
Time Series

Computer Lab A

Andreas Charitos (andch552), Ruben Muñoz (rubmu773)

2019-09-16

Contents

1	Computations with simulate data	2
1.1	A)	2
1.2	B)	2
1.3	C)	2
2	Visualization, detrending and residuals analysis of Rhine data	3
2.1	A)	3
2.2	B)	4
2.3	C)	4
2.4	D)	5
2.5	E)	5
3	Analisis of oil and gas time series	6
3.1	A)	6
3.2	B)	6
3.3	C)	7
3.4	D)	7
3.5	E)	8
4	Appendix	9
4.1	Code	9
4.1.1	Code used for Computations with simulate data	9
4.1.2	Code used for Visualization, detrending and residuals analysis of Rhine data	10
4.1.3	Code used for Analisis of oil and gas time series	11

1 Computations with simulate data

1.1 A)

Generate two time series $x_t = -0.8x_{t-2} + w_t$ where $x_0 = x_1 = 0$ and $x_t = \cos(\frac{2\pi t}{5})$ with 100 observations each. Apply a smoothing filter $v_t = 0.2(x_t + x_{t-1} + x_{t-2} + x_{t-3} + x_{t-4})$ to these two series and compare how the filter has affected them.

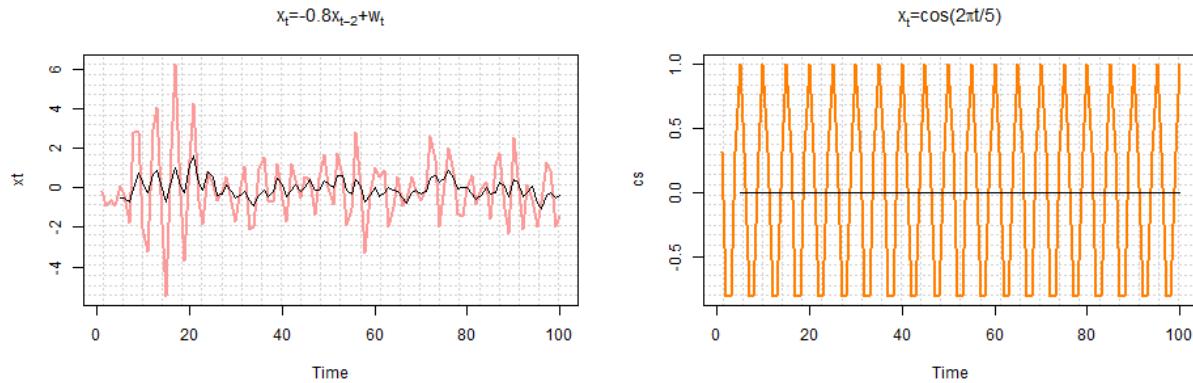


Figure 1: Comparation of ACF plots.

1.2 B)

Consider time series $x_t - 4x_{t-1} + 2x_{t-2} + x_{t-5} = w_t + 3w_{t-2} + w_{t-4} - 4w_{t-6}$. Write an appropriate R code to investigate whether this time series is causal and invertible.

```
## [1] "Not casual or invertible"
```

1.3 C)

Use built-in R functions to simulate 100 observations from the process $x_t + \frac{3}{4}x_{t-1} = w_t - \frac{1}{9}w_{t-2}$, compute sample ACF and theoretical ACF, use seed 54321. Compare the ACF plots.

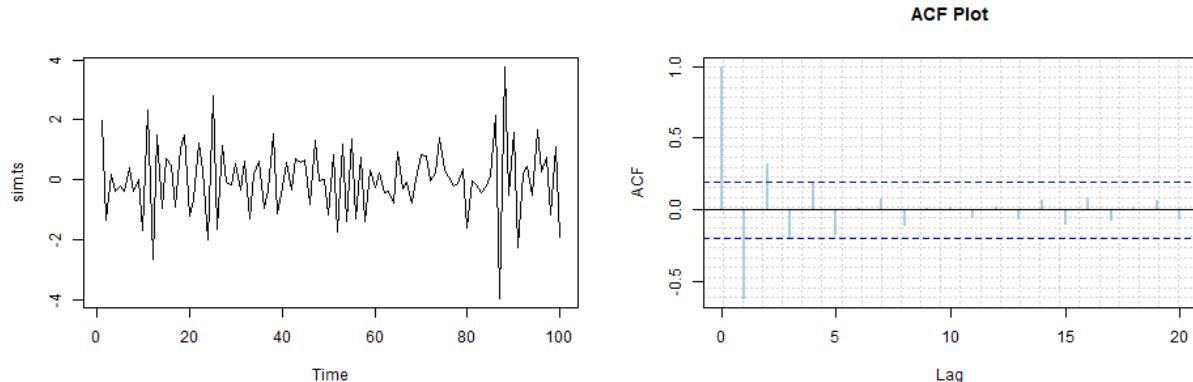


Figure 2: Comparation of ACF plots.

2 Visualization, detrending and residuals analysis of Rhine data

The dataset **Rhine.csv** contains monthly concentrations of total nitrogen in the Rhine River in the period 1989-2002.

2.1 A)

Import the data to R, convert it appropriately to *ts* object (use function *ts()*) and explore it by plotting the time series, creating scatterplots of x_t against x_{t-1}, \dots, x_{t-12} . Analyze the time series plot and scatter plots: Are there any trends, linear or seasonal, in the time series? When during the year is the concentration highest? Are there any special patterns in the data or scatterplots? Does the variance seem to change over time? Which variables in the scatterplots seem to have a significant relation to each other?

Figure 3 shows the plotted data as a time series giving a sense of having a descending trend and also a possible sense of periodicity. As instructed in this lab, by using the scatterplots with lag, we can see in the beginning as lag 1 that there is a significant level of correlation.

This in itself, is a signal that indeed there is a semblance of seasonality to the data with that lag. As we move along the Scatterplot in figure 4 both left and right, it shows that when compared with lag 5 or 6 the correlation is almost non-existent, but then again, proving its periodicity as we move further in the lag up to 12 then again we can see a high correlation of the data with itself, thus proving its seasonality.

It is also worth to mention that this seasonality seems to appear yearly, given the lag being 1 and 12 as well as the meaning of the unit being 1 month.

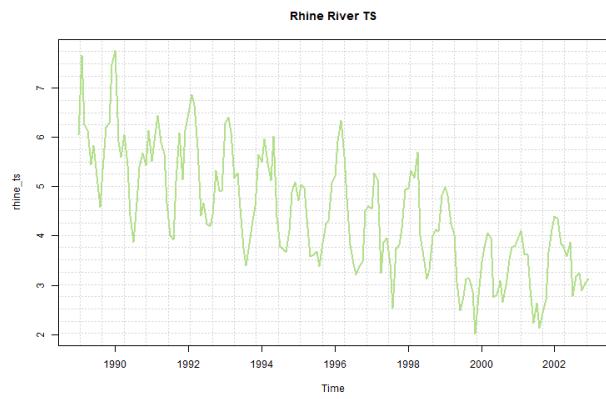


Figure 3: Time series plot

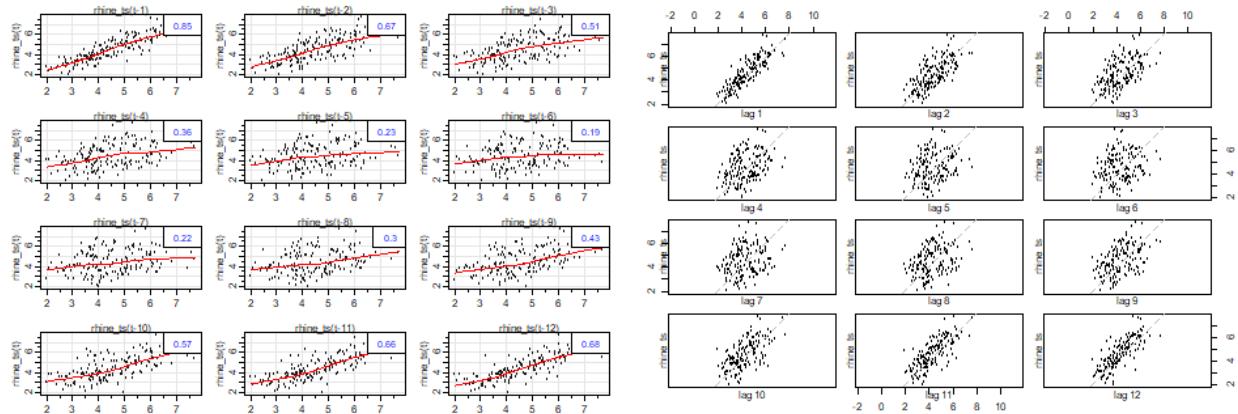


Figure 4: Plotted data as a time series plot with *lag1()* left and *lag()* right.

2.2 B)

Eliminate the trend by fitting a linear model with respect to t to the time series is there a significant time trend? Look at the residual pattern and the sample ACF of the residuals and comment on this pattern might be related to the seasonality series.

