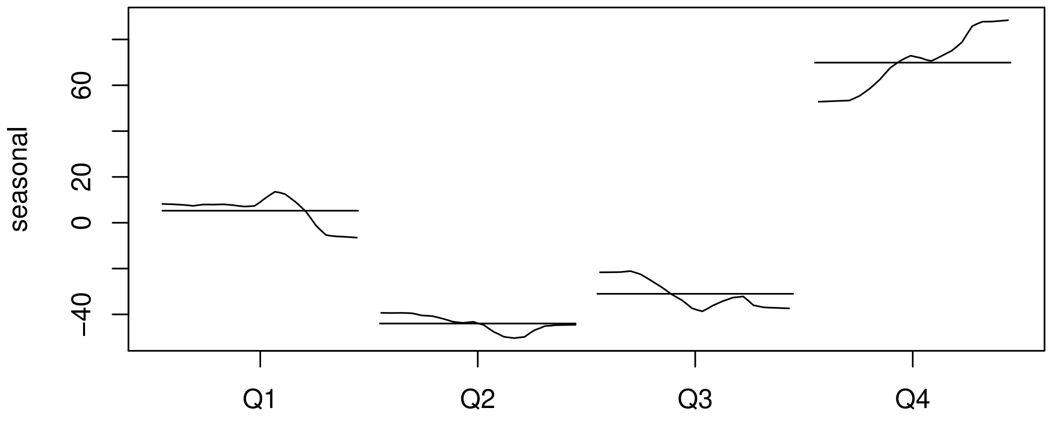
The remainder series confirms that the series is well decomposable without any further transformation: hardly any periodic pattern is visible and the variance seems stable.

d) Experimenting with the `s.window` parameter and doing `monthplots` iteratively shows that a value about 15 could be reasonable

```r
res.stl = stl(Beer.ts, s.window = 15)
plot(res.stl)
```



```r
monthplot(res.stl)
```

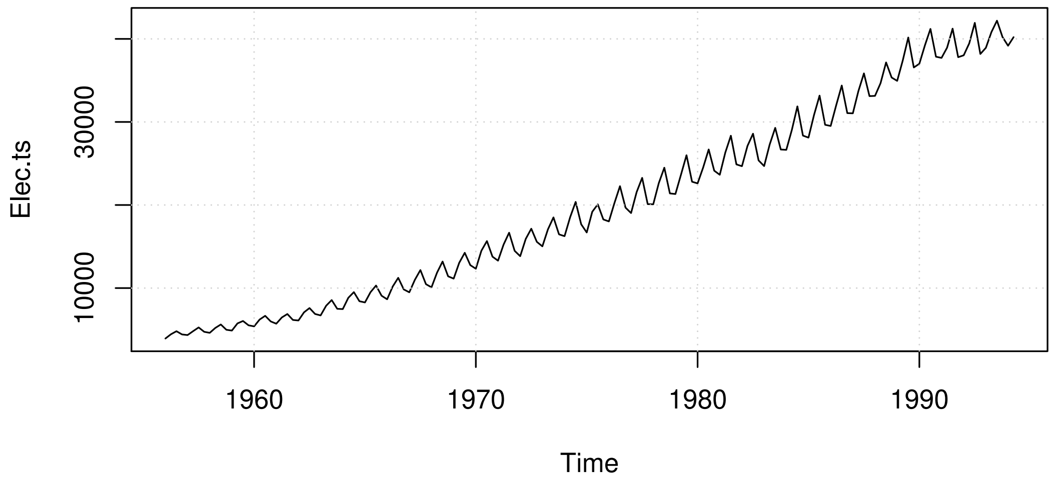


The monthplot indicates that the seasonal effect of beer production has been decreasing for the first quarter and increasing for the last quarter. The remaining quarters show a quite stable seasonal impact.

Solution 11.3

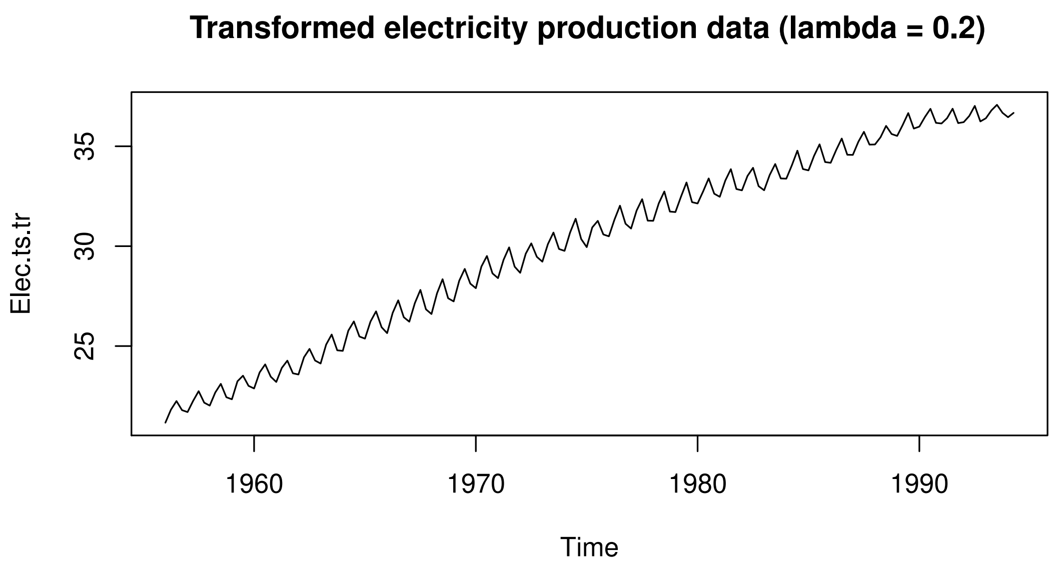
a)
```r
Elec = read.table("./Data/AustralianElectricity.csv",
                  sep = ";", header=T)
```

```r
setwd(wd)
plot(Elec.ts)
grid()
```



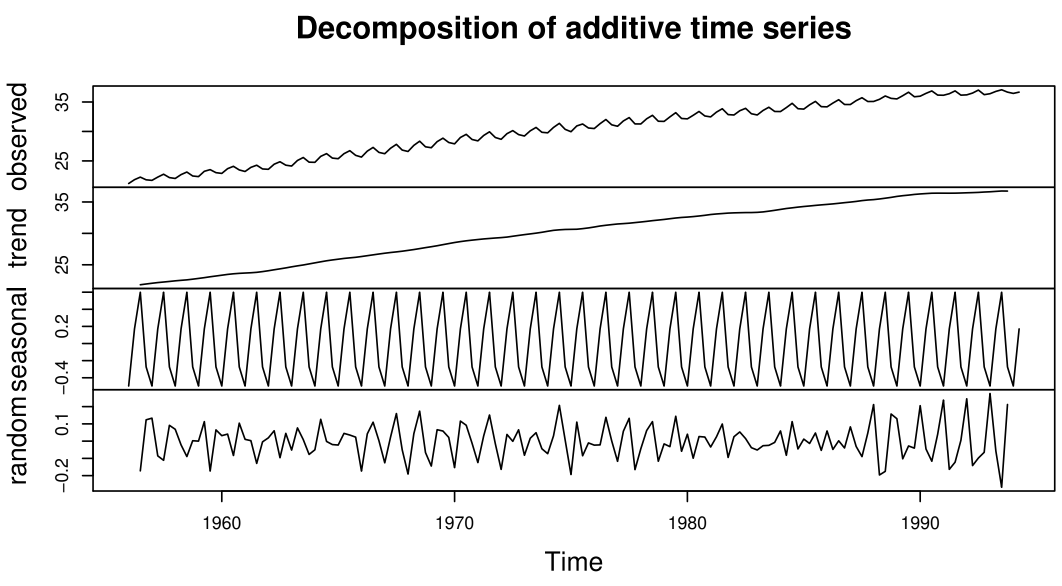
b) It is obvious that the variance at the beginning of the recordings is smaller than at the end. A value of 0.2 – 0.3 for the Box-Cox parameter does a good job to stabilize the variance.

```r
box.cox <- function(x, lambda) {
  if (lambda==0) log(x) else (x^lambda - 1)/lambda
}

Elec.ts.tr = box.cox(Elec.ts, 0.2)
plot(Elec.ts.tr,
main="Transformed electricity production data (lambda = 0.2)")
```



c) We decompose the time series by means of the basic `decompose()` function after trasforming it as in b)

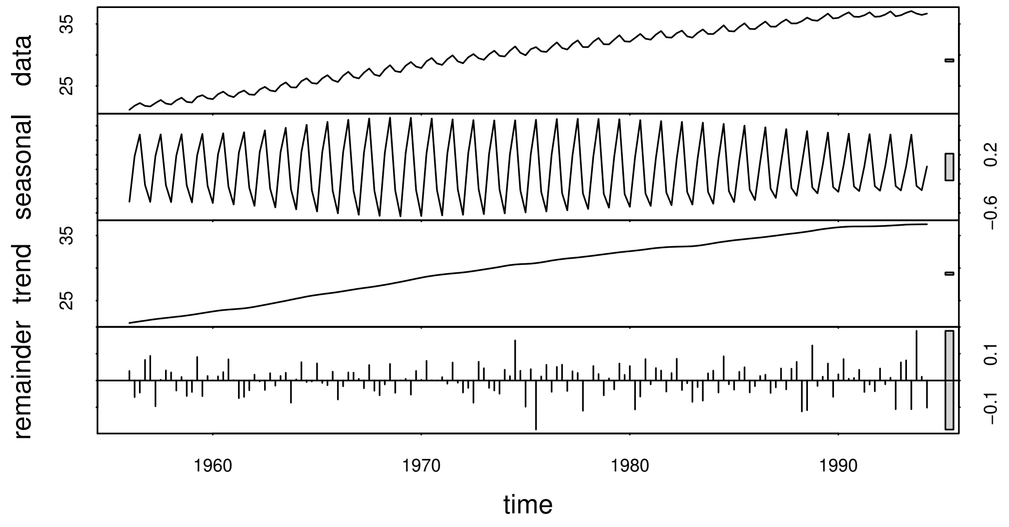
```
res.dec = decompose(Elec.ts.tr)
plot(res.dec)
```



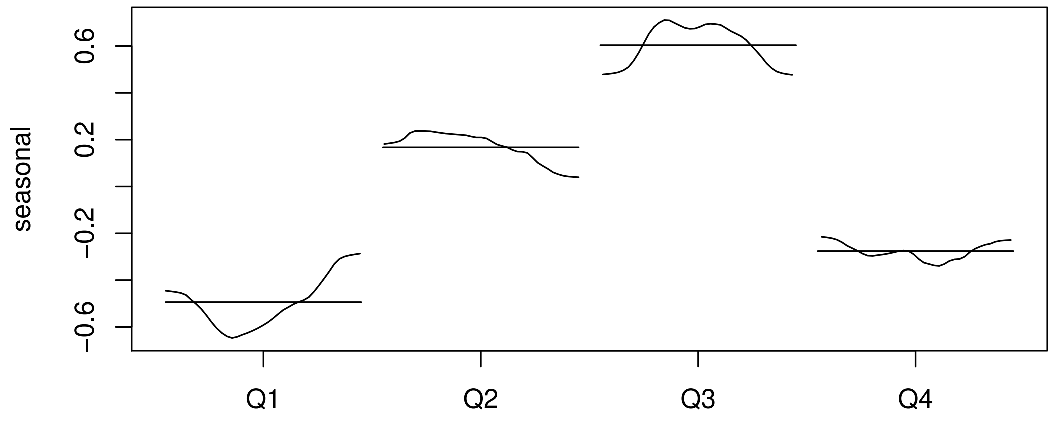
The remainder series confirms that the series is reasonably well decomposable after the transformation: hardly any periodic pattern is visible and the variance seems stable. However, at the end of the time series a seasonal pattern becomes evident.

d) The result in c) shows that the seasonal effects are varying over time (even if we try to transform the data properly) Experimenting with the **s.window** parameter and doing **monthplots** iterativelyshows that a value about 10 could be reasonable

```
res.stl = stl(Elec.ts.tr, s.window = 10)
plot(res.stl)
```



```
monthplot(res.stl)
```



Again, we see that the remainder sequence in the STL decomposition has a purely random appearance and that the monthplots produces reasonably smooth changes in the quarterly contribution to seasonality.