Beispiel in R: Verfahren zur Modellierung von ZR mit Saison und Trend

Regina Tüchler

November 2, 2009

Beispiel: Zeitreihenanalyse der Übernachtungs-Daten:

Wir haben Daten mit monatlichen Übernachtungszahlen in allen Kategorien in Österreich für den Zeitraum Jänner 1995 bis Mai 2004.

Die Daten `nightts-monthly.rda` werden geladen. Sie müssen sich dafür im working directory befinden.

```
> load("nightts-monthly.rda")
```

Die Daten sind jetzt im Arbeitsspeicher und unter dem Namen `night` verfügbar.

1. Schritt: Zeitreihenplot wird in Fig. 1 gezeichnet.

```
> plot(night)
```

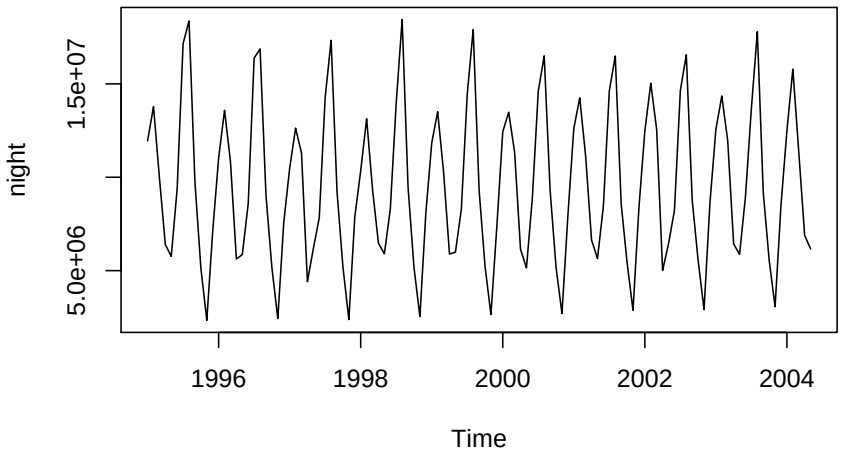

Figure 1: Zeitreihenplot der Übernachtungs-Daten.

2. Schritt: Analyse des Zeitreihenplots, Wahl einer Methode:

Welche Strukturen sind in Fig. 1 zu erkennen? Sollen die Daten transformiert werden? Gibt es Ausreißer? Sind Brüche erkennbar?

Man sieht sehr deutlich die Saisonalität mit 2 Hauptsaisonen bei diesen Monatsdaten. Da die Amplitude gleich bleibt, kann die Saisonalität als additive Saisonalität modelliert werden. Es ist kaum ein Trend erkennbar. Die Daten müssen nicht transformiert werden. Es sind keine Brüche erkennbar.

Ein Verfahren, das die Saisonalität modelliert, muss verwendet werden. Wir werden die Daten der Jahre Jänner 1995-Dezember 2003 zuerst gemäß dem klassischen Komponentenmodell zerlegen, und zwar mit der Methode der kleinen Trends und den R-Funktionen `decompose` und `stl`.

Danach werden wir mit Hilfe des Holt-Winters Verfahrens eine Prognose für die ersten 5 Monate des Jahres 2004 erstellen.

Die Methode der kleinen Trends

Wir führen diese Methode für die Jahre 1995-2003 durch und speichern diese kürzere ZR in `nightts`:

```
> nightts <- window(night, end = c(2003, 12))
```

Die Methode der kleinen Trends ist nicht vorgefertigt in R vorhanden. Die notwendigen Operationen bestehen im Wesentlichen aber nur im Bilden von Zeilen- und Spaltensummen von Matrizen und können daher leicht durchgeführt werden. Damit R die notwendigen Operationen durchführen kann, ordnen wir die Daten bis 2003 (das sind 108 Datenpunkte) so in einer R-Matrix an, dass in den Zeilen die Jahre 1995 - 2003 stehen und in den Spalten die Monate Jän. - Dez. Dazu wird der Befehl `matrix` verwendet. Die Argumente sind die ZR: `nightts`, die Anzahl der Spalten: `ncol = 12`, durch das Argument `byrow=TRUE` wird angegeben, dass die Elemente von `nightts` zeilenweise in die Matrix eingetragen werden sollen. Die Daten werden mit Namen `matrixnight` bezeichnet:

```
> matrixnight <- matrix(nightts, ncol = 12, byrow = TRUE)
```

Zuerst wird die Trendkomponente als Jahresdurchschnitt der Daten geschätzt. Die Berechnung erfolgt hier mit dem Befehl `rowSums`. Das ist die Zeilensumme in einer Matrix und entspricht bei unseren Daten daher der Summe über alle Monate eines Jahres. Die Trendkomponenten $\hat{m}_j$ werden für die 9 Jahre durch folgende Werte geschätzt:

```
> night.trend <- rowSums(matrixnight)/12

[1] 9759543 9411360 9092168 9262266 9394420 9473874 9592557 9733704 9830582
```

Der Trend wird für alle Monate in einem Jahr konstant angenommen und mit Hilfe der Funktion `ts` als R-ZR gespeichert. Die Argumente von `ts` sind: Die Datenpunkte: `rep(night.trend, rep(12, 9))` - hier werden die 9 verschiedenen Trendwerte über alle 12 Monate des jeweiligen Jahres konstant angenommen und in einen Vektor der Länge 108 zusammengesetzt; der Startpunkt ist Jänner 1995: start = c(1995,1); die Anzahl an Saisonkomponenten pro Jahr: `freq = 12`:

```
> night.trend <- ts(rep(night.trend, rep(12, 9)), start = c(1995,
+     1), freq = 12)
```

Die Daten werden nun um den Trend bereingt und in `notrend` gespeichert:

```
> notrend <- matrix(nightts - night.trend, ncol = 12, byrow = TRUE)
```

Nun können die Saisonkomponenten aus den um den Trend bereinigten Daten berechnet werden. Für jeden Monat werden sie als Durchschnitt über alle 9 Jahre berechnet. Die Berechnung erfolgt hier mit dem Befehl `colSums`. Das ist die Spaltensumme in einer Matrix und entspricht bei unseren Daten daher der Summe über alle 9 Jahre. Für die Saisonkomponenten der 12 Monate werden daher die folgenden Werte geschätzt:

```
> night.season <- colSums(notrend)/9
 [1]  2239591.0  4244785.0  1460619.7 -3616793.1 -3639254.3  -953776.4
 [7]  5363580.0  7856536.5  -323466.6 -4210859.5 -6850915.5 -1570046.9
```

Man kann jetzt z.B. ausgeben, in welchem Monat die Saisonkomponente am größten/kleinsten ist. Die Funktion `which` liefert den Index der entsprechenden Monate:

```
> which(night.season == max(night.season))
[1] 8
> which(night.season == min(night.season))
[1] 11
```

Im August erreicht die ZR ihren größten und im November ihren kleinsten Jahreswert.

Die Saisonkomponenten nehmen jeweils denselben Wert in allen Jahren an und werden als ZR gespeichert:

```
> night.season <- ts(rep(night.season, 9), start = c(1995, 1),
+     freq = 12)
```

Wir zeichnen die ZR, ihre Trendkomponente und ihre Schätzung in Fig. 2

```
> plot(nightts, type = "l")
> lines(night.trend, col = "blue")
> lines(night.trend + night.season, col = "red")
```

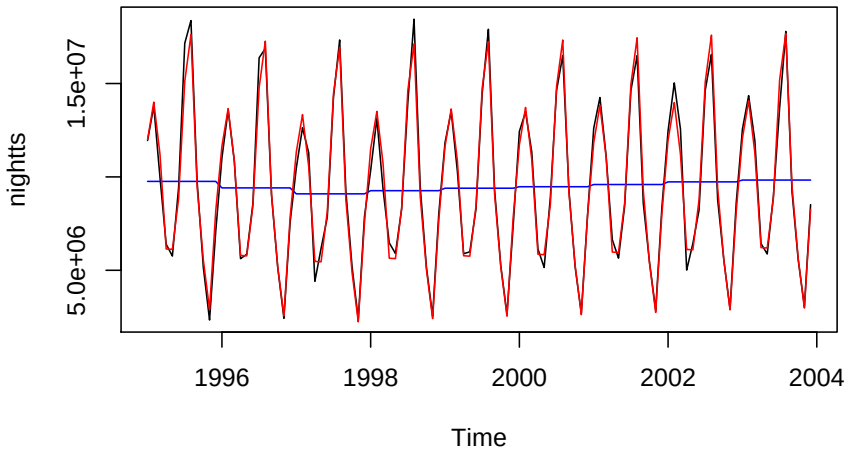

Figure 2: Die Methode der kleinen Trends für die Nächtigungsdaten.

Klassische Zerlegung mittels `decompose`

Die Funktion `decompose` führt in `R` eine klassische Saison-Trend Zerlegung mittels gleitendem Durchschnitt durch. Wir zerlegen die Nächtigungsdaten der Jahre 1995 bis 2003:

```
> nightts.dec <- decompose(nightts)
```

In `nightts.dec` sind jetzt die Saison- und die Trendkomponenten für die einzelnen Zeitpunkte unter `nightts.dec$seasonal` und `nightts.dec$trend` enthalten. In `nightts.dec$random` stehen die Residuen für die einzelnen Zeitpunkte. Eine grafische Darstellung von allen Komponenten liefert der Plot-Befehl in Fig. 3.

```
> plot(nightts.dec)
```

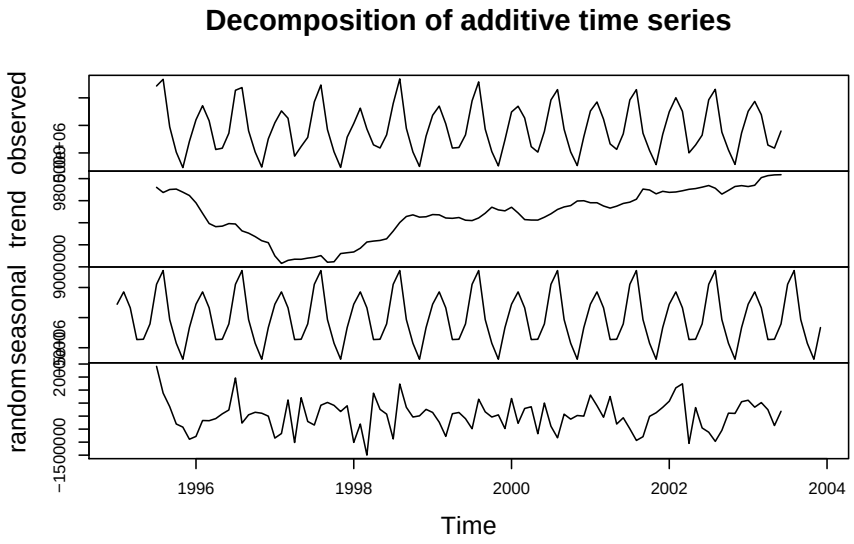

Figure 3: Saison-Trend Zerl. d. Nächtigungs-Daten mittels `decompose`.

Dort sieht man der Zerlegungsformel $x_t = \hat{m}_t + \hat{s}_t + \varepsilon_t$ entsprechend zeilenweise die ZR x_t, $\hat{m}_t$, $\hat{s}_t$ und ε_t.

Klassische Zerlegung mittels `stl`

Eine weitere Funktion zur Durchführung von klassischen Saison-Trend Zerlegungen ist `stl`. Sie bietet viel mehr Möglichkeiten als `decompose` und enthält eine Reihe von zusätzlichen Optionen, die wir in dieser LV aber nicht verwenden werden. Für die klassische Saison-Trend Zerlegung muss im Argument die Option `s.window = "periodic"` gewählt werden.

Für die Nächtigungsdaten der Jahre 1995-2003 ergibt sich daher:

```
> nightts.stl <- stl(nightts, s.window = "periodic")
```

In `nightts.stl` sind nun spaltenweise die Saison-, Trend und Restkomponente für die einzelnen Zeitpunkte enthalten. Auch hier gibt es die Möglichkeit mit `plot` die einzelnen Komponenten zu zeichnen.

Sowohl bei `decompose` als auch bei `stl` muss man bei der Interpretation automatisch generierten Plots darauf achten, dass die in den verschiedenen Subplots verwendeten Maßstäbe nicht gleich sind! So ist in unserem Beipiel, wie aus dem ZR-Plot von x_t auch deutlich wird, die

```
> plot(nightts.stl)
```

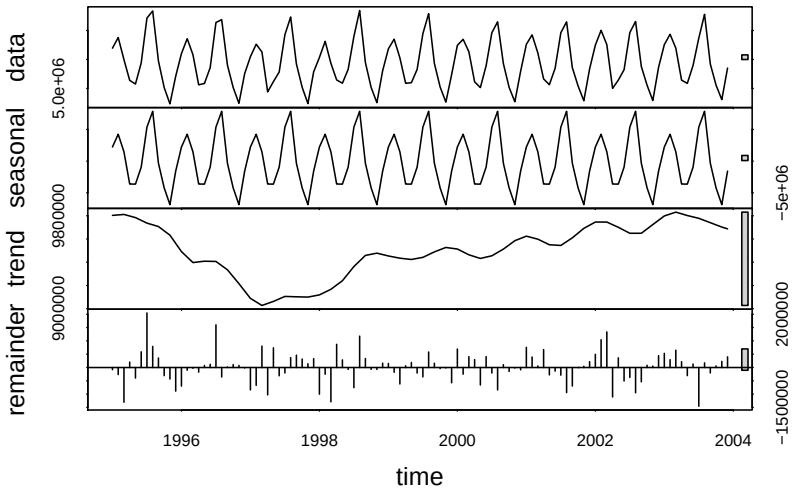

Figure 4: Saison-Trend Zerl. d. Nächtigungs-Daten mittels `stl`.

Trendänderung viel geringer als die Saisonschwankungen.

Holt-Winters Verfahren

Aufgabenstellung: Die Nächtigungsdaten der Jahre 1995-2003 sollen mit dem Holt-Winters Verfahren geglättet werden und Jän.-Mai 2004 sollen prognostiziert werden. Dabei soll der Glättungsparameter zur Niveauschätzung den Wert 0.2, der zur Trendschätzung den Wert 0.4 und der zur Saisonalitätsschätzung den Wert 0.4 bekommen.

Da der Zeitreihenplot kaum eine Trendänderung zeigte, soll dieses Modell dann mit einem Modell ohne Trendkomponente verglichen werden.

3. Schritt: Die Daten Jän. 1995 bis Dez. 2003 sollen Input für das Holt-Winters Verfahren sein, um eine Prognose bis Mai 2004 zu erstellen. Die Daten teilen wir in `past` und in `future`:

```
> past <- window(night, end = c(2003, 12))
> future <- window(night, start = c(2004, 1))
```

Die Argumente in der `R`-Funktion `HoltWinters` sind die Daten der Jahre 1995-2003: `past`, die Glättungsparameter: `alpha=0.2`, `beta=0.4`, `gamma=0.4`. Der Output wird hier in `model` gespeichert.

```
> model <- HoltWinters(past, alpha = 0.2, beta = 0.4, gamma = 0.4)
```

Die Werte für den Prognosezeitraum können mit `R` über die Funktion `predict` automatisch berechnet werden. Dabei muss die Anzahl der zu prognostizierenden Zeitpunkte (in unserem Beispiel 5) im Argument `n.ahead = 5` angegeben werden. Die Vorhersage für Jän. bis Mai 2004 wird hier im Objekt `progn` gespeichert:

```
> progn <- predict(model, n.ahead = 5)
```

In Fig. 5 wird mit dem Plot-Befehl die ZR gemeinsam der geglätteten ZR für die Jahre 1995-2003 gezeichnet. Soll auch die Prognose für 2004 eingezeichnet werden, muss im Argument auch die Option `predicted.values = progn` dazugeschrieben werden. Schließlich können mit lines(future) auch die Originaldaten für 2004 dazugezeichnet werden.

```
> plot(model, predicted.values = progn)
> lines(future)
```

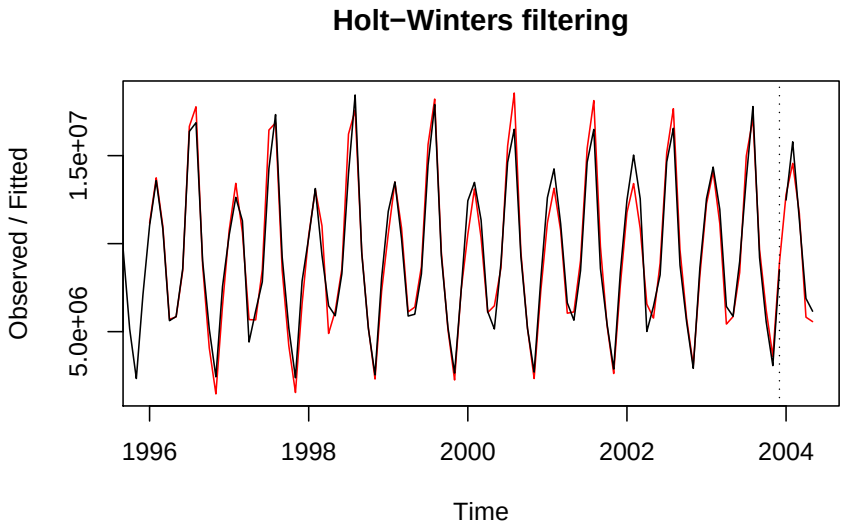

Figure 5: Holt-Winters für die Nächtigungs-Daten.

Wir führen nun die Befehle für die Modellierung ohne Trendkomponente aus. Der Wert des Parameters `beta` muss daher auf 0 gesetzt werden:

```
> model.notrend <- HoltWinters(past, alpha = 0.2, beta = 0, gamma = 0.4)
> progn.notrend <- predict(model.notrend, n.ahead = 5)
```

4. Schritt: Modellwahl:

Es sollen nun das Modell, das Niveau, Trend und Saison enthält mit jenem, das ohne Trendkomponente geschätzt wurde, verglichen werden.

Zunächst zeigt ein optischer Vergleich von Fig. 5 und Fig. 6, dass zwischen den beiden Modellen kaum ein Unterschied besteht. Wir werden daher für einen genaueren Vergleich Maßzahlen verwenden.

Die Fehlerquadratsummen zwischen den Einschrittvorhersagen und den wahren Werten für die Jahre 1995-2003 sind für die Modellwahl geeignet:

```
> model$SSE

[1] 7.209973e+13

> model.notrend$SSE

[1] 5.794333e+13
```

```
> plot(model.notrend, predicted.values = progn.notrend)
> lines(future)
```

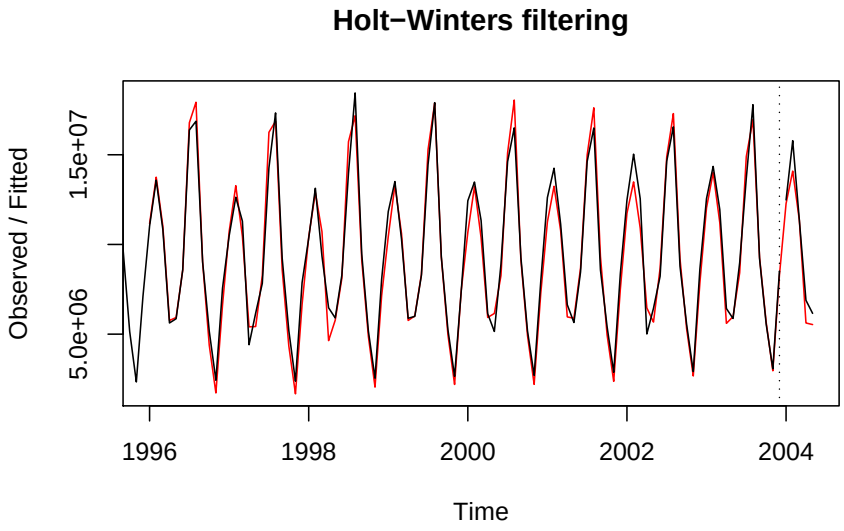

Figure 6: Holt-Winters ohne Trend für die Nächtigungs-Daten.

Die `SSE` ist für das einfachere Modell (ohne Trendkomponente) kleiner. Auf Basis dieser Maßzahl würde man daher dieses Modell wählen.

Wir können die Residuen auch grafisch vergleichen. In Fig. 7 werden die geglätteten Werte, die sich ja in der Spalte `"xhat"` im Objekt `model$fitted` befinden von den wahren Werten `past` abgezogen und gezeichnet (rot für das Modell ohne Trend). Aus dieser Zeichnung könnte man auch nicht die Wahl zwischen den beiden Modellen treffen. Erst die obige `SSE` lässt eine genauere Aussage zu.

Da wir für das Jahr 2004 die wahren Nächtigungszahlen kennen, können wir die Schätzungen mit diesen wahren Werten vergleichen. Es eignet sich hier die mittlere Fehlerquadratsumme über die Monate Jän.-Mai 2004:

```
> true <- matrix(future, 5, 1)
> est <- matrix(progn[, 1], 5, 1)
> est.notrend <- matrix(progn.notrend[, 1], 5, 1)
> mse <- colSums((est - true)^2)/5
[1] 642524322316
> mse.notrend <- colSums((est.notrend - true)^2)/5
[1] 983116477953
```

Auch die mittlere Fehlerquadratsumme über den Prognosezeitraum ist für das Modell ohne Trend geringer. Daher wählt man auch auf Basis dieser Maßzahl das einfachere Modell.

Beispiel: Erdgasverbrauch mit multiplikativem Holt-Winters Verfahren

Die Daten `UKgas` sind im Package `stats` enthalten und sind Quartalsdaten über den UK Gasverbrauch von 1960 - 1986.

```
> plot(past - model$fitted[, "xhat"])
> lines(past - model.notrend$fitted[, "xhat"], col = "red")
```

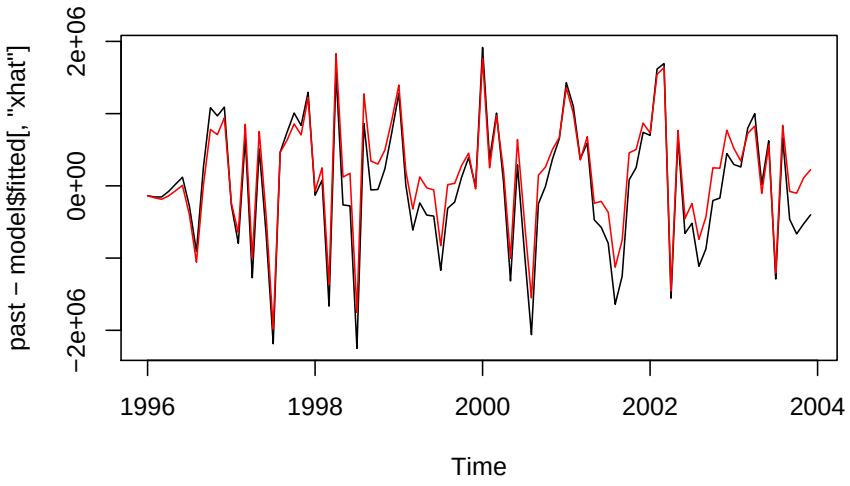

Figure 7: Residuenvergleich von Holt-Winters mit/ohne Trend für die Nächtigungs-Daten.

```
> data(UKgas)
```

Der Zeitreihenplot in Fig. 8 zeigt eine wachsende Amplitude. Daher ist ein Modell zu wählen, das die saisonale Komponente in multiplikativer Form enthält.

Das multiplikative Holt-Winters Verfahren wird hier auf Basis der Daten bis 1980 durchgeführt und eine Prognose für 5 Jahre wird berechnet.

```
> past <- window(UKgas, end = c(1980, 4))
> future <- window(UKgas, start = c(1981, 1))
> model <- HoltWinters(past, seasonal = "multiplicative")
> progn <- predict(model, n.ahead = 20)

> plot(model, progn)
> lines(UKgas)
```

```
> plot(UKgas)
```

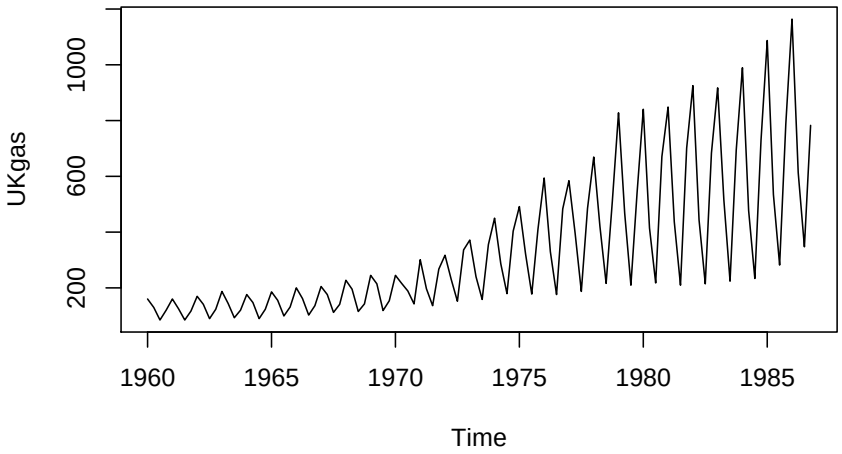

Figure 8: Die UKgas-Daten.

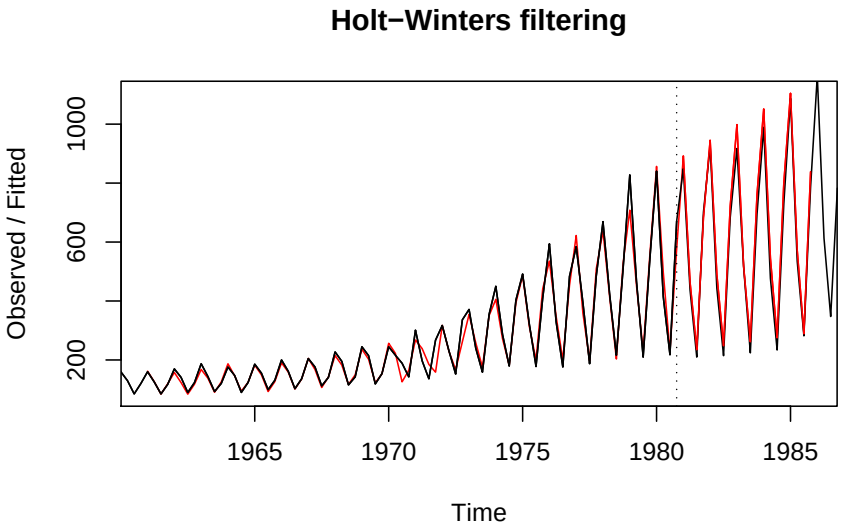

Figure 9: Das Holt-Winters Verf. für die UKgas-Daten.