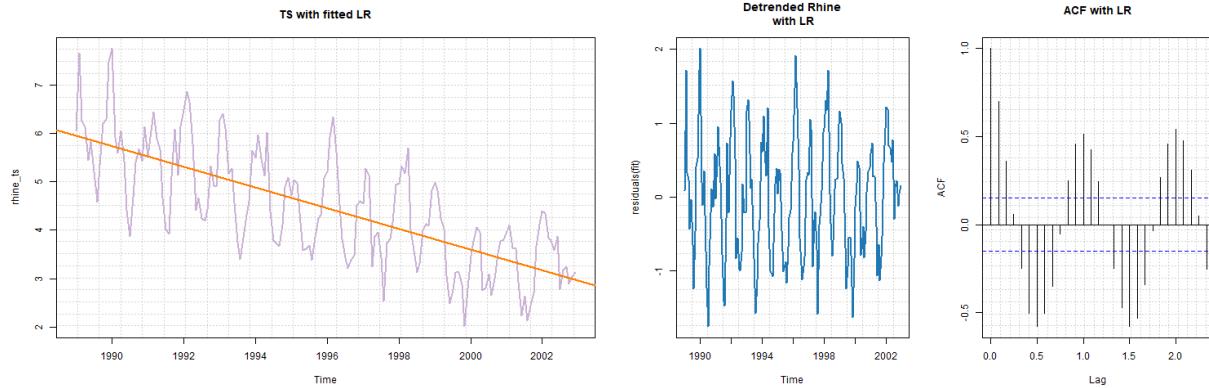


Figure 5: *Analisis of the TS with a fitted LR at the left and its residuals analysys..*

Figure 5, first plot, shows the TS fitted with a Linear Regression, this is showing a decreasing trend. In the other half of the figure, e can see the residual pattern showing an interesting um and down movement, or simmilar to seasonal, remanising of what the lag scatterplots showed.

Finally in the ACF of the residuals it's showing what could be reffered to a beautiful seadonality, thus also related to the lag scatterplots from before.

2.3 C)

Eliminate the trend by fitting a kernel smoother with respect to t to the time series (choose a reasonable bandwidth yourself so the fit looks reasonable). Analyze the residual pattern and the sample ACF of the residuals and compare it to the ACF from step b). Conclusions? Do residuals seem to represent a stationary series?

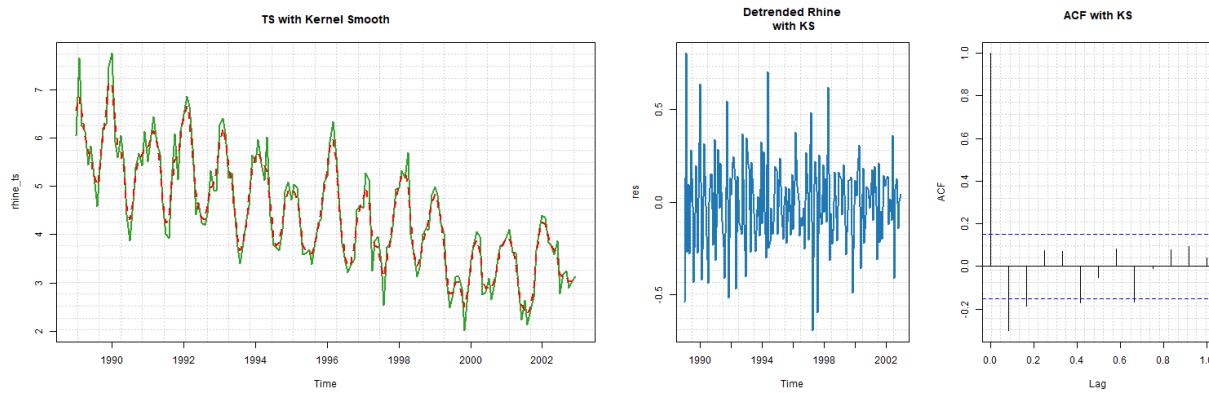


Figure 6: *Analisis of the TS with a smoothing kernel at the left and its residuals analysys.*

Figure 6 has both the visual result of what the Kernel smooth does to the TS data. The smoother in a way, tries to reduce w , white noise, thus its expected that the seasonality before observed in the data with a linear regression will also be less present. The ressimuals do seem to represent the behavior closer to a stationary series.

2.4 D)

Eliminate the trend by fitting the following so-called seasonal means model:

$$x_t = \alpha_0 + \alpha_1 t + \beta_1 I(\text{month} = 2) + \dots + \beta_{12} I(\text{month} = 12) + w_t$$

where $I(x) = 1$ if x is true and 0 otherwise. Fitting of this model will require you to augment data with a categorical variable showing the current month, and then fitting a usual linear regression. Analyze the residual pattern and the ACF of residuals.

Following this method, in figure 7, we can see that in the detrended plot with seasonal mean the seasonal behavior is little bit more visible but still not quite there. Although the ACF does show a more understandable seasonality with the lag unit being a month.

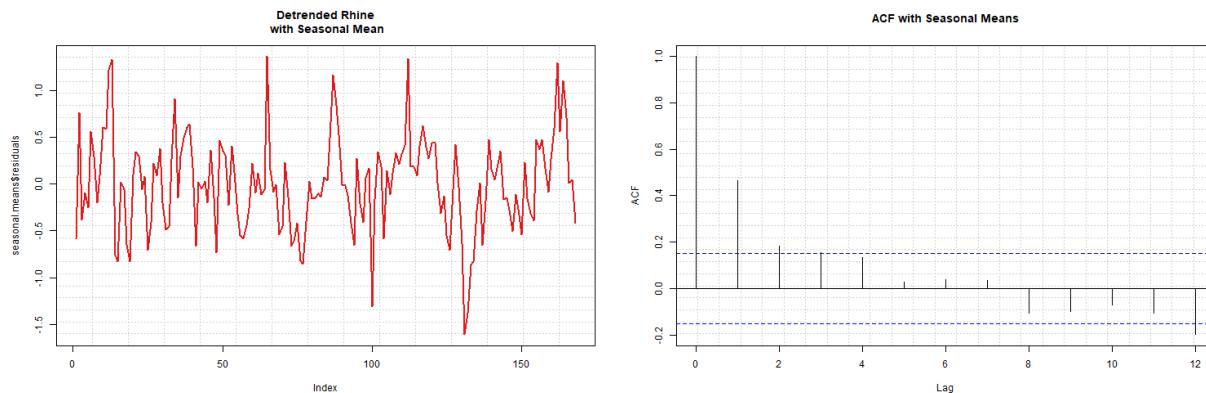


Figure 7: Time series after trend elimination with a seasonal means model.

2.5 E)

Perform stepwise variable selection in model from step d). Which model gives you the lower AIC value? Which variables are left in the model?

```
## [1] -202.0227

## Start:  AIC=-202.02  ## TotN_conc ~ month_enc + Time  ##  ##
RSS      AIC  ## <none>          43.237 -202.023 ## - month_enc 11   68.524
111.761 -64.477 ## - Time       1    118.387 161.624   17.499

## [1] -202.0227
```

According to the shown results both seem to perform equally well with the same respective AIC score.

3 Analisys of oil and gas time series

Weekly time series *oil* and *gas* present in the package *astsa* show the oil prices in dollars per barrel and gas prices in cents per dollar.

3.1 A)

Plot the given time series in the same graph. Do they look like stationary series? Do the processes seem to be related to each other? Motivate your answer.

Figure 8 (left plot) shows what we would describe visually as something that is not a stationary series.

3.2 B)

Apply log-transform to the time series and plot the transformed data. In what respect did this transformation made the data easier for the analysis?

Figure 8 (right) does shows that the applied transformation made the data easier at least for visual analisys. This is thanks to the 2d reduction that a log transformation brings to greater numbers hen compared to relative small ones, allowing a clearer comparison loosing less detail of the smaller movements in the TS.

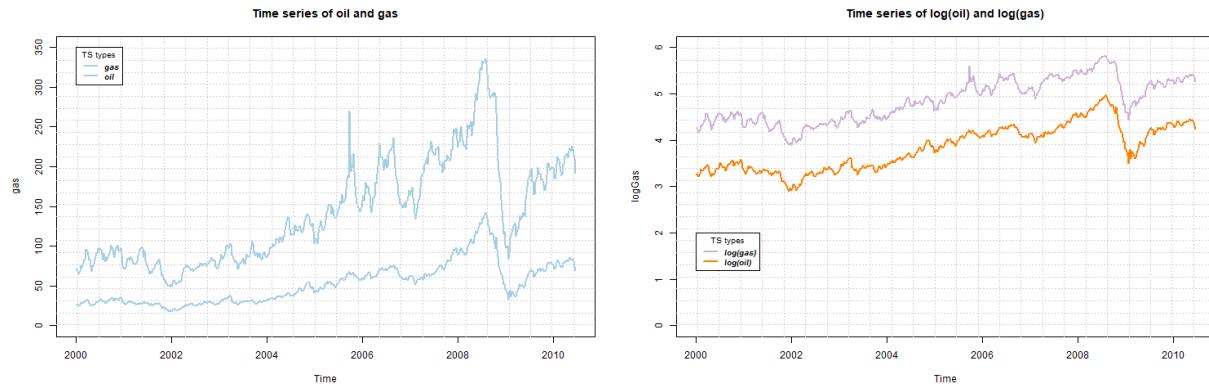


Figure 8: Time series before (to the left) and after (to the right) the agumentation of a log-transformation.

3.3 C)

To eliminate trend, compute the first difference of the transformed data, plot the detrended series, check their ACFs and analyse the obtained plots. Denote the data obtained here as x_t (oil) and y_t (gas).

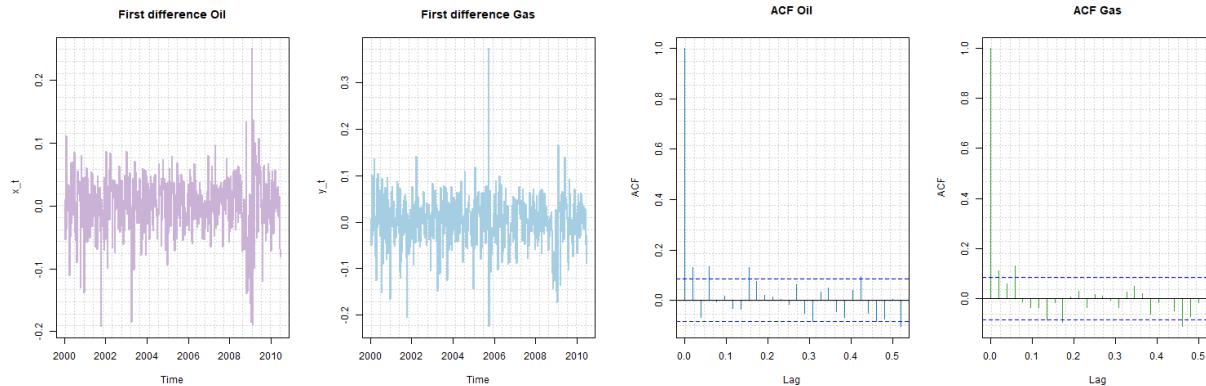


Figure 9: Time series analysis of Oil and Gas First difference and ACF respectively.

3.4 D)

Exhibit scatterplots of x_t and y_t for up to three weeks of lead time of x_t ; include a nonparametric smoother in each plot and comment the results: are there outliers? Are the relationships linear? Are there changes in the trend?

Figure 10 does show a slight decrease of the linear relationship as the lead increases.

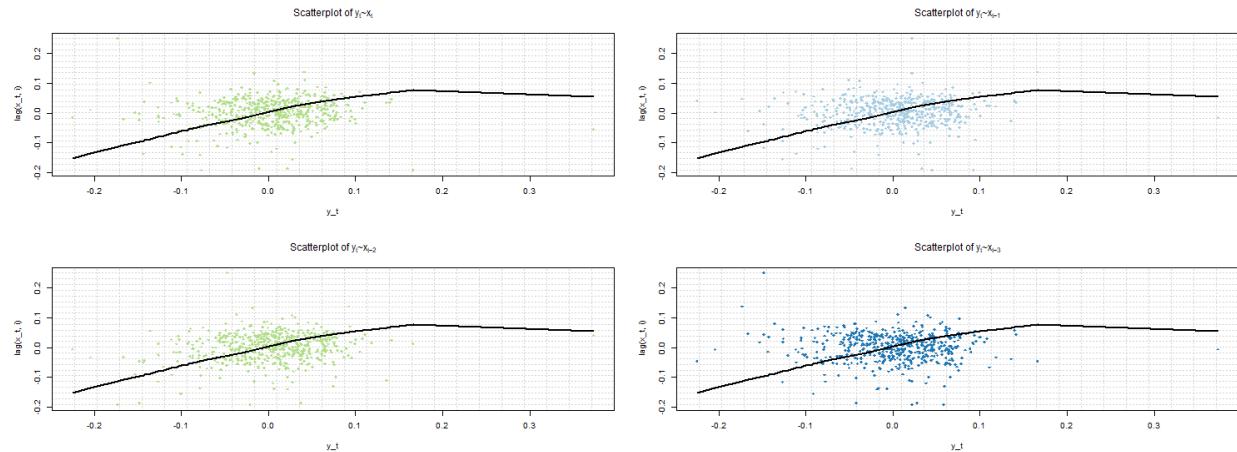


Figure 10: Time series analysis scatterplot with 0-3 weeks of lead time.

3.5 E)

Fit the following model: $y_t = \alpha_0 + \alpha_1 I(x_t > 0) + \beta_1 x_t + \beta_2 x_{t-1} + w_t$ and check which coefficients seem to be significant. How can this be interpreted? Analyze the residual pattern and the ACF of the residuals.

```
## (Intercept) time(rhine_ts) ## TRUE TRUE
```

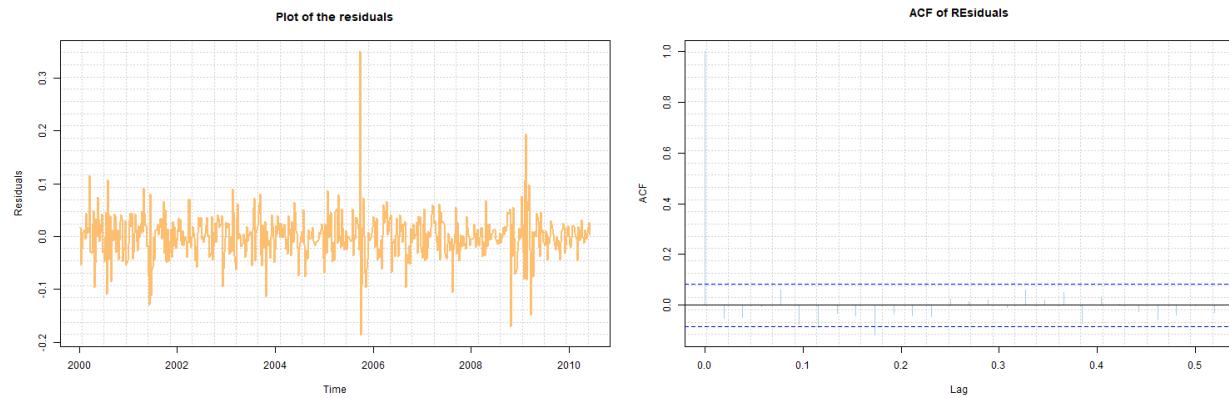


Figure 11: *Time series analysis scatterplot with 0-3 weeks of lead time.*

4 Appendix

4.1 Code

4.1.1 Code used for Computations with simulate data

```

1 library(ggplot2)
2 library(ggfortify)
3 # autoplot(USAccDeaths)
4 library(forecast)
5 set.seed(54321)
6 require(RColorBrewer)
7 pal <- brewer.pal(9, "Paired")
8 xt = arima.sim(list(order = c(2, 0, 0), ar = c(0, -0.8)),
9                 n = 100, start.innov = c(0, 0), n.start = 2)
10 t <- 1:100
11 cs = cos((2 * pi * t)/5)
12 # apply the filter
13 f_coefs = rep(0.2, 5)
14 xt_filtered = filter(xt, filter = f_coefs, sides = 1)
15 cs_filtered = filter(cs, filter = f_coefs, sides = 1)
16 col1 <- sample(pal, 1)
17 col2 <- sample(pal, 1)
18 check_causality <- function(z) {
19     return(sqrt(Re(z)^2 + Im(z)^2))
20 }
21 inv_caus_func <- function(AR_operator, MA_operator) {
22     n1 <- length(AR_operator) - 1
23     n2 <- length(MA_operator) - 1
24     res <- polyroot(AR_operator)
25     res2 <- polyroot(MA_operator)
26     causality <- sapply(res, function(y) {
27         check_causality(y)
28     })
29     # print(causality>1)
30     invert <- sapply(res2, function(y) {
31         check_causality(y)
32     })
33     # print(invert>1) print(sum(causality>1))
34     # print(sum(invert>1))
35     if ((sum(causality > 1) == n1) & (sum(invert > 1) ==
36         n2)) {
37         print("The series is invertible and causal")
38     } else if (sum(causality > 1) == n1) {
39         print("The series is causal only!")
40     } else if (sum(invert > 1) == n2) {
41         print("The series is invertible only !")
42     } else {
43         print("Not causal or invertible ")
44     }
45 }
46 ar_operator = c(1, -4, 2, 0, 0, 1)
47 ma_operator = c(1, 0, 3, 0, 1, 0, -1)
48 inv_caus_func(ar_operator, ma_operator)

```

4.1.2 Code used for Visualization, detrending and residuals analysis of Rhine data

```

1 library(astsa)
2 rhine <- read.csv2("data/Rhine.csv", sep = ";")
3 rhine_ts <- ts(rhine[, 4], start = c(rhine[1, 1], 1), end = c(2002,
4   12), frequency = 12)
5 fit <- lm(rhine_ts ~ time(rhine_ts), na.action = NULL)
6 # summary(fit)
7 col3 <- sample(pal, 1)
8 col4 <- sample(pal, 1)
9 plot.ts(rhine_ts, col = col3, main = "TS with fitted LR",
10   panel.first = grid(25, 25), lwd = 2)
11 abline(fit, col = col4, lwd = 2)
12 legend(130, 7, legend = c("TS Rhine", "LR"), col = c(col3,
13   col4), lty = 1, cex = 0.8, text.font = 4, bg = "white",
14   lwd = 2)
15 par(mfrow = c(1, 2))
16 plot(residuals(fit), main = "Detrended Rhine\n with LR",
17   col = sample(pal, 1), panel.first = grid(15, 25), lwd = 2)
18 acf(residuals(fit), 28, main = "ACF with LR", panel.first = grid(25,
19   25))
20 kernelSmooth <- ksmooth(time(rhine_ts), rhine_ts, "normal",
21   bandwidth = 0.2)
22 col5 <- sample(pal, 1)
23 plot.ts(rhine_ts, col = col5, main = "TS with Kernel Smooth",
24   panel.first = grid(25, 25), lwd = 2)
25 lines(kernelSmooth, col = "red", lwd = 2, lty = 2)
26 legend(130, 7, legend = c("TS Rhine", "KS"), col = c(col5,
27   "red"), lty = 1, cex = 0.8, text.font = 4, bg = "white",
28   lwd = 2)
29 res <- (rhine_ts - kernelSmooth$y)
30 par(mfrow = c(1, 2))
31 plot(res, main = "Detrended Rhine\n with KS", col = sample(pal,
32   1), panel.first = grid(15, 25), lwd = 2)
33 acf(res, 12, main = "ACF with KS", panel.first = grid(25,
34   25))
35 new_rhine <- cbind(rhine, month_enc = c("January", "February",
36   "March", "April", "May", "June", "July", "August", "September",
37   "October", "November", "December"))
38 new_rhine$month_enc <- as.factor(new_rhine$month_enc)
39 str(new_rhine)
40 seasonal.means <- lm(TotN_conc ~ month_enc + Time, data = new_rhine)
41 # plot(seasonal.means$)
42 par(mfrow = c(1, 2))
43 plot(seasonal.means$residuals, main = "Detrended Rhine\n with Seasonal Mean",
44   col = sample(pal, 1), panel.first = grid(15, 25), lwd = 2,
45   type = "l")
46 acf(seasonal.means$residuals, 12, main = "ACF with Seasonal Means",
47   panel.first = grid(25, 25))
48 temp <- step(seasonal.means, direction = "both", trace = 0,
49   steps = 1)
50 temp$anova$AIC
51 summary(temp)
52 library(MASS)

```

```

53 temp1 <- stepAIC(seasonal.means, direction = "both")
54 temp1$effects

```

4.1.3 Code used for Analisys of oil and gas time series

```

1 library(astsa)
2 rhine <- read.csv2("data/Rhine.csv", sep = ";")
3 rhine_ts <- ts(rhine[, 4], start = c(rhine[1, 1], 1), end = c(2002,
4 12), frequency = 12)
5 fit <- lm(rhine_ts ~ time(rhine_ts), na.action = NULL)
6 # summary(fit)
7 col3 <- sample(pal, 1)
8 col4 <- sample(pal, 1)
9 plot.ts(rhine_ts, col = col3, main = "TS with fitted LR",
10 panel.first = grid(25, 25), lwd = 2)
11 abline(fit, col = col4, lwd = 2)
12 legend(130, 7, legend = c("TS Rhine", "LR"), col = c(col3,
13 col4), lty = 1, cex = 0.8, text.font = 4, bg = "white",
14 lwd = 2)
15 par(mfrow = c(1, 2))
16 plot(residuals(fit), main = "Detrended Rhine\n with LR",
17 col = sample(pal, 1), panel.first = grid(15, 25), lwd = 2)
18 acf(residuals(fit), 28, main = "ACF with LR", panel.first = grid(25,
19 25))
20 kernelSmooth <- ksmooth(time(rhine_ts), rhine_ts, "normal",
21 bandwidth = 0.2)
22 col5 <- sample(pal, 1)
23 plot.ts(rhine_ts, col = col5, main = "TS with Kernel Smooth",
24 panel.first = grid(25, 25), lwd = 2)
25 lines(kernelSmooth, col = "red", lwd = 2, lty = 2)
26 legend(130, 7, legend = c("TS Rhine", "KS"), col = c(col5,
27 "red"), lty = 1, cex = 0.8, text.font = 4, bg = "white",
28 lwd = 2)
29 res <- (rhine_ts - kernelSmooth$y)
30 par(mfrow = c(1, 2))
31 plot(res, main = "Detrended Rhine\n with KS", col = sample(pal,
32 1), panel.first = grid(15, 25), lwd = 2)
33 acf(res, 12, main = "ACF with KS", panel.first = grid(25,
34 25))
35 new_rhine <- cbind(rhine, month_enc = c("January", "February",
36 "March", "April", "May", "June", "July", "August", "September",
37 "October", "November", "December"))
38 new_rhine$month_enc <- as.factor(new_rhine$month_enc)
39 str(new_rhine)
40 seasonal.means <- lm(TotN_conc ~ month_enc + Time, data = new_rhine)
41 # plot(seasonal.means$)
42 par(mfrow = c(1, 2))
43 plot(seasonal.means$residuals, main = "Detrended Rhine\n with Seasonal Mean",
44 col = sample(pal, 1), panel.first = grid(15, 25), lwd = 2,
45 type = "l")
46 acf(seasonal.means$residuals, 12, main = "ACF with Seasonal Means",
47 panel.first = grid(25, 25))
48 temp <- step(seasonal.means, direction = "both", trace = 0,

```

```
49     steps = 1)
50 temp$anova$AIC
51 summary(temp)
52 library(MASS)
53 temp1 <- stepAIC(seasonal.means, direction = "both")
54 temp1$effects
```

LAB2_TS

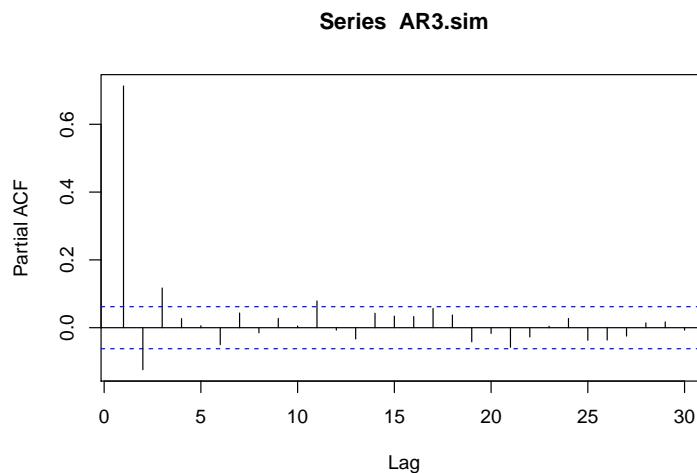
Andreas C Charitos (andch552), Ruben Muñoz (rubmu773)

9/30/2019

Assignment 1. Computations with simulated data

a)

Generate 1000 observations from AR(3) process with $\phi_1 = 0.8$, $\phi_2 = -0.2$, $\phi_3 = 0.1$. Use these data and the definition of PACF to compute ϕ_{33} from the sample, i.e. write your own code that performs linear regressions on necessarily lagged variables and then computes an appropriate correlation. Compare the result with the output of function `pacf()` and with the theoretical value of ϕ_{33}



The partial autocorrelation is the association between X_t and X_{t+k} with the linear dependence of X_{t+1} through X_{t+k-1} removed. Given by the formula :

$$pacf(X_t, X_{t+k}) = \text{Corr}(X_t, X_{t+k} | X_{t+1} = x_{t+1}, \dots, X_{t+k-1} = x_{t+k-1})$$

The results we obtain are similar calculating the corellation with linear regression between $X_t \sim X_{t-1} + X_{t-2}$ and $X_{t-3} \sim X_{t-1} + X_{t-2}$ and the output of the `pacf()` function for the simulated data and the theoretical PACF.

Est.Corr	Sim.PACF	Theo.PACF
0.1146076	0.1170643	0.1

b)

Simulate an AR(2) series with $\phi_1 = 0.8$, $\phi_2 = 0.1$ and $n = 100$. Compute the estimated parameters and their standard errors by using three methods: method of moments (Yule-Walker equations), conditional least squares and maximum likelihood (ML) and compare their results to the true values. Which method does seem to give the best result? Does theoretical value for ϕ_2 fall within confidence interval for ML estimate?

Table with the estimated coefficients

	phi1	phi2
YW	0.8571752	-0.0199902
OLS	0.9386075	-0.0910831
MLE	0.9015078	-0.0354404
TRUE	0.8000000	0.1000000

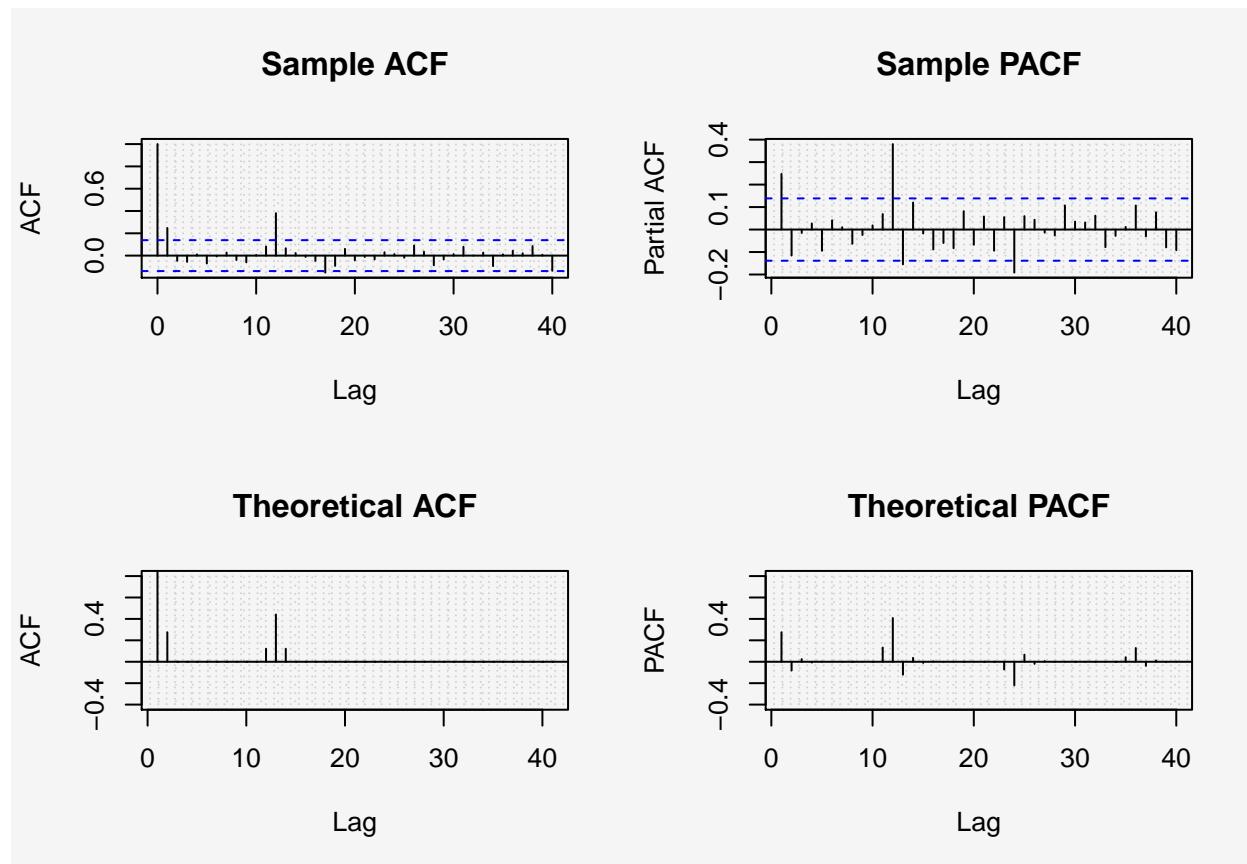
The above table gives the estimated coefficients given the 3 methods. As we can see the Yule-Walker method seems to have the most accurate estimates.

```
## The esitmated interval is : -0.2192624 0.1483816
## The value of phi_2 is within the estimated interval?
## [1] TRUE
```

As we can see the value of ϕ_2 is within the CI for the ML estimate.

c)

Generate 200 observations of a seasonal $ARIMA(0, 0, 1)x(0, 0, 1)_{12}$ model with coefficients $\Theta = 0.6$ and $\theta = 0.3$ by using `arima.sim()`. Plot sample ACF and PACF and also theoretical ACF and PACF. Which patterns can you see at the theoretical ACF and PACF? Are they repeated at the sample ACF and PACF?



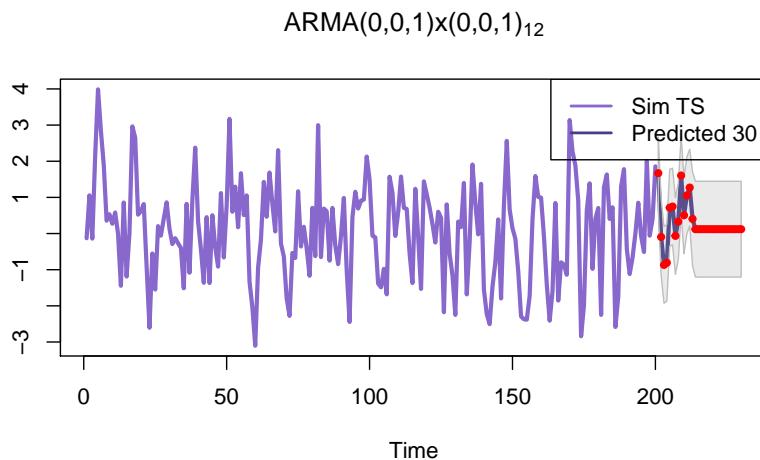
As we can see from the plots of the simulated and theoretical ACF the pattern at lag 1 and the pattern at

lag 11-13 are observed on both plots. For the PACF plots we can see some seasonal patterns that are in the theoretical PACF are also present at the simulated PACF.

d)

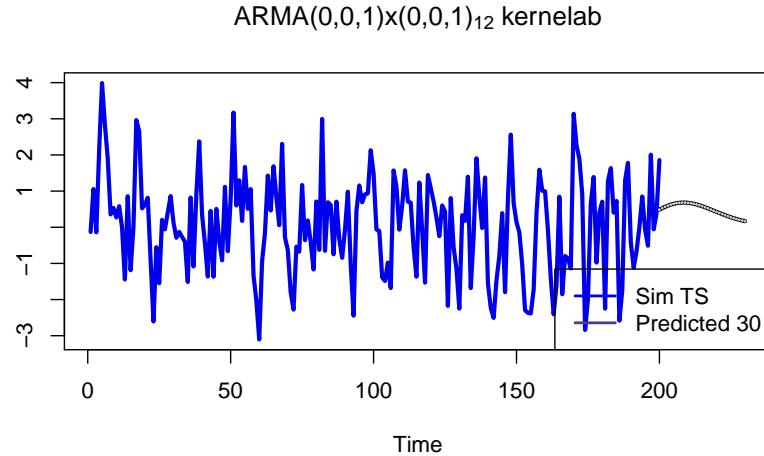
Generate 200 observations of a seasonal $ARIMA(0,0,1)\times(0,0,1)_{12}$ model with coefficients $\Theta = 0.6$ and $\theta = 0.3$ by using arima.sim(). Fit $ARIMA(0,0,1)\times(0,0,1)_{12}$ model to the data, compute forecasts and a prediction band 30 points ahead and plot the original data and the forecast with the prediction band. Fit the same data with function gausspr from package kernlab (use default settings). Plot the original data and predicted data from t=1 to t=230. Compare the two plots and make conclusions.

Plot of predictions with $ARIMA(0,0,1)x(0,0,1)_{12}$



Plot of predictions with kernlab

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

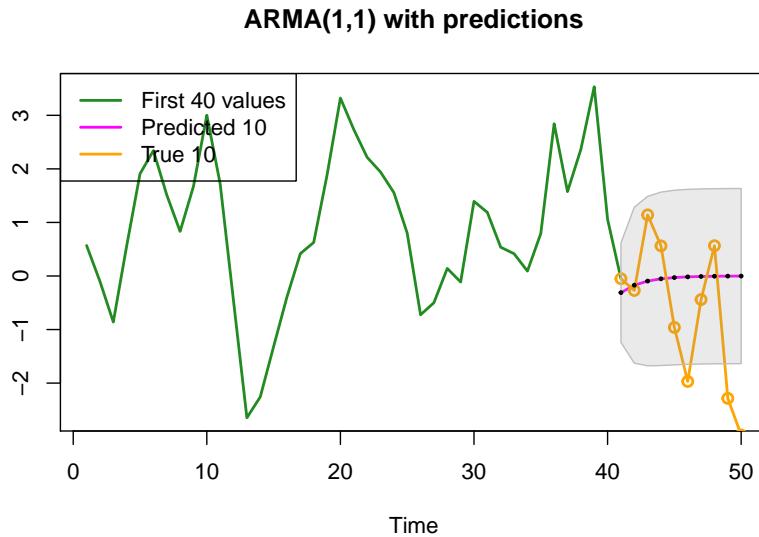


As we can see comparing the plots using the $ARIMA(0,0,1)x(0,0,1)_{12}$ we fitted and the results from the kernelab the $ARIMA(0,0,1)x(0,0,1)_{12}$ was able to produce better predictions compared to kernelab. This result may be explained due to the fact that kernelab wasn't able to capture the seasonality or trend because the prediction is based on the width of the kernel and might some previous values not include in the kernel estimate. Also the gaussian kernel which is symmetric returns the most probable prediction (or mean prediction).

e)

Generate 50 observations from $ARMA(1,1)$ process with $\phi = 0.7$, $\theta = 0.5$. Use first 40 values to fit an $ARMA(1,1)$ model with $\mu = 0$. Plot the data, the 95% prediction band and plot also the true 10 values that you initially dropped. How many of them are outside the prediction band? How can this be interpreted?

Plot of the predictions for ARMA(1,1)



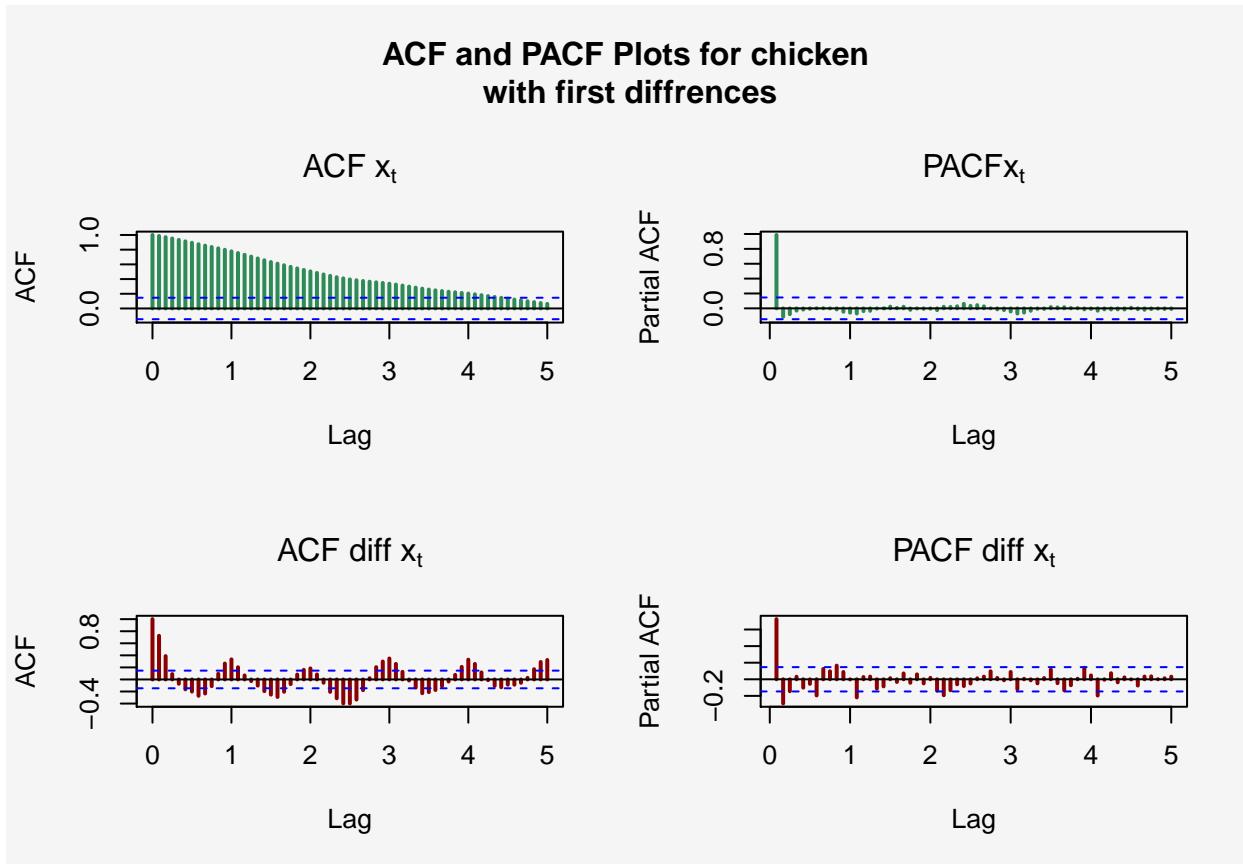
```
## The number of the values outside the prediction band is: 3
```

Assingment 2.ACF and PACF diagnostics

a)

For data series chicken in package astsa (denote it by x_t), plot 4 following graphs up to 40 lags: $ACF(x_t)$, $PACF(x_t)$, $ACF(\nabla x_t)$, $PACF(\nabla x_t)$ (group them in one graph). Which $ARIMA(p, d, q)$ or $ARIMA(p, d, q)x(P, D, Q)_s$ models can be suggested based on this information only? Motivate your choice.

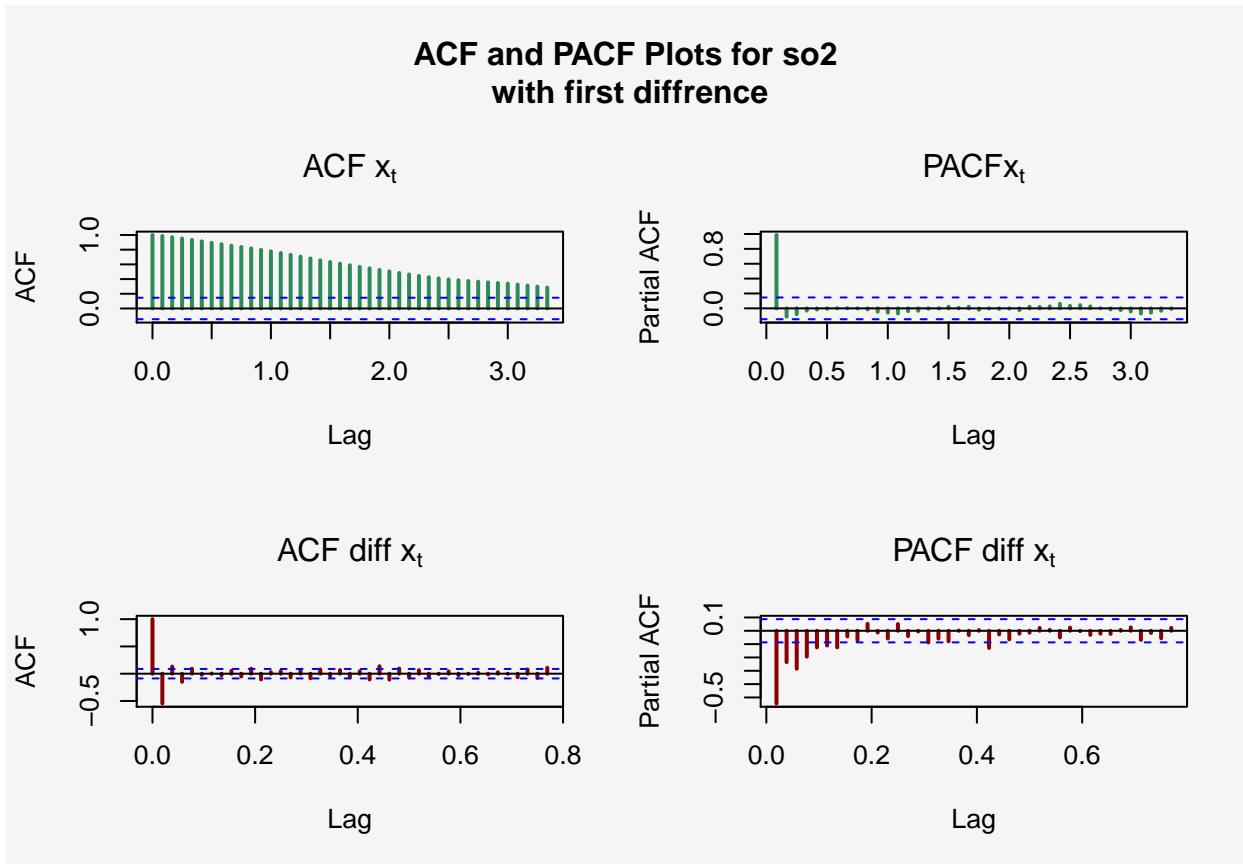
ACF-PACF Plots for chicken data



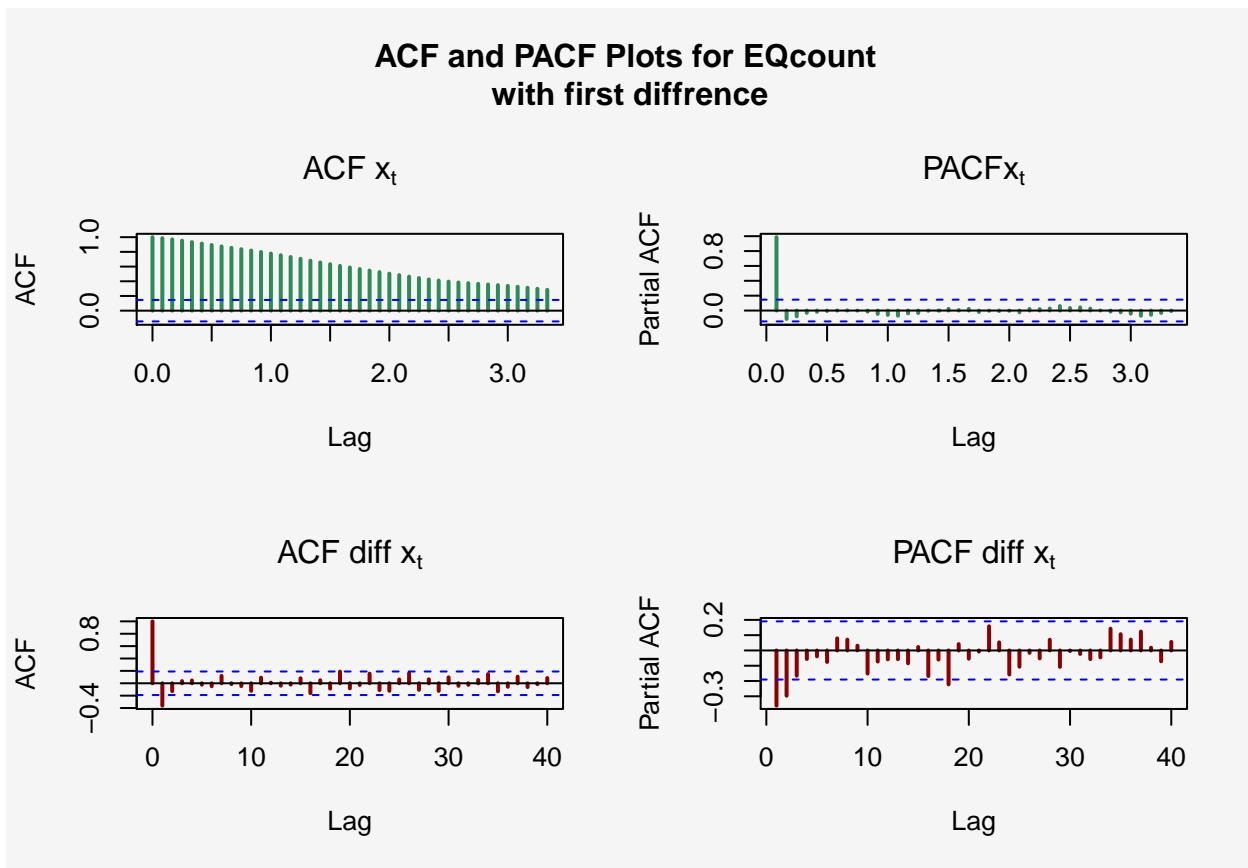
b)

Repeat step 1 for the following datasets: so2, EQcount, HCT in package astsa

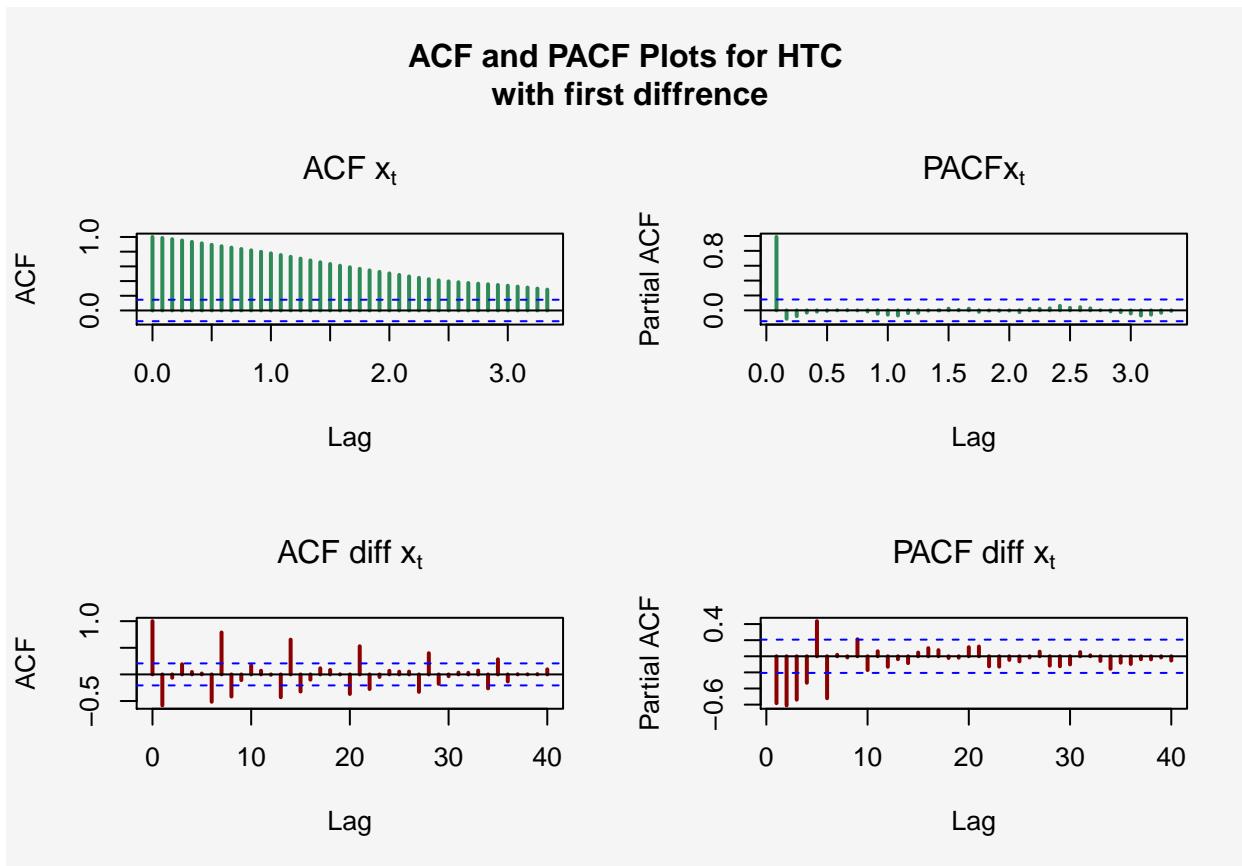
ACF-PACF PLots for so2 data



ACF-PACF PLots for EQcount data



ACF-PACF PLots for HCT data



Summary table for the ACF-PACF

	chicken	diff(chicken)	so2	diff(so2)
ACF	slow decay differencing needed	sesonal cycle patter presen	fast decay but need differencing	tails off after lag 0.02
PACF	seasonal pattern present	cut off after lag1	tails off quickly	tails off after 0.18

	EQcount	diff(EQcount)	HCT	diff(ECT)
ACF	tails off after lag 8	tails off after lag 1	tails off after lag 18	slow decay tails off after lag 1
PACF	the bars are in the boarders	tails off after lag1	tails off after lag7	tails off after lag5

From the ACF-PACF Plots and the above table we can propose the following models :

- For the chicken data Starting from the nonseasonal part we can suggest an AR(2) from ACF-PACF and for the seasonal s=12 an MA(1) The final model is a $SARIMA(2, 1, 0)x(1, 0, 0)_{12}$

- For the so2 data It's very hard to distinguish a model but maybe an $ARMA(1, 1, 1)$ according to ACF-PACF plots of difference.
- For the EQcount From ACF of EQcount difference is needed and according to ACF of difference and MA(1).The final model is $ARIMA(0, 1, 1)$
- For the HCT According to ACF of HCT difference is needed.From PACF of difference we can suggest an $AR(5)$ and form ACF an $MA(1)$. The final model is $ARIMA(5, 1, 1)$

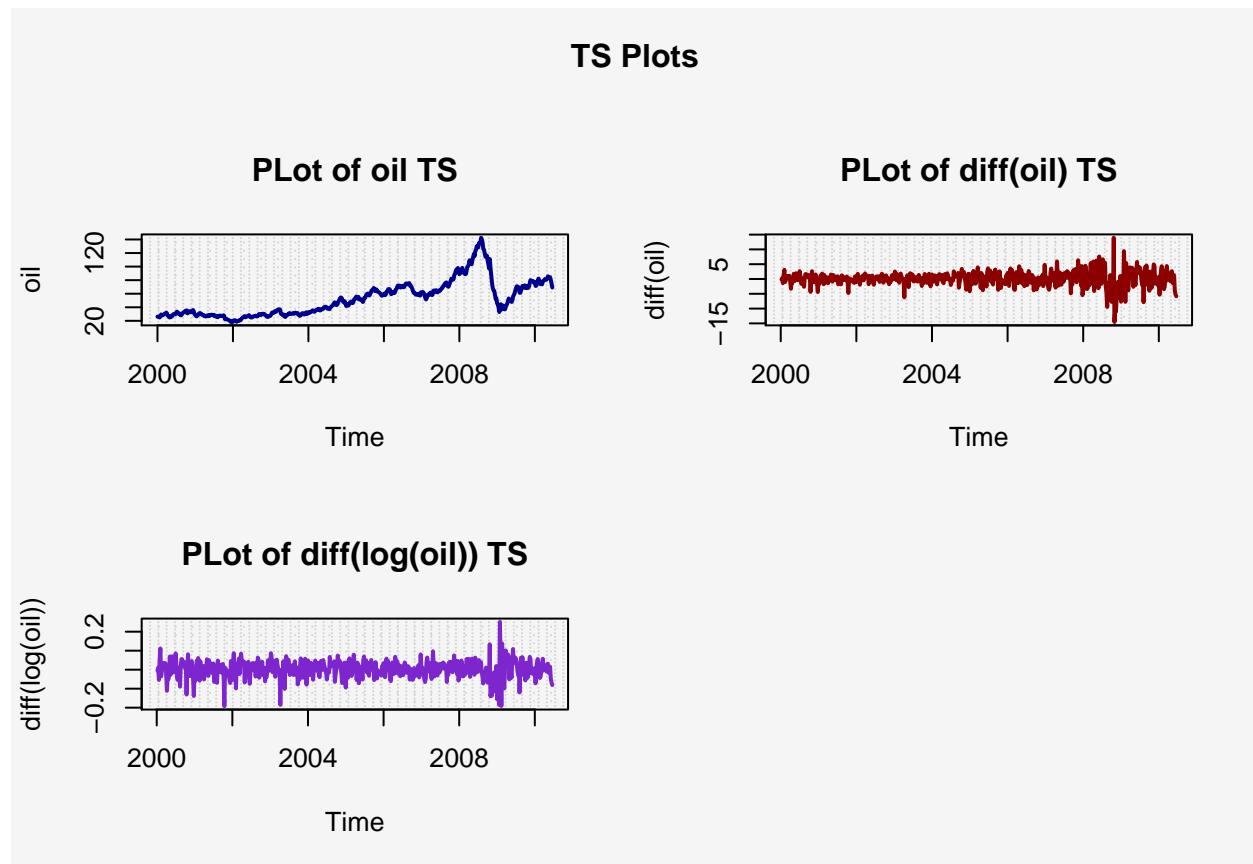
Assignment 3.Arima modeling cycle

a)

Find a suitable $ARIMA(p, d, q)$ model for the data set oil present in the library astsa. Your modeling should include the following steps in an appropriate order: visualization, unit root test, detrending by differencing (if necessary), transformations (if necessary), ACF and PACF plots when needed, EACF analysis, Q-Q plots, Box-Ljung test, ARIMA fit analysis, control of the parameter redundancy in the fitted model. When performing these steps, always have 2 tentative models at hand and select one of them in the end. Validate your choice by AIC and BIC and write down the equation of the selected model. Finally, perform forecasting of the model 20 observations ahead and provide a suitable plot showing the forecast and its uncertainty.

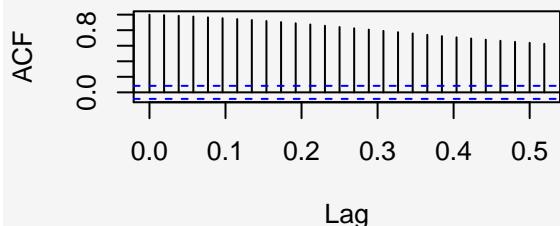
TS -ACF-PACF Plots

We start by making some diagnostic plots for the original time series ,the first difference and the log first difference.

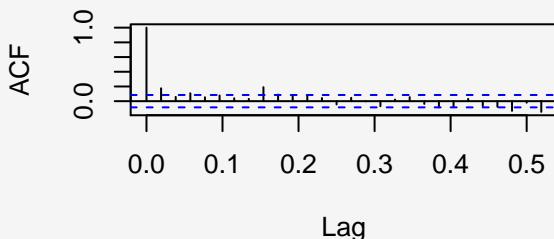


ACF Plots

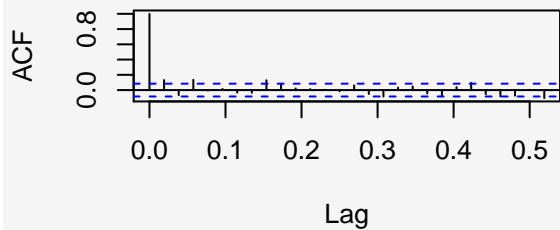
Series oil

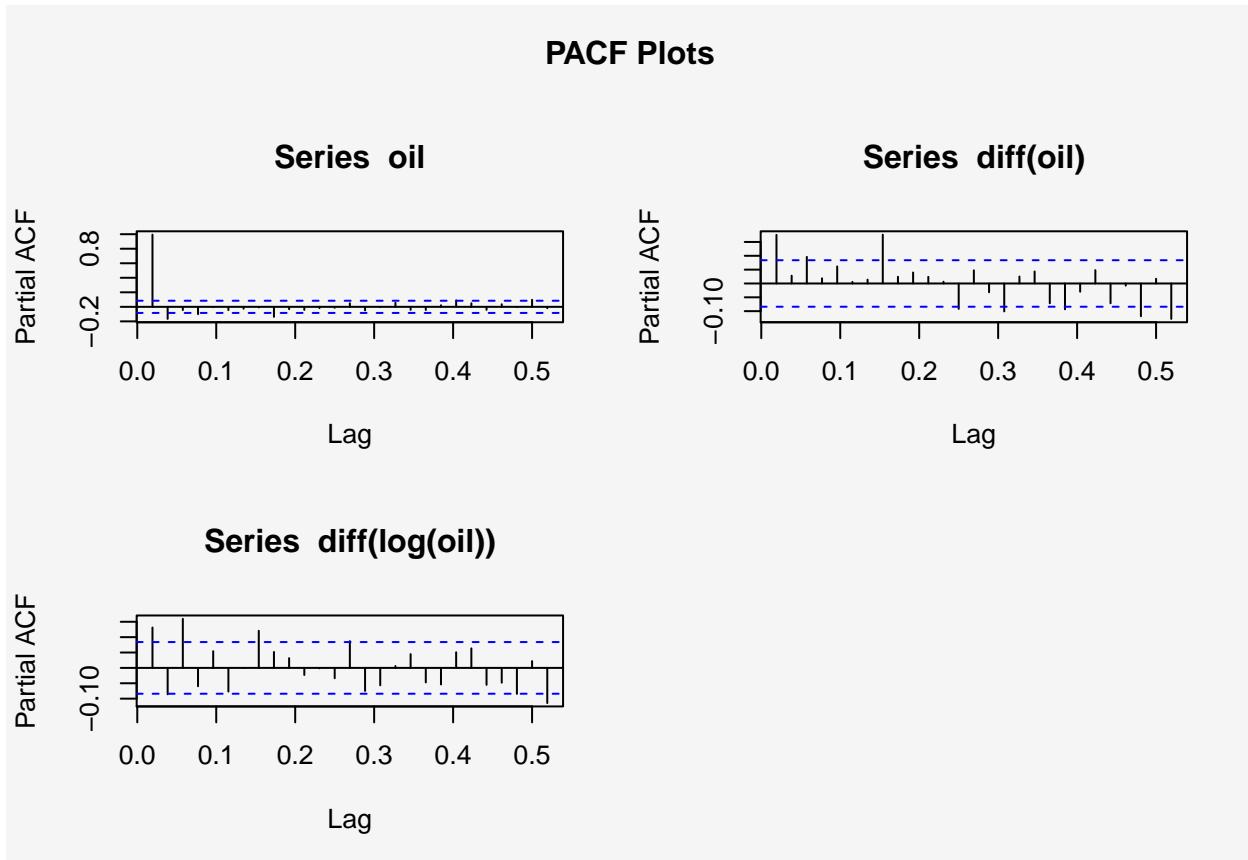


Series diff(oil)



Series diff(log(oil))

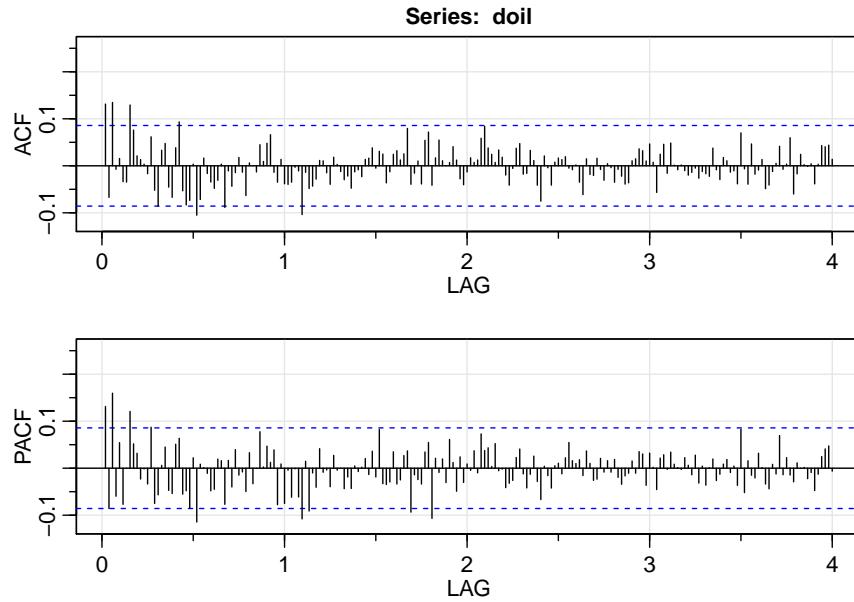




From the plots we conclude that working with the log of first difference of the original data seems reasonable because we have a stationary process. Here we report the Dickey-Fuller test for stationarity and as we can see the p-value suggests that data are stationary.

```
##  
## Augmented Dickey-Fuller Test  
##  
## data: doil  
## Dickey-Fuller = -6.3708, Lag order = 8, p-value = 0.01  
## alternative hypothesis: stationary
```

Plot of the ACF-PACF for $\nabla \log(x_t)$



Now we are going to use the eacf in order to identify best model combinations.

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o x o o o o x o o o o o o o
## 1 x o x o o o o o x o o o o o o o
## 2 x x x o o o o o o x o o o o o o o
## 3 x x x o o o o o o o x o o o o o o
## 4 x o x o o o o o o o x o o o o o o
## 5 x x x o x o o x o o o o o o o o o
## 6 o x x o x x o x o o o o o o x o
## 7 o x x x x x x o x o o o o o o o
```

From the matrix we distinguish 2 models 1.ARMA(0,3) and ARMA(1