Musterlösungen zu Serie 11

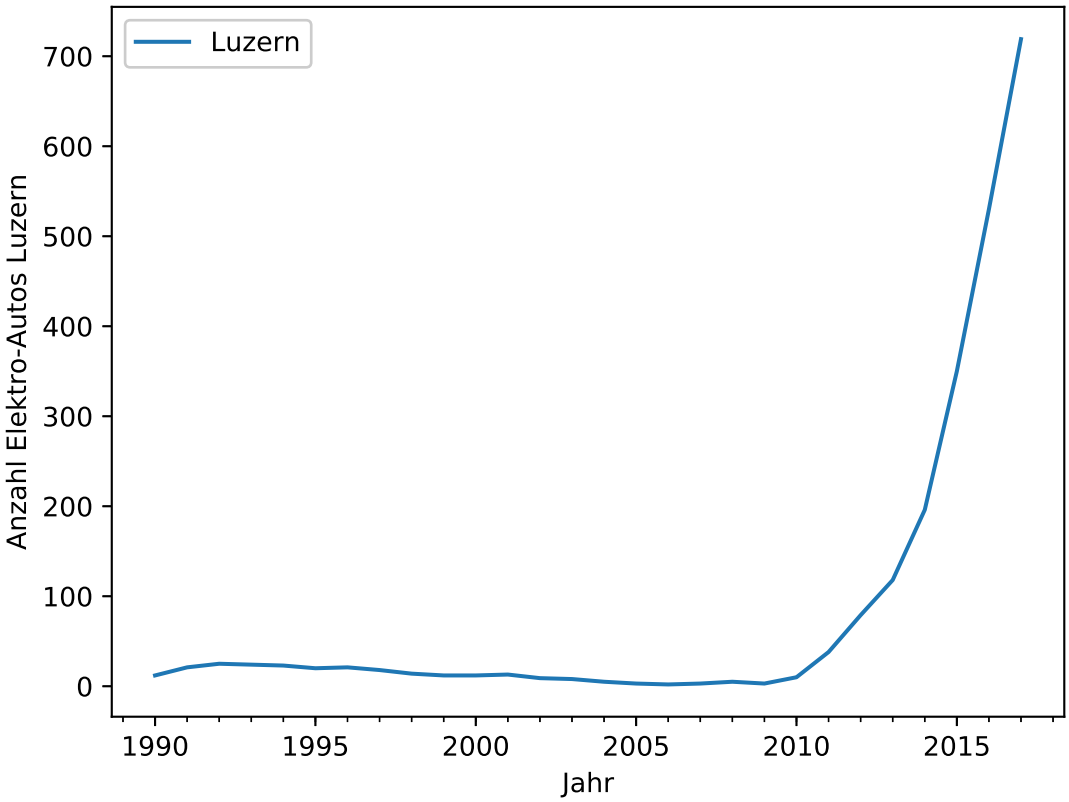
Lösung 11.1

a) -

b) -

c) Die Datei mit den Elektro-Auto-Daten sind in der Datei `PW_electric.csv` abgespeichert. Wir lesen die Datei mit der Funktion **`pd.read_csv()`** ein. (zu **R**)

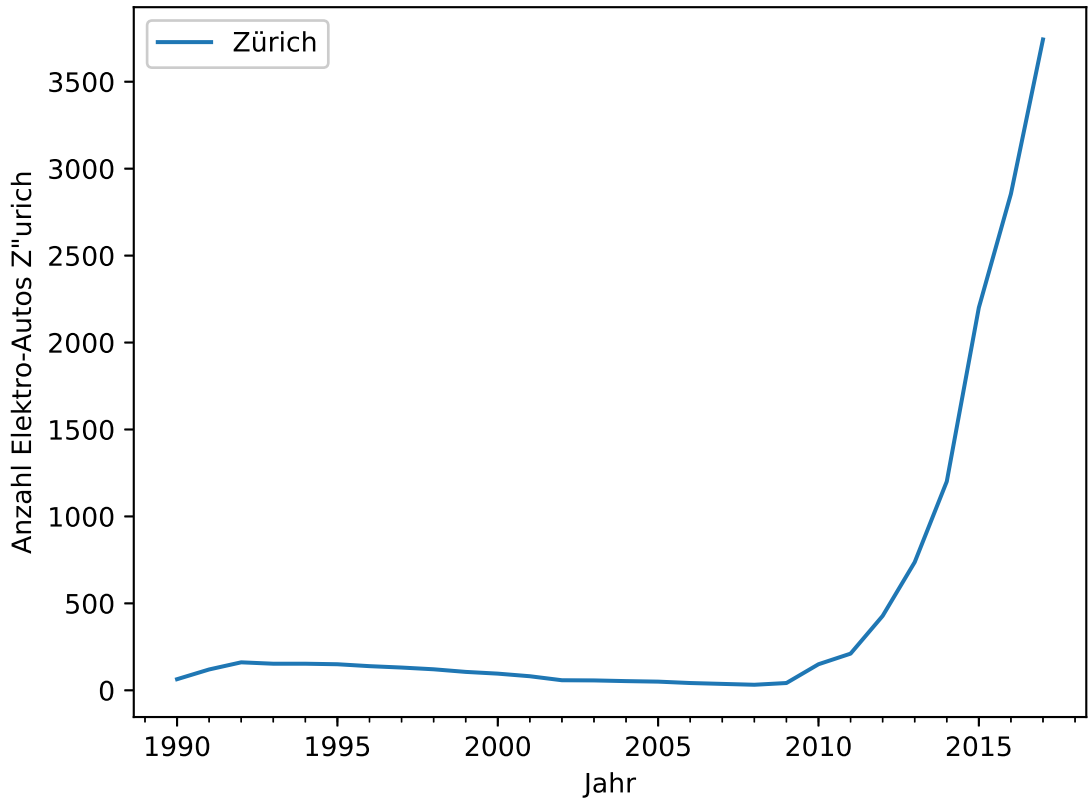
```
import matplotlib.pyplot as plt
import pandas as pd
from pandas import DataFrame
import numpy as np
pw_electric = pd.read_csv('.../PW_electric.csv', sep=',',
                          skiprows=2, header=0,
                          encoding='utf-8',
                          index_col=0)
pw_electric.head()
pw_electric_luzern = DataFrame(pw_electric.ix["Luzern",1:])
pw_electric_luzern
pw_electric_luzern["Year"] = pd.DatetimeIndex(pw_electric_luzern.index)
pw_electric_luzern.set_index("Year", inplace=True)
pw_electric_luzern.plot()
plt.xlabel("Jahr")
plt.ylabel("Anzahl Elektro-Autos Luzern")
plt.show()
```



d) (zu R)

```
import matplotlib.pyplot as plt
import pandas as pd
from pandas import DataFrame
import numpy as np

pw_electric = pd.read_csv('.../PW_electric_90to17.csv', sep=',',
                          skiprows=2, header=0,
                          encoding = "ISO-8859-1",
                          index_col=0)
pw_electric.head()
pw_electric_zurich = DataFrame(pw_electric.ix["Z\"urich",1:])
pw_electric_zurich
pw_electric_zurich["Year"] = pd.DatetimeIndex(pw_electric_zurich.index)
pw_electric_zurich.set_index("Year", inplace=True)
pw_electric_zurich.plot()
```

```
plt.xlabel("Jahr")
plt.ylabel("Anzahl Elektro-Autos Z\"urich")
```

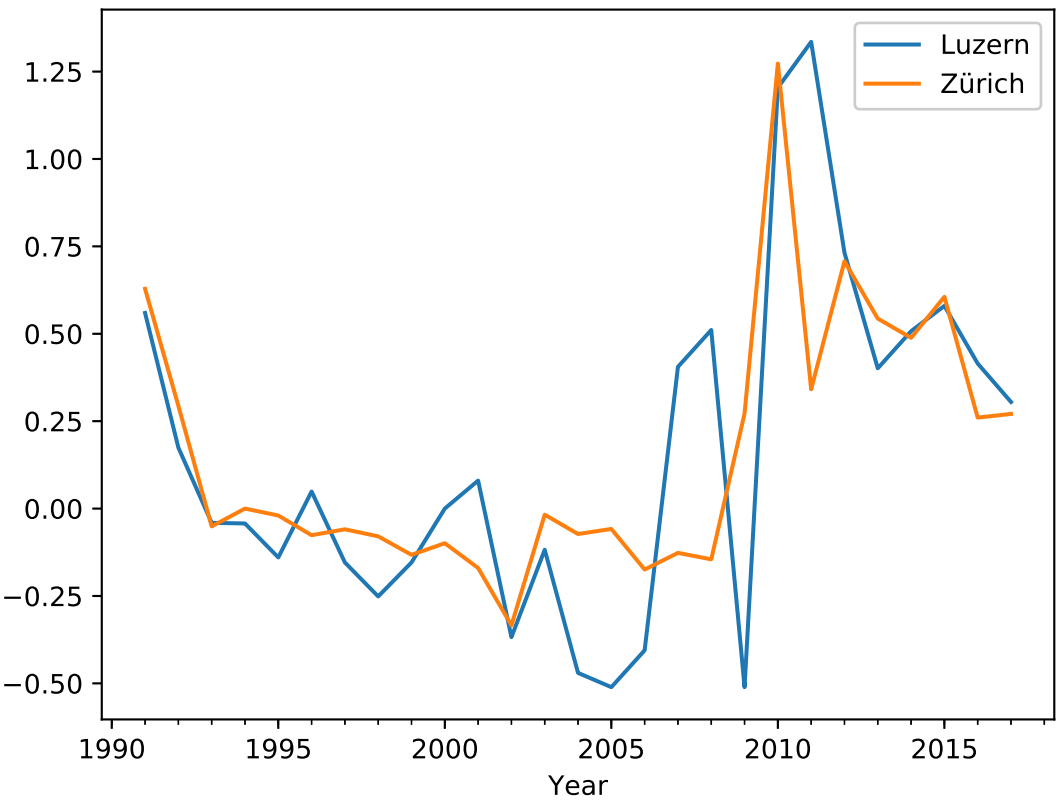


e) Um die Anzahl der Elektro-Autos zwischen den Kantonen auf faire Art und Weise miteinander zu vergleichen, müssten die Daten mit der Gesamtzahl von Autos normalisiert werden. Falls diese zusätzlichen Informationen nicht vorhanden sind, dann ist es angebracht, den *relativen Zuwachs* der Anzahl von Elektro-Autos zu vergleichen. Wie im Unterricht behandelt wurde, können wir den relativen Zuwachs durch die Differenz der Logarithmen schätzen:

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

Die verschobene Zeitreihe kann in **Python** mit Hilfe der Methode **.shift()** bestimmt werden. (zu **R**)

```
import matplotlib.pyplot as plt
import pandas as pd
from pandas import DataFrame
import numpy as np
# Relativer Zuwachs in Luzern
pw_electric_luzern["rel"] = np.log(pw_electric_luzern.astype('float'))
- np.log(pw_electric_luzern.shift(1).astype('float'))
# Relativer Zuwachs in Zuerich
pw_electric_zurich["rel"] = np.log(pw_electric_zurich.astype('float'))
- np.log(pw_electric_zurich.shift(1).astype('float'))
pw_rel = pd.DataFrame({"Luzern" : pd.Series(pw_electric_luzern["rel"]),
  "Z<U+FFFD><U+FFFD>rich" : pd.Series(pw_electric_zurich["rel"])})
pw_rel.plot()
```



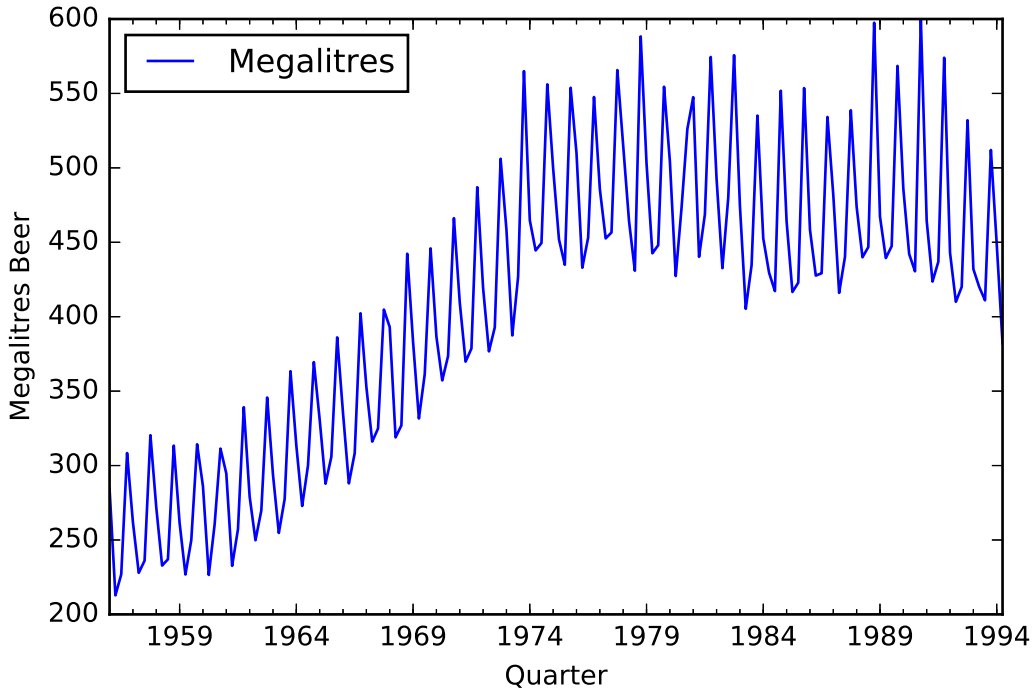
Die Graphik kann dahingehend interpretiert werden, dass sich der Hype um die Käufe von elektrischen Autos in Zürich ein Jahr früher im Vergleich zu Luzern

abzeichnete, nämlich im Jahr 2009, der Durchbruch hat sich aber in beiden Kantonen im Jahr 2010 etwa gleich zugetragen.

Lösung 11.2

a) (zu R)

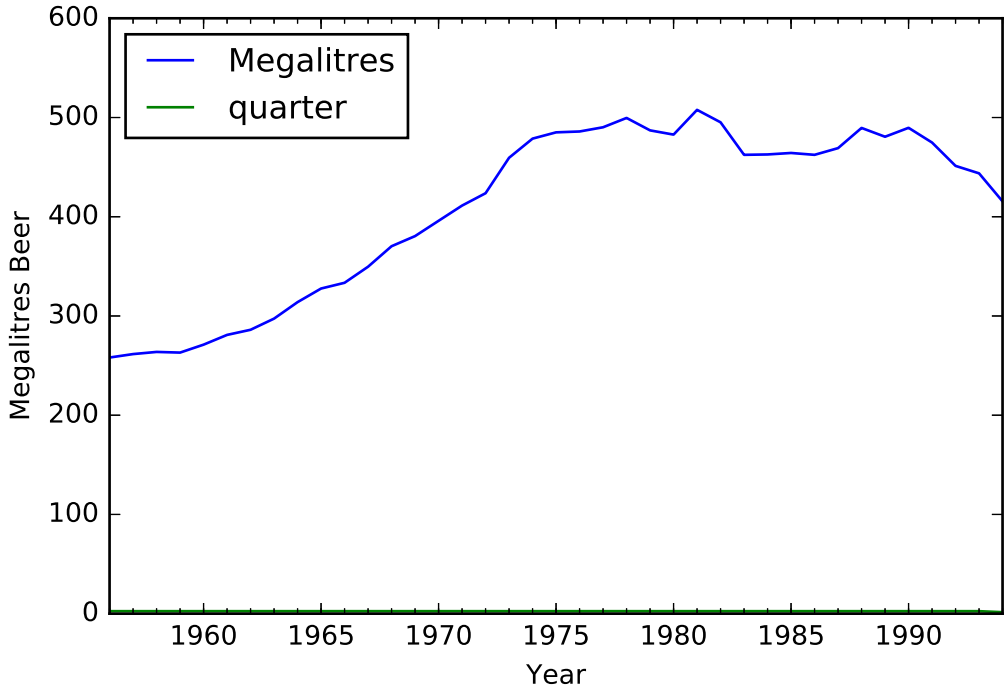
```
AusBeer = pd.read_csv(".../AustralianBeer.csv",sep=";",header=0)
AusBeer.head()
AusBeer["Quarter"] = pd.DatetimeIndex(AusBeer["Quarter"])
AusBeer.set_index("Quarter", inplace=True)
AusBeer.columns=["Megalitres"]
AusBeer.head()
AusBeer.describe()
AusBeer.plot()
plt.ylabel("Megalitres Beer")
```



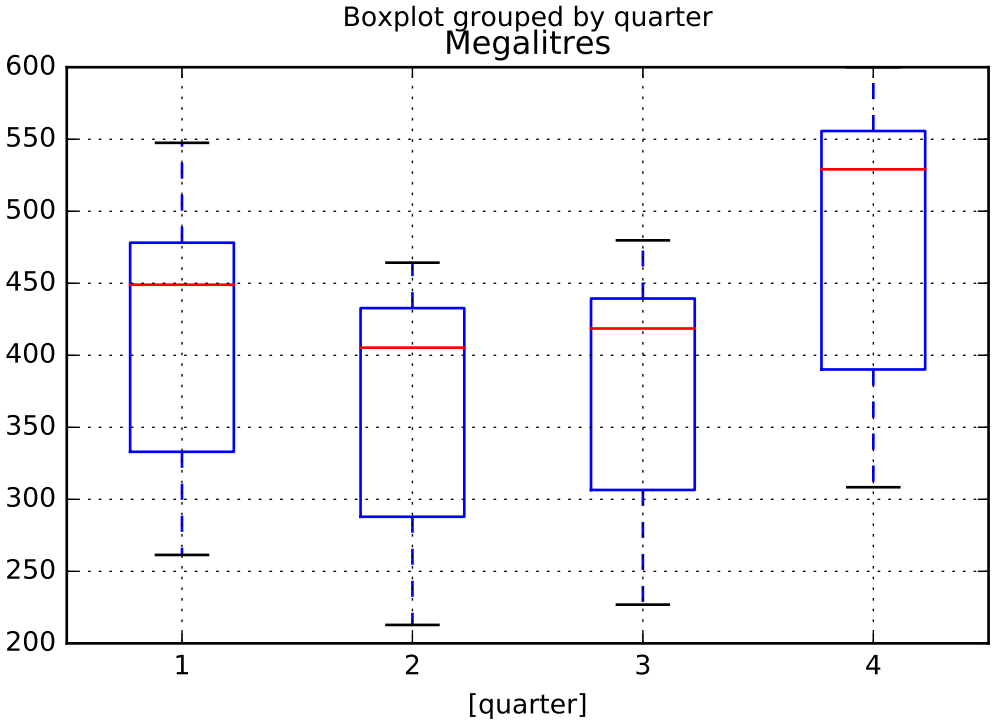
b) (zu R)

```
AusBeer.head()
AusBeer.describe()
AusBeer.resample("A").mean().head()
AusBeer.resample("A").mean().describe()
```

```
AusBeer.resample("A").mean().plot()
plt.ylabel("Megalitres Beer")
plt.xlabel("Year")
```

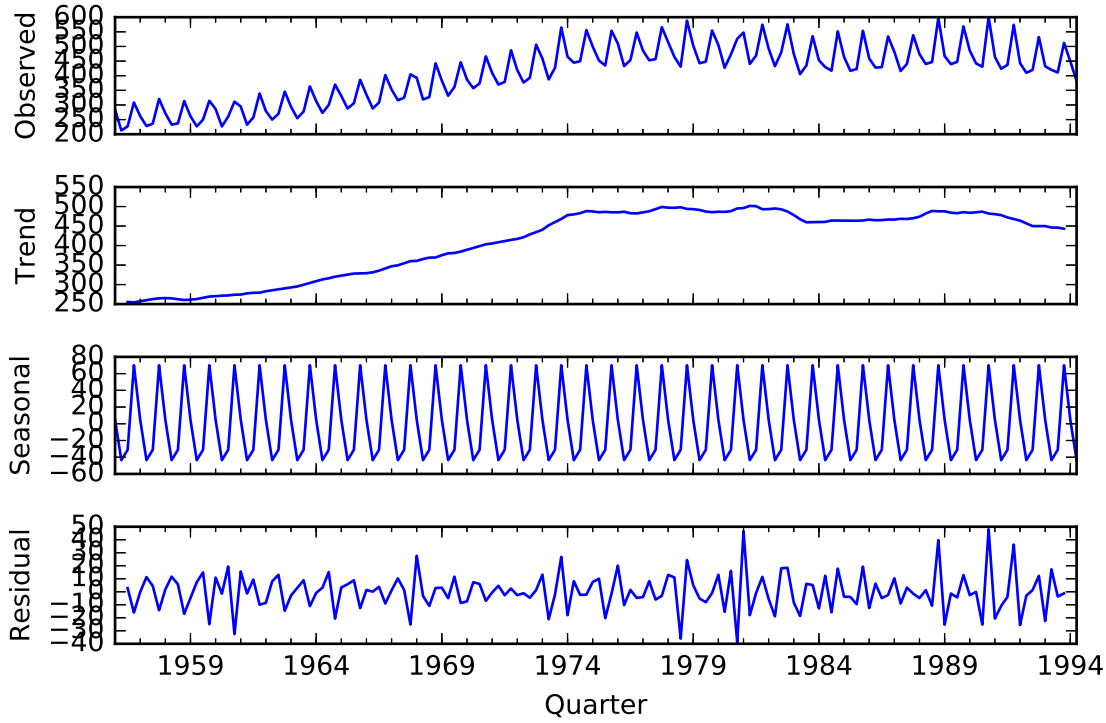



```
AusBeer["quarter"] = AusBeer.index.quarter
AusBeer.boxplot(by = "quarter")
plt.ylabel("Megalitres Beer")
```



c) Die Graphik der Zeitreihe aus Teilaufgabe (a) legt nahe, dass die saisonalen Effekte ziemlich stabil über den Zeitraum verteilt sind, d.h. die Varianz über die Zeit kann als konstant betrachtet werden. Für die Zerlegung ist somit keine Transformation nötig. Der Restteil der Zeitreihe bestätigt, dass die Zeitreihe durchaus ohne weitere Transformation zerlegt werden kann : es ist kaum ein Muster darin erkennbar und die Varianz scheint stabil zu sein. (zu **R**)

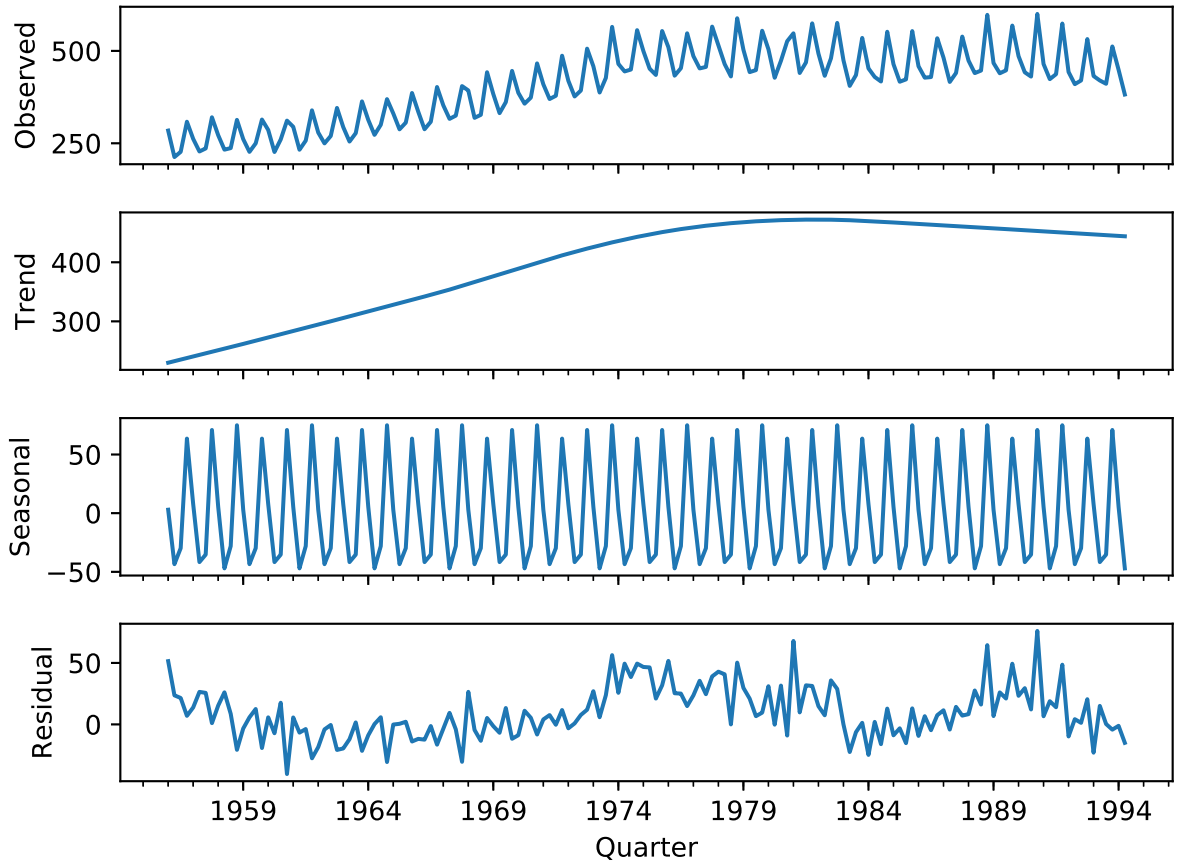
```
from statsmodels.tsa.seasonal import seasonal_decompose
seasonal_decompose(AusBeer["Megalitres"], model="additive", freq=4).plot()
```



Die Zeitreihe des Restterms bestätigt, dass die Zeitreihe ohne weitere Transformation zerlegt werden kann. Es ist kaum ein periodisches Muster in der Zeitreihe des Restterms erkennbar, und die Varianz scheint stabil konstant zu sein.

d) (zu **R**)

```
from stldecompose import decompose
AusBeer_stl = decompose(AusBeer["Megalitres"], period=12)
AusBeer_stl.plot();
```

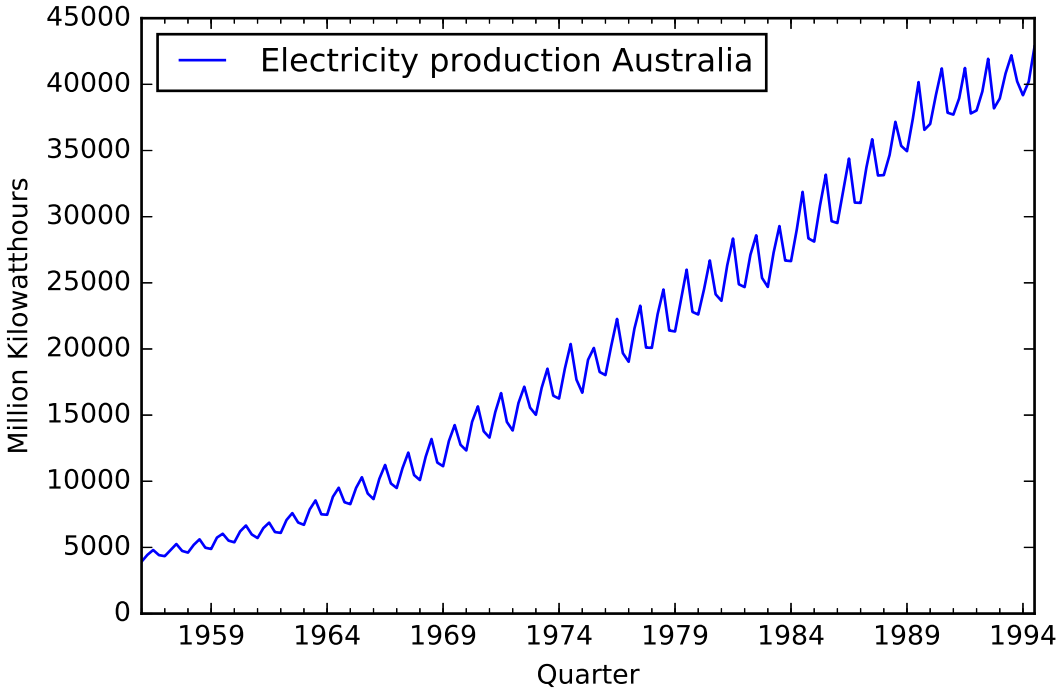


Wir erkennen nun, dass die saisonale Kurve nicht mehr konstant ist. Es ist auch im Falle der STL-Zerlegung kein periodisches Muster in der Zeitreihe des Restterms erkennbar.

Lösung 11.3

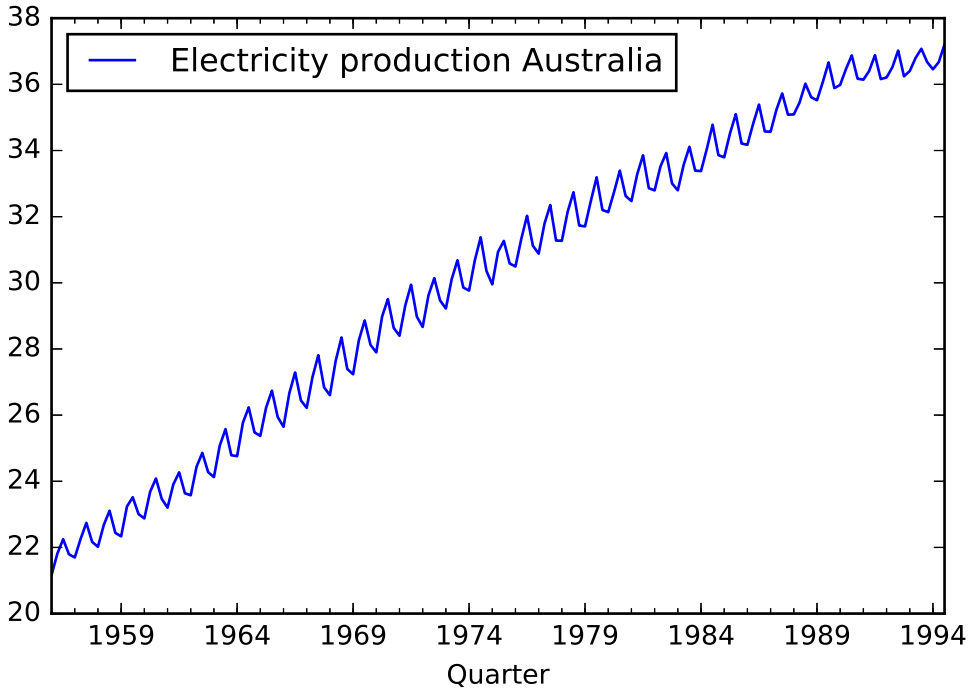
a) (zu R)

```
Electricity = pd.read_csv(".../AustralianElectricity.csv", sep=";", header=
Electricity.head()
Electricity["Quarter"] = pd.DatetimeIndex(Electricity["Quarter"])
Electricity.set_index("Quarter", inplace=True)
Electricity.columns=["Electricity production Australia"]
Electricity.head()
Electricity.plot()
plt.ylabel("Million Kilowatthours")
```



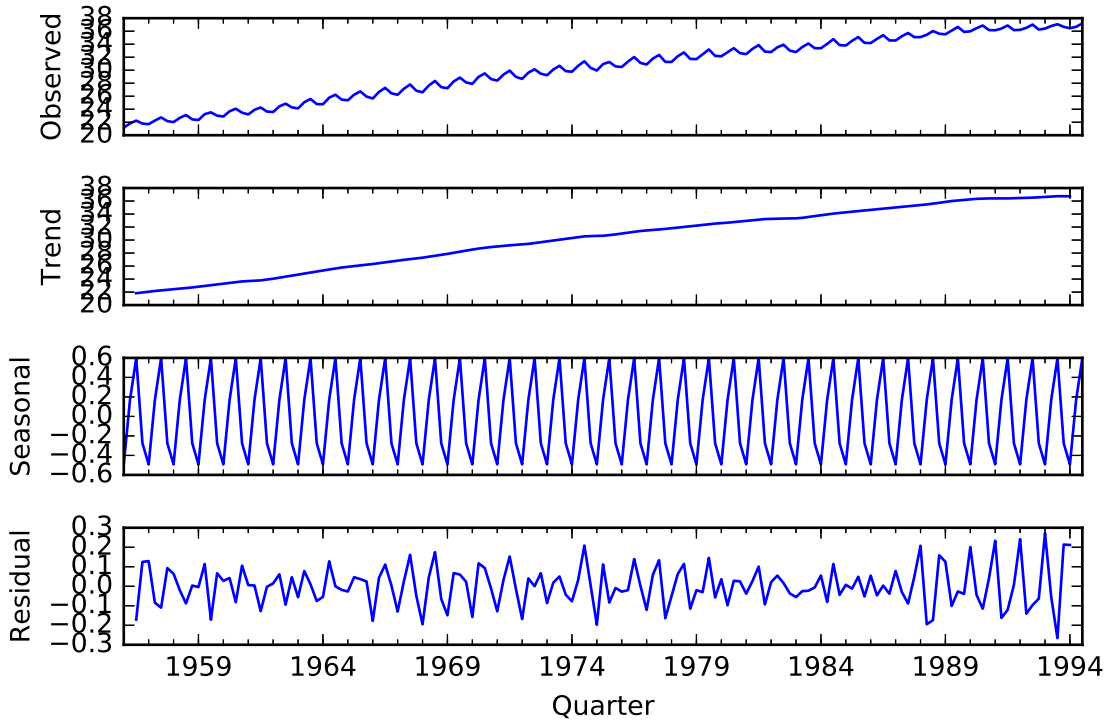
b) Es ist offensichtlich, dass die Varianz am Anfang der Aufzeichnung kleiner ist als am Ende. Ein Wert von 0.2 für den Box-Cox Parameter funktioniert gut und stabilisiert die Varianz. (zu **R**)

```
def boxcox(x,lambd):
  return np.log(x) if (lambd==0) else (x**lambd-1)/lambd

Electricity_tr = boxcox(Electricity, 0.2)
Electricity_tr.plot()
plt.ylabel("Transformed Electricity")
```



c) (zu **R**)

```
from statsmodels.tsa.seasonal import seasonal_decompose
seasonal_decompose(Electricity_tr, model="additive", freq=4).plot()
```



Die Restreihe bestätigt, dass die Zeitreihe nach einer Datentransformation gut zerlegt werden kann. Es ist kaum noch ein Muster bei der Reihe des Restterms ersichtlich. Einzig am Ende der Zeitreihe wird ein bestimmtes saisonales Muster ersichtlich.

d) Experimentiert man mit dem Parameterwert von **`period`**, so ergibt sich, dass ein vernünftiger Parameterwert 4 ist. (zu **R**)

```
from stldecompose import decompose
Electricity_stl = decompose(Electricity_tr, period=4)
Electricity_stl.plot();
```

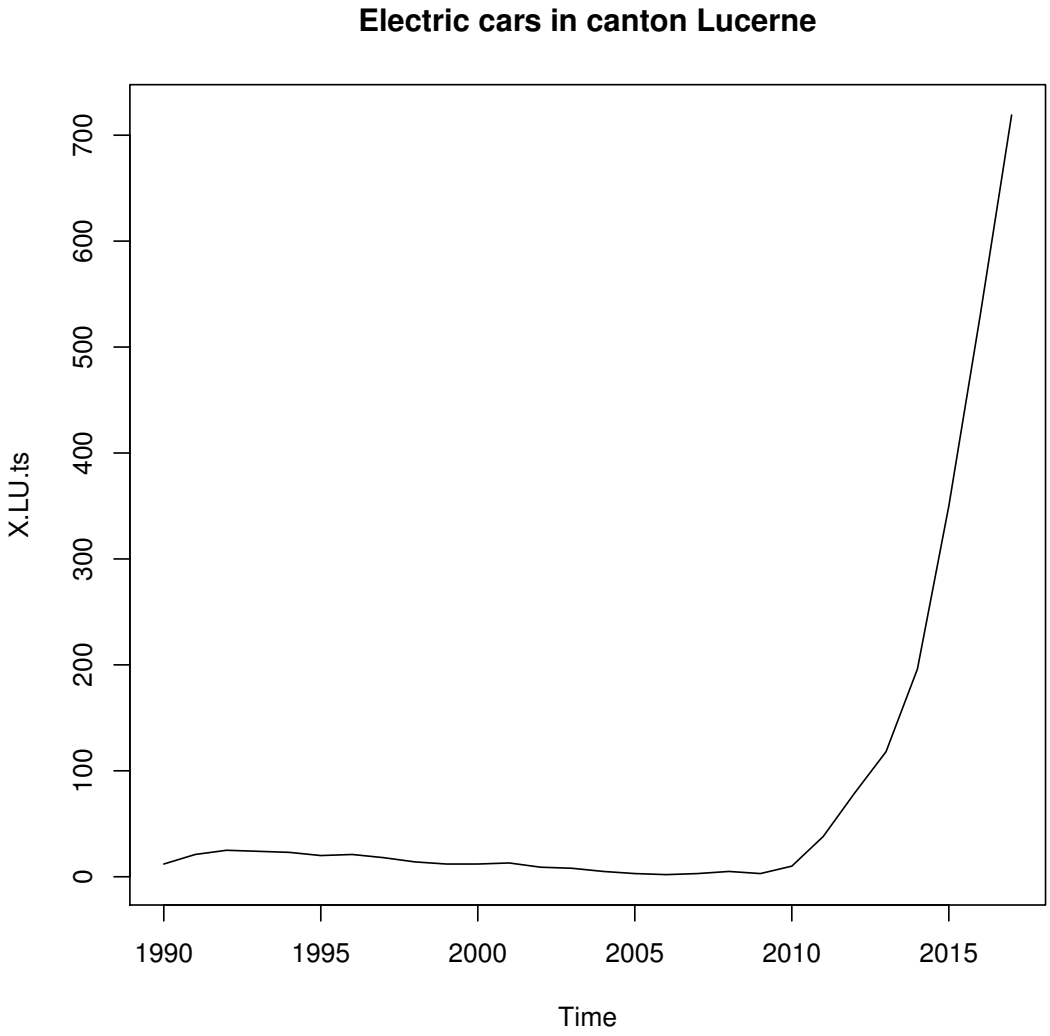
Wir sehen wiederum, dass die Zeitreihe des Restterms in der STL Zerlegung ein vollkommen zufälliges Erscheinungsbild aufweist.

R-Code

Aufgabe 11.1

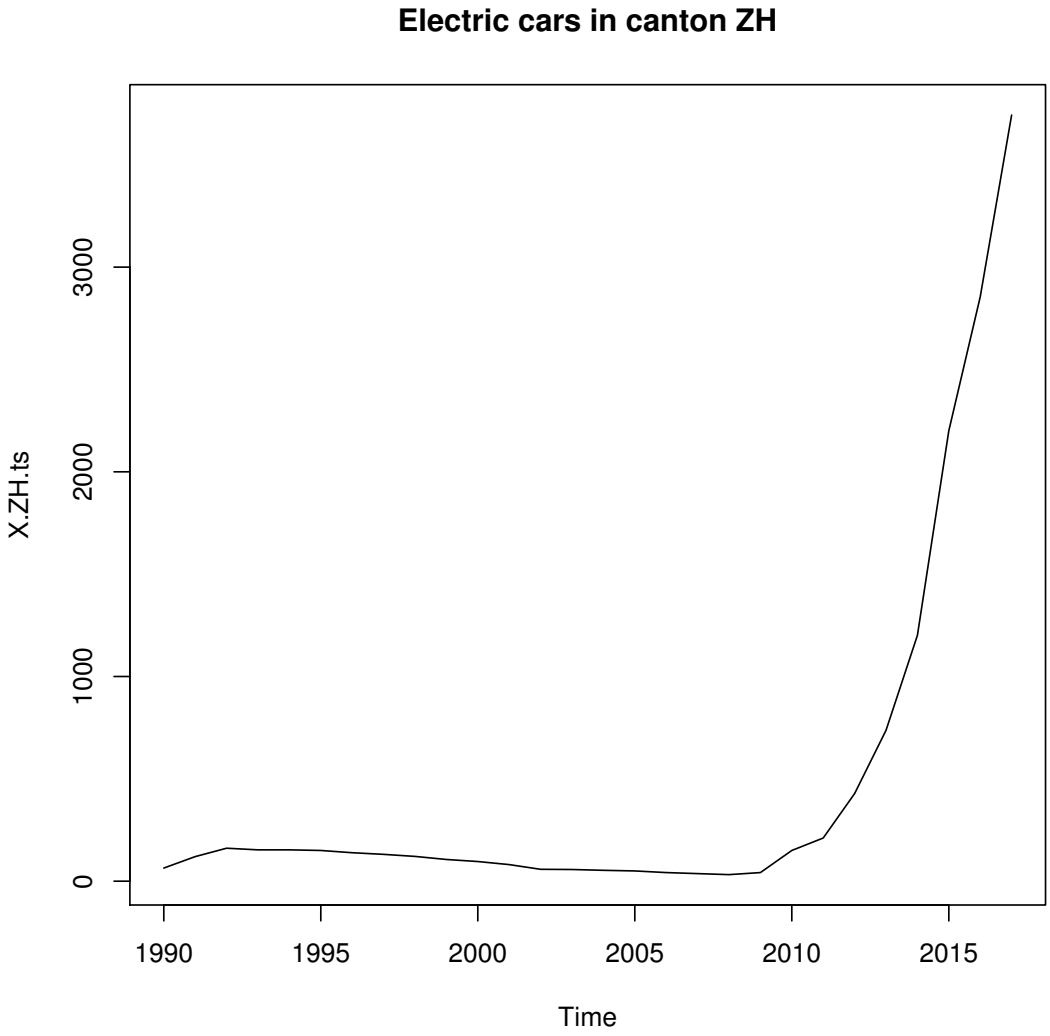
c) (zu **Python**) Die Daten zu elektrischen Autos sind in der Datei `PW_electric.csv` abgespeichert. Wir lesen die Datei mit der Funktion **`read.table()`** ein.

```
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 Lucerne data (first row)
X.LU = as.numeric(X[3, 2:29])

# generate a time series
X.LU.ts = ts(X.LU, start = 1990, end = 2017, frequency = 1)
plot(X.LU.ts, main = "Electric cars in canton Lucerne")
```



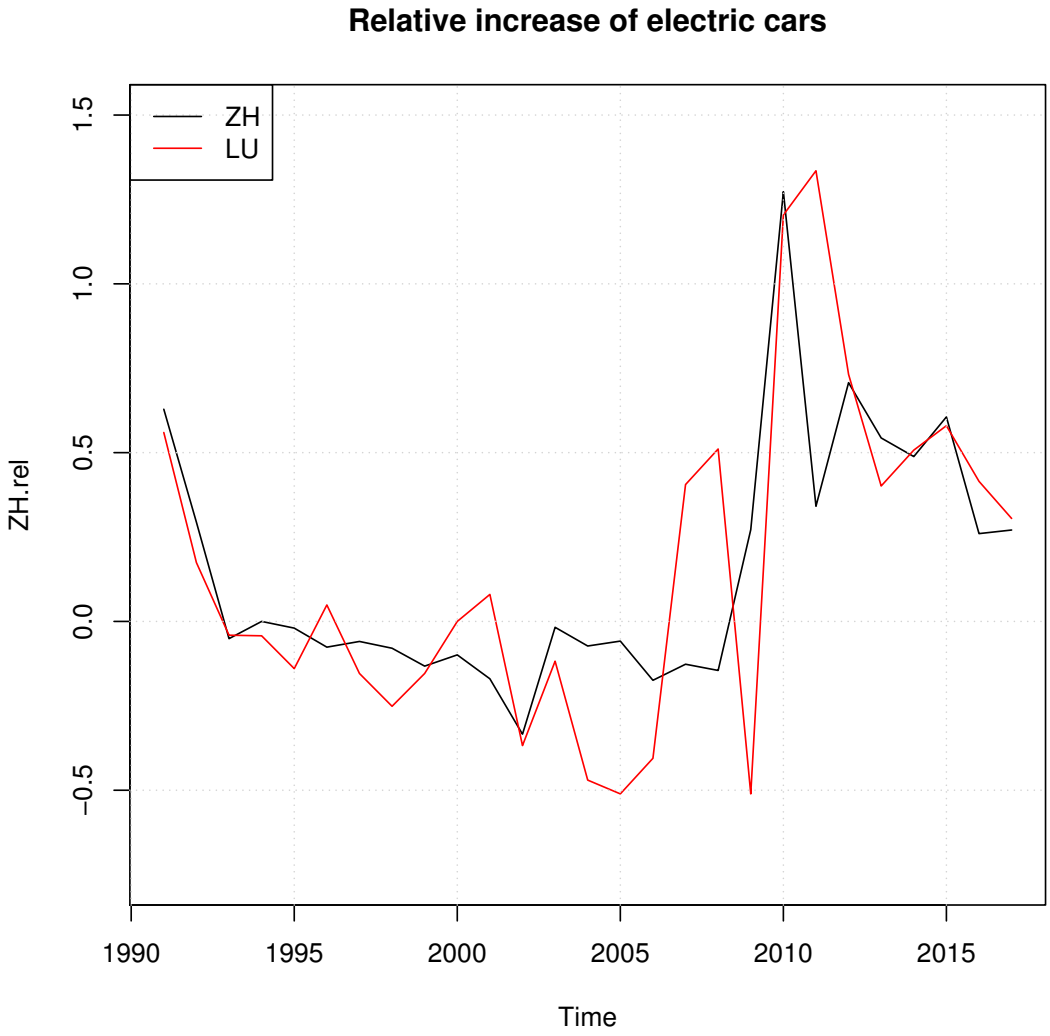
d) (zu **Python**)

```
# extract the Zurich data (first row)
X.ZH = as.numeric(X[1, 2:29])

# generate a time series
X.ZH.ts = ts(X.ZH, start = 1990, end = 2017, frequency = 1)
plot(X.ZH.ts, main = "Electric cars in canton ZH")
```



e) Die zeitlich verschobene Zeitreihe kann in **R** mit Hilfe der Funktion **lag()** bestimmt werden. (zu **Python**)

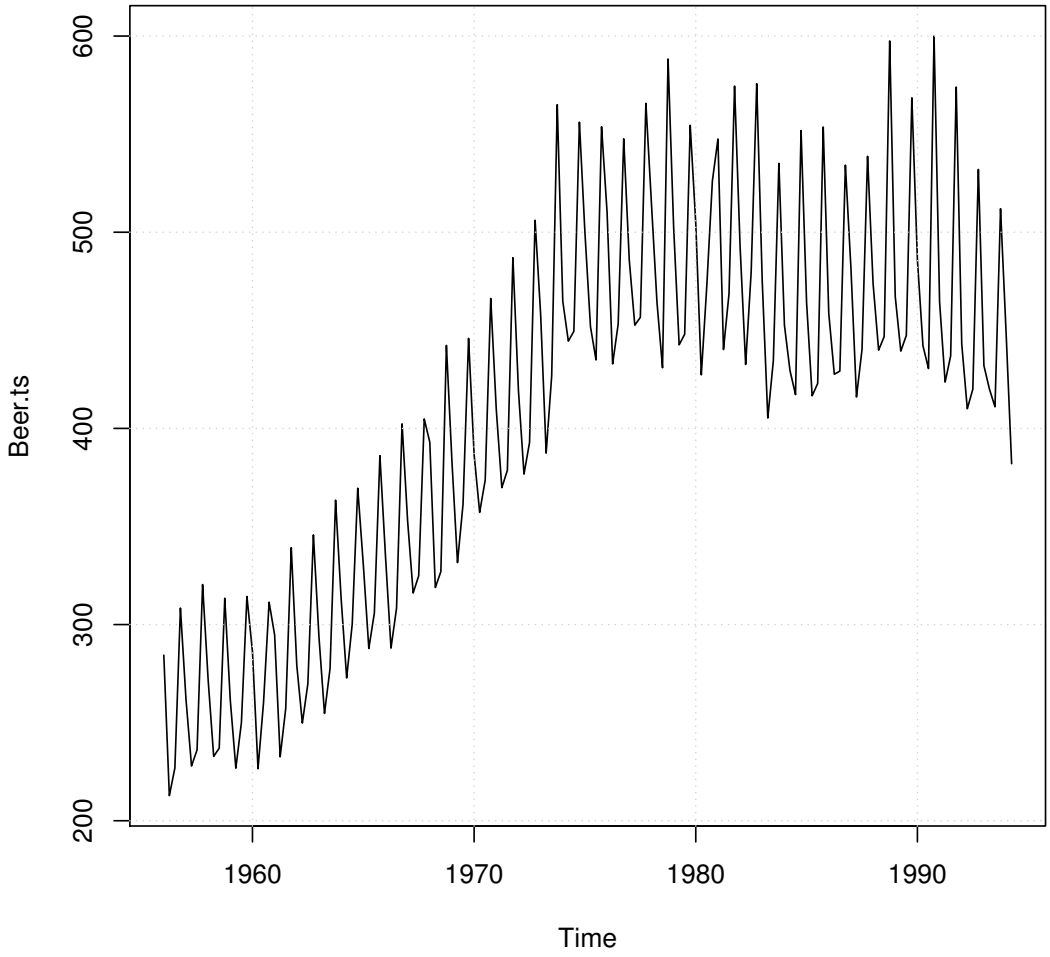
```
ZH.rel = log(X.ZH.ts) - log(lag(X.ZH.ts, -1))
plot(ZH.rel, main = "Relative increase of electric cars",
    ylim = c(-0.75, 1.5))

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



Aufgabe 11.2

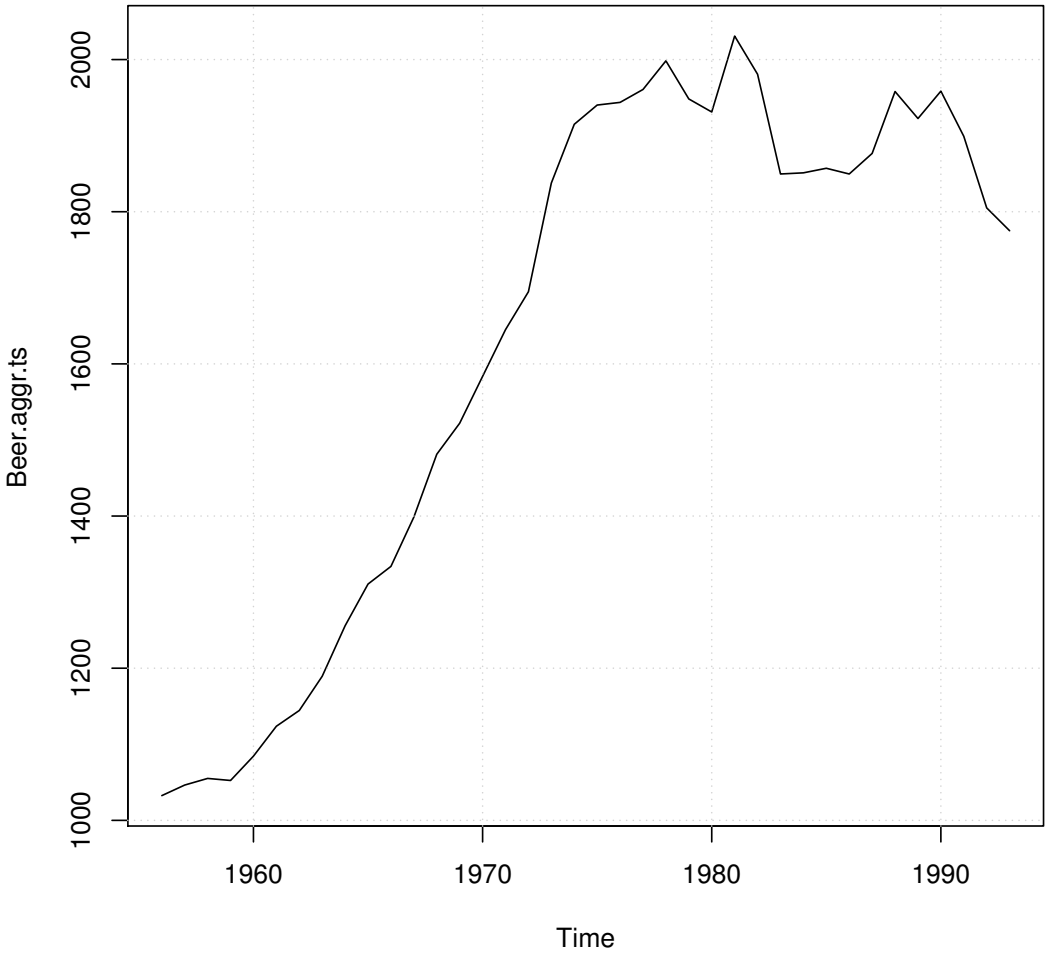
a) (zu **Python**)

```
Beer = read.csv("Daten/AustralianBeer.csv", sep = ";")
Beer.ts = ts(Beer[, 2], start = c(1956, 1), end = c(1994,
    2), frequency = 4)
plot(Beer.ts)
grid()
```

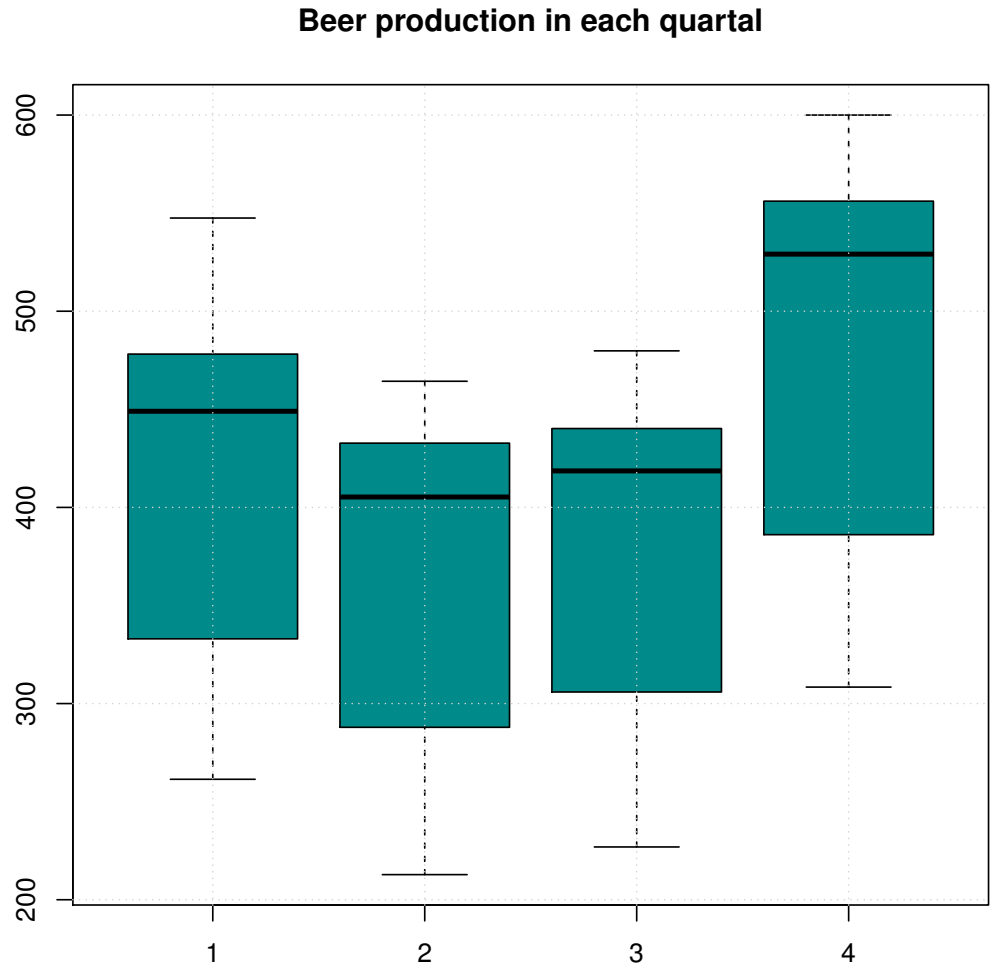


b) (zu **Python**)

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

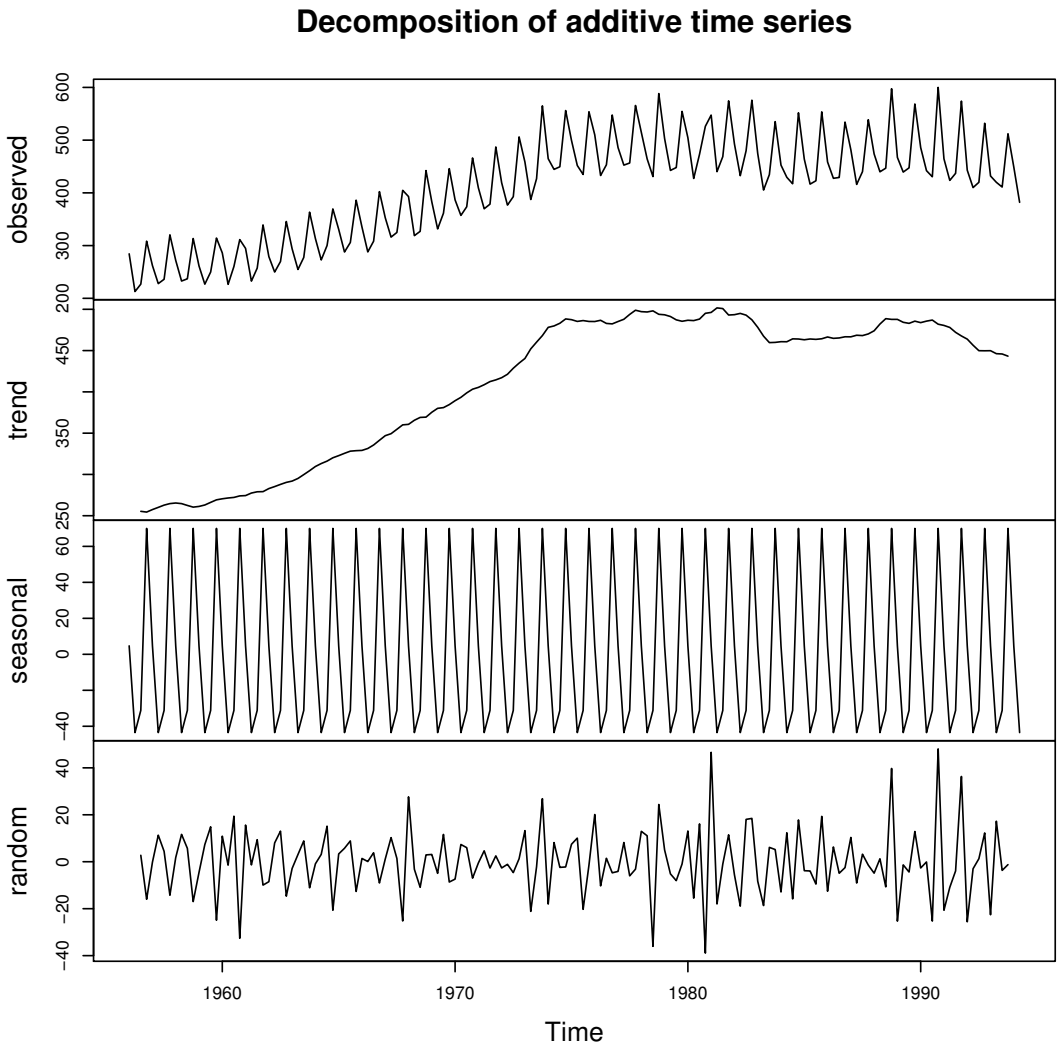



```
# boxplot the data
boxplot(Beer.ts ~ cycle(Beer.ts), col = "darkcyan",
    main = "Beer production in each quartal")
grid()
```



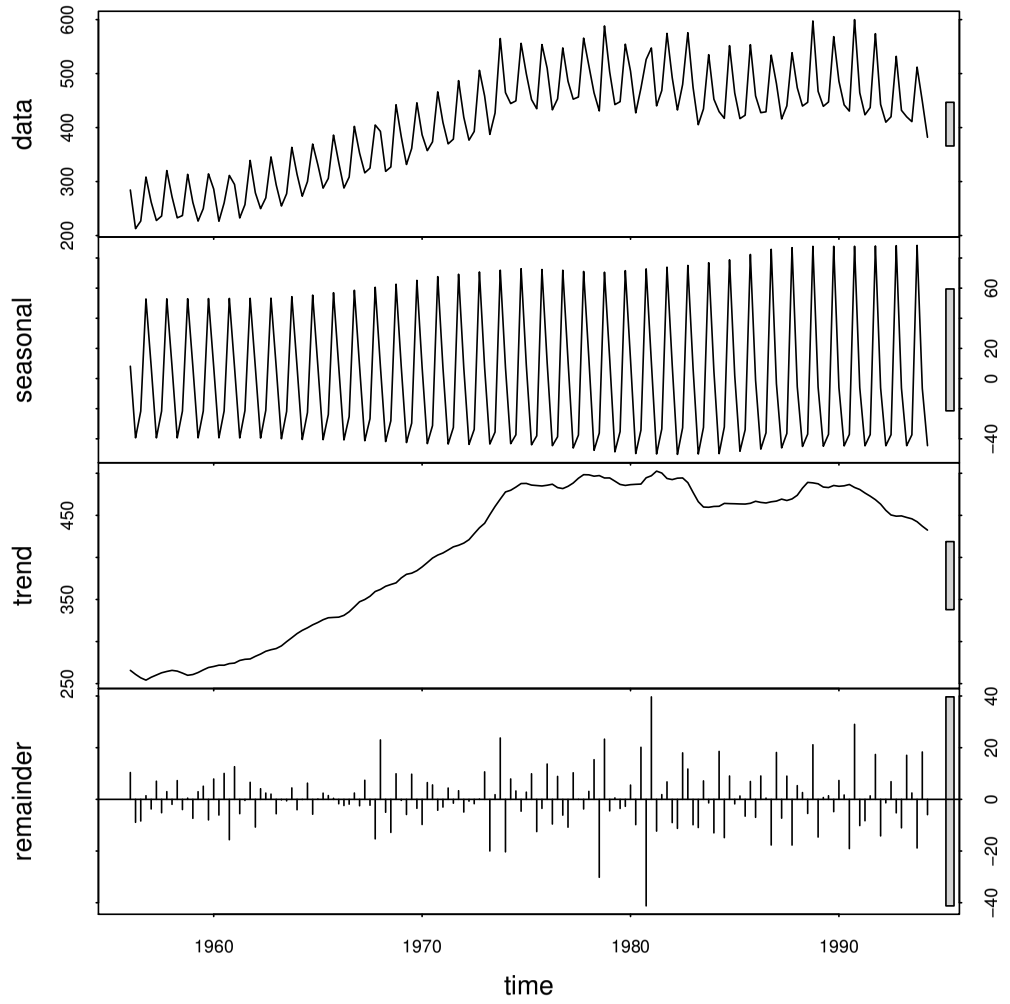
c) (zu **Python**)

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

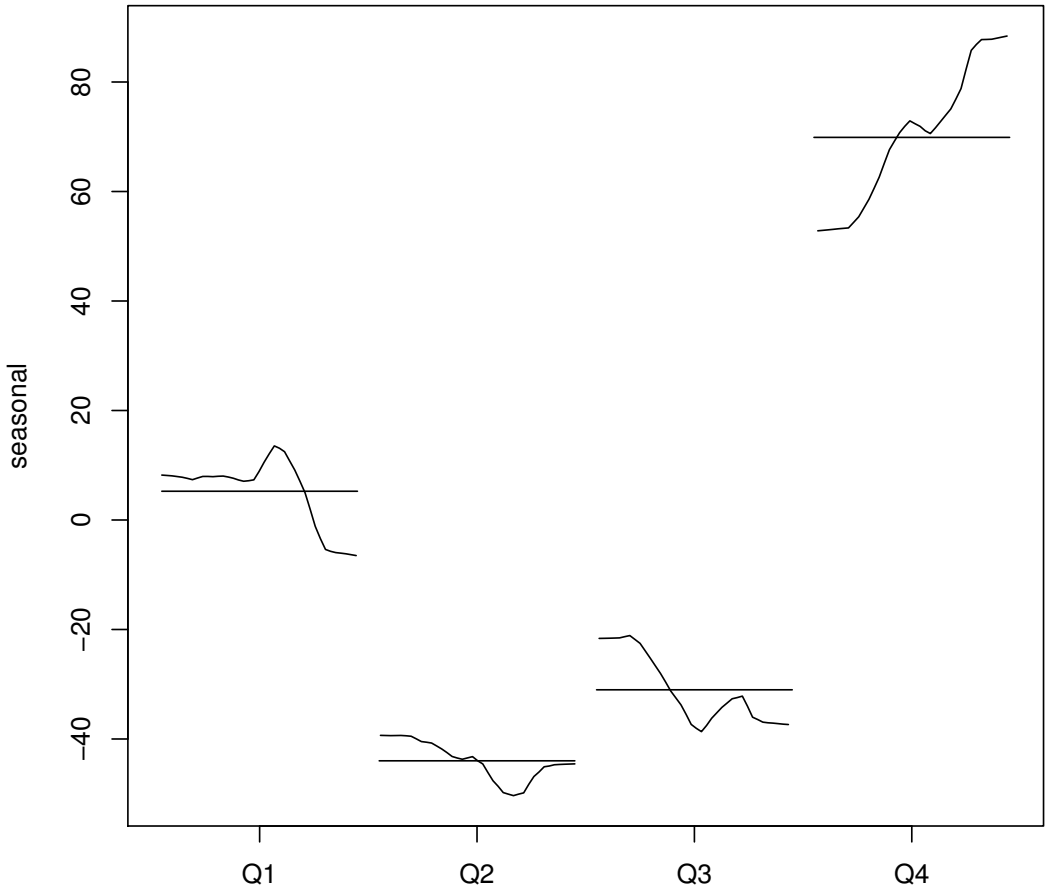


d) (zu **Python**) Experimenting with the **s.window** paramter and doing **monthplots** iteratively shows that a value about 15 could be reasonable

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




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



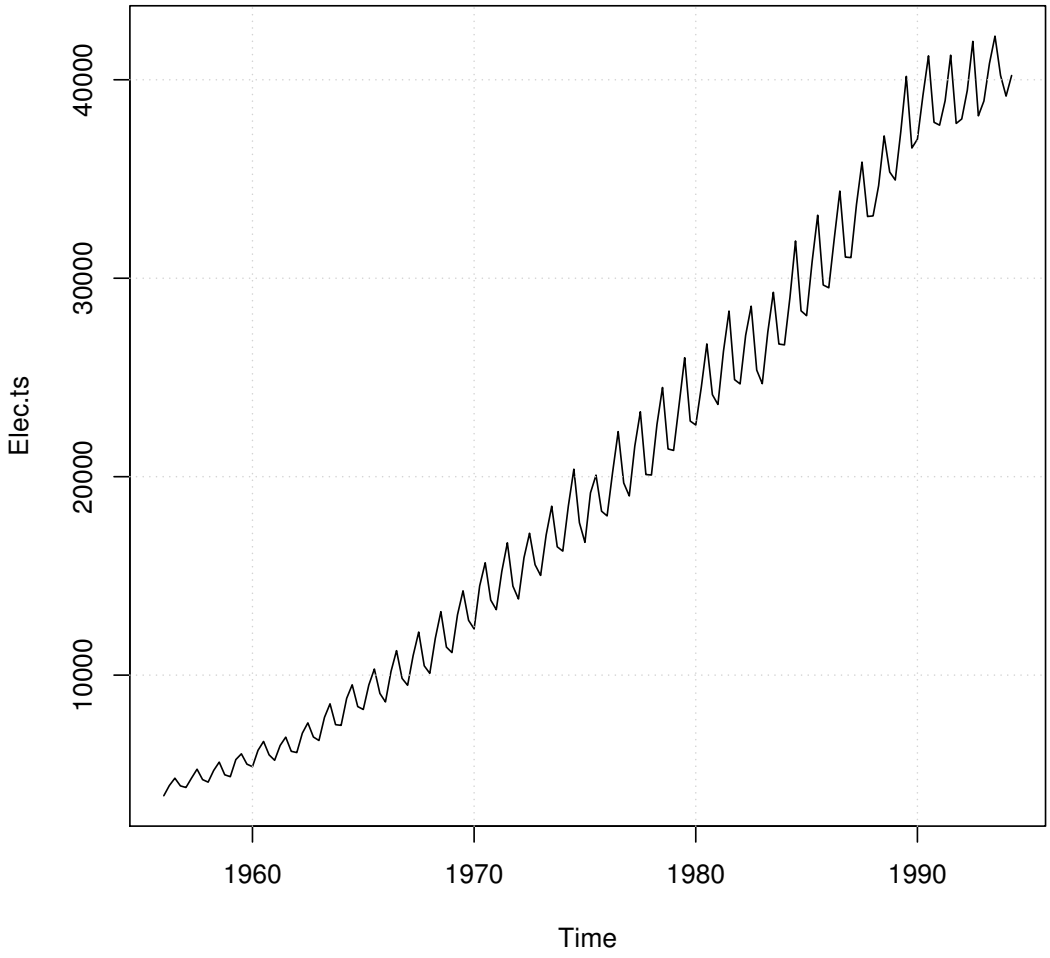
The monthplot indicates that the seasonal effect of beer production has been decreasing for the first quarter and increasing for the last quarter. The remaining quarters show a quite stable seasonal impact.

Aufgabe 11.3

a) (zu **Python**)

```
Elec = read.csv("Daten/AustralianElectricity.csv",
    sep = ";")
Elec.ts = ts(Elec[, 2], start = c(1956, 1), end = c(1994,
    2), frequency = 4)
plot(Elec.ts)
```

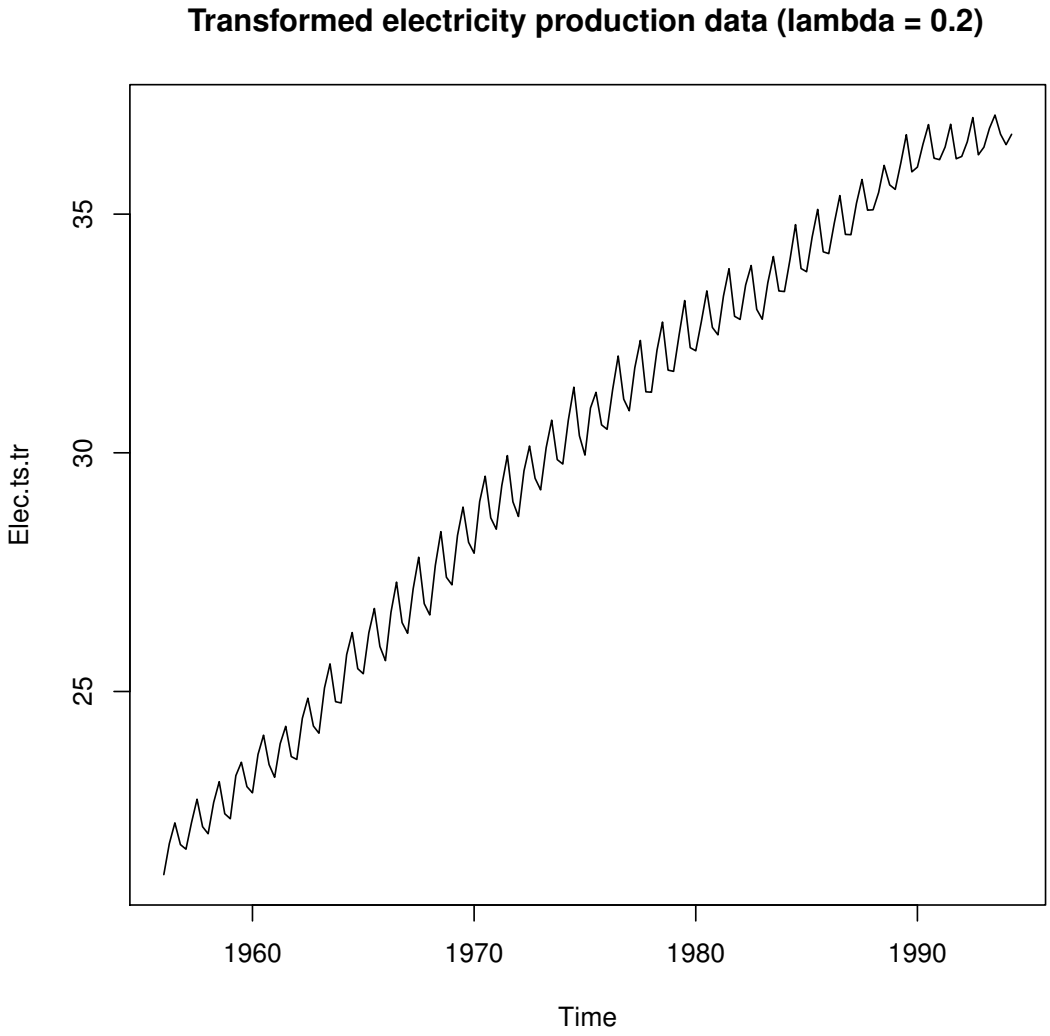
```
grid()
```



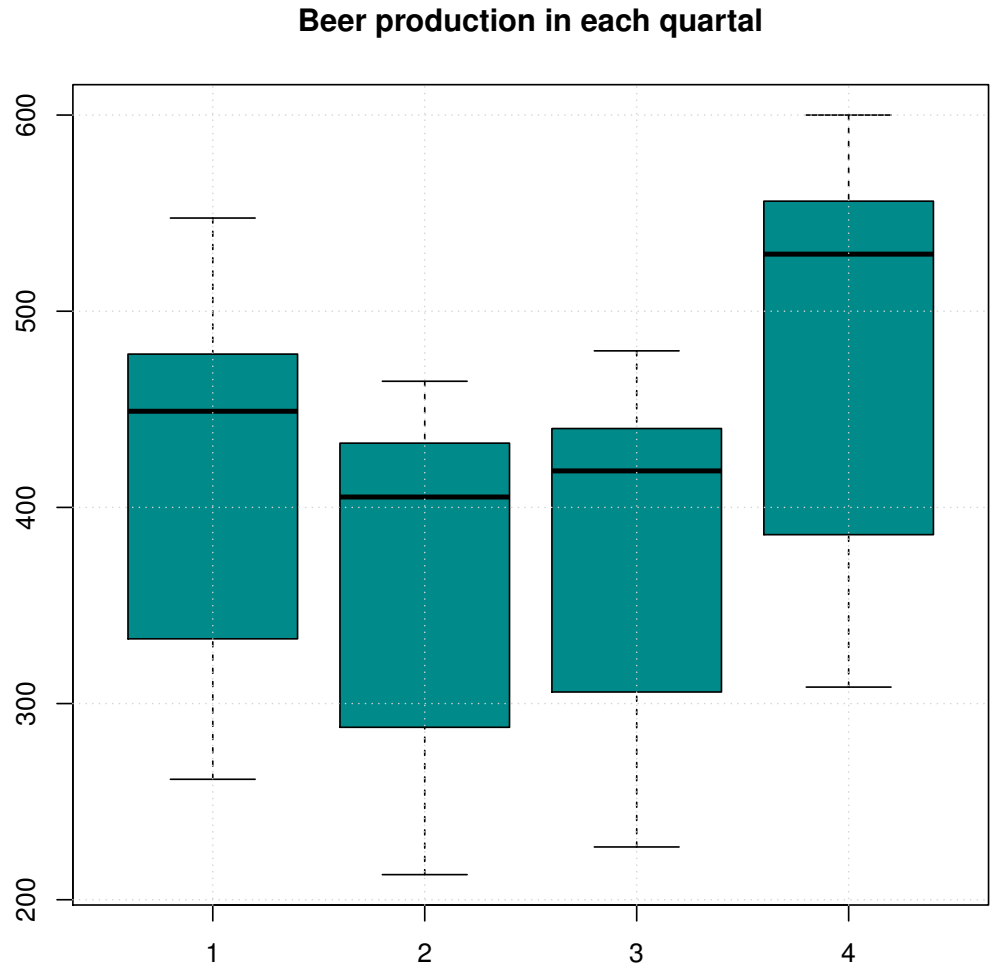
b) Es ist offensichtlich, dass die Varianz am Anfang der Aufzeichnung kleiner ist als am Ende. Ein Wert von 0.2 für den Box-Cox Parameter funktioniert gut und stabilisiert die Varianz.

(zu **Python**)

```
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 =
```

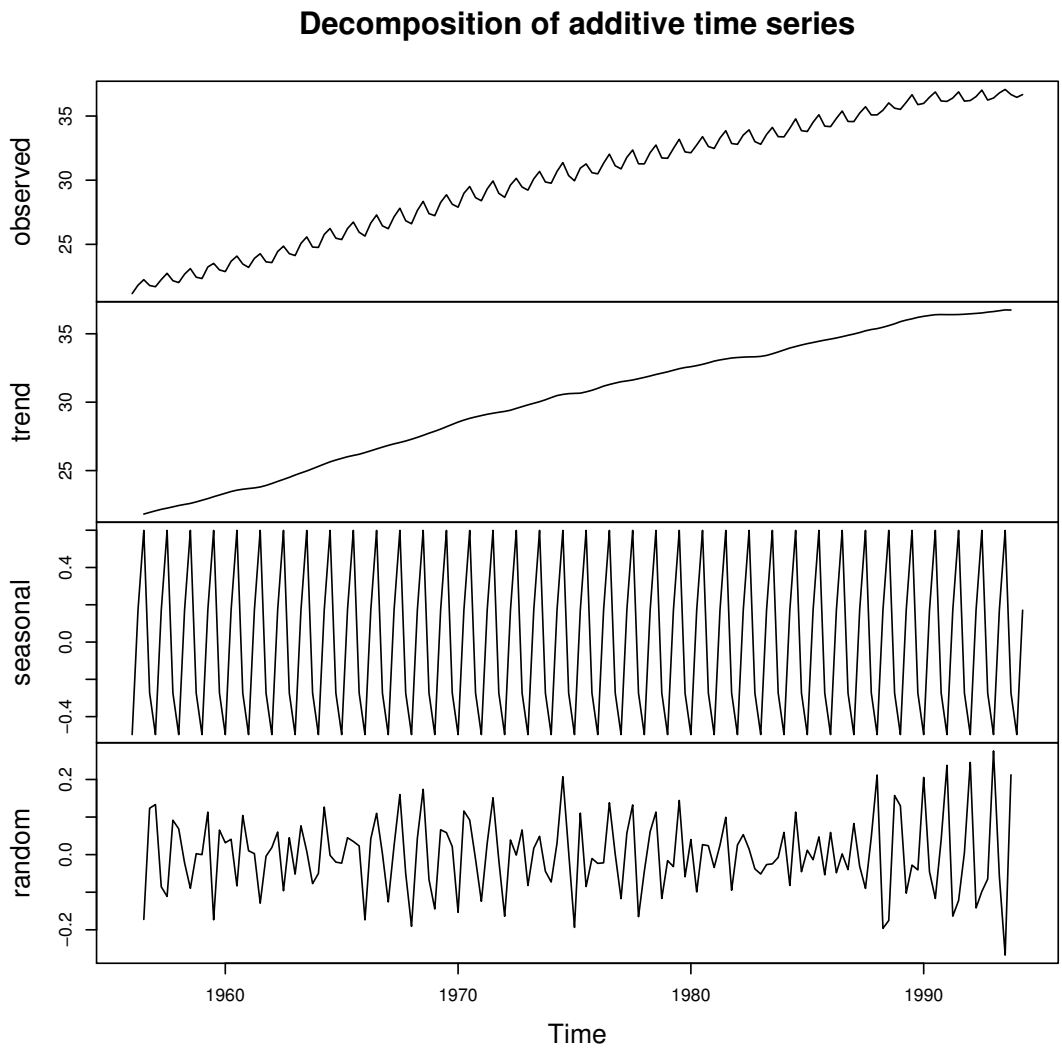



```
# boxplot the data
boxplot(Beer.ts ~ cycle(Beer.ts), col = "darkcyan",
    main = "Beer production in each quartal")
grid()
```



c) Wir zerlegen die Zeitreihe mit Hilfe von der Funktion **decompose()**, nachdem wir die Transformation aus der Teilaufgabe (b) angewandt haben. (zu **Python**)

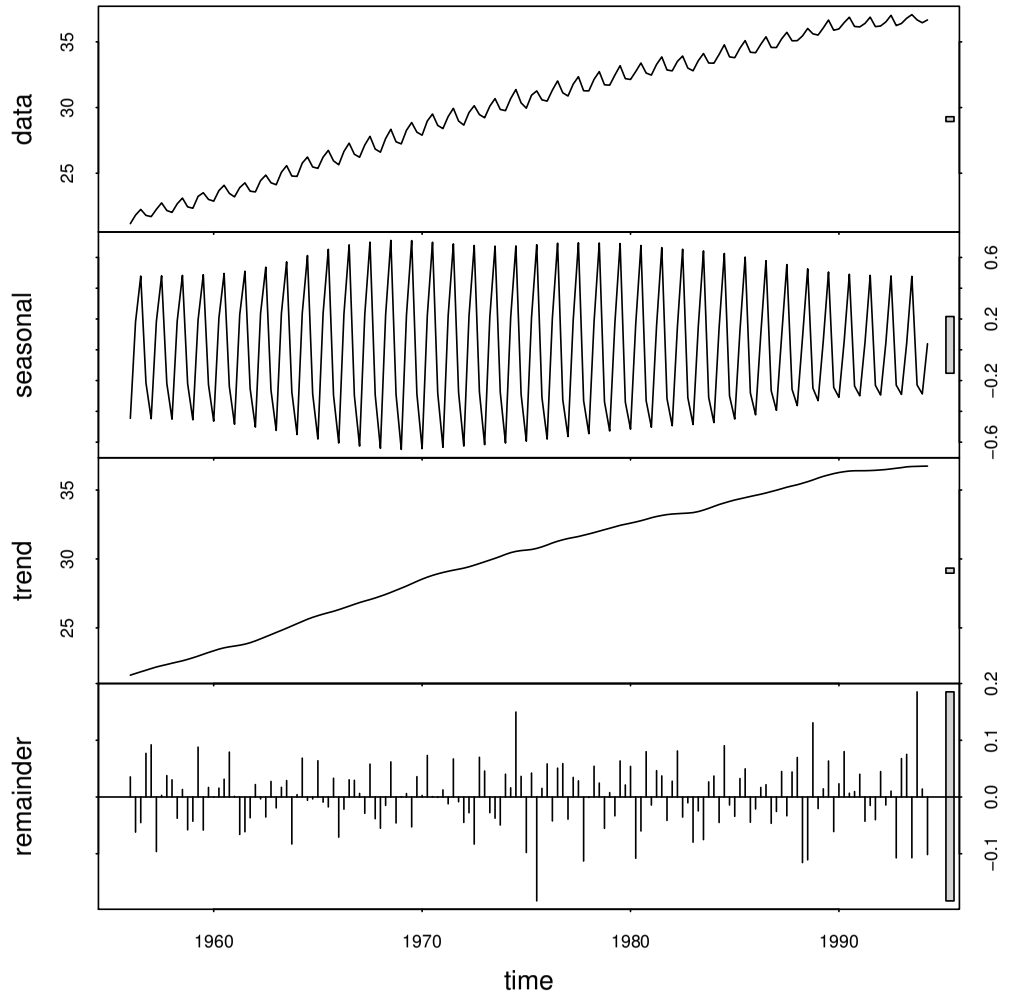
```
res.dec = decompose(Elec.ts.tr)
plot(res.dec)
```



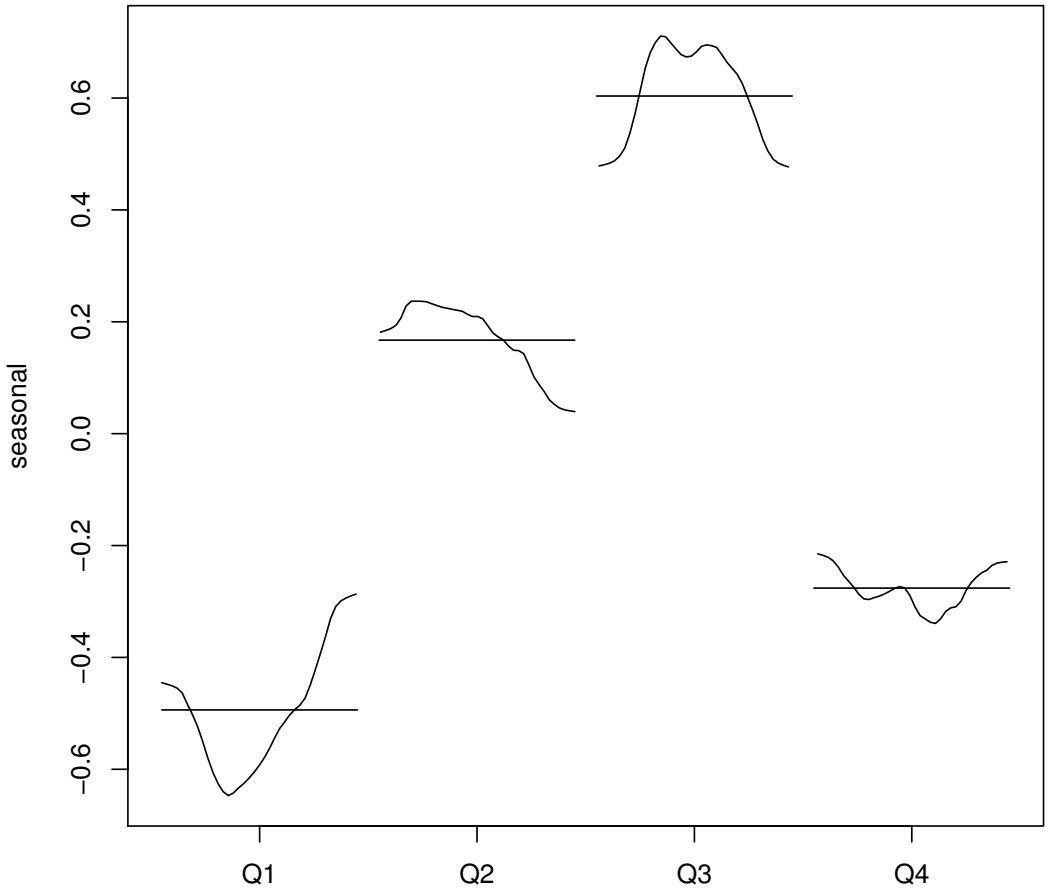
Die resultierende Zeitreihe bestätigt, dass die Zeitreihe vernünftig gut zerlegbar ist nach der Datentransformation. Es ist so gut wie kein Muster mehr erkennbar, und die Varianz scheint stabilisiert zu sein. Dennoch ist am Ende der Zeitreihe noch ein saisonales Muster erkennbar.

d) Das Resultat aus Teilaufgabe c) zeigt, dass saisonale Effekte variieren über die Zeit (selbst wenn wir versuchen, die Daten korrekt zu transformieren). Wenn wir mit dem Parameterwert von **`s.window`** experimentieren und den Befehl **`monthplots`** iterativ ausprobieren, zeigt sich, dass ein Wert von 10 vernünftig sein dürfte. (zu **Python**)

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




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



Wir sehen wiederum, dass die restliche Zeitreihe in der STL Zerlegung ein vollkommen zufälliges Erscheinungsbild aufweist und dass monthplots ziemlich vernünftige Aenderungen im vierteljährlichen Beitrag zur Saisonalität beiträgt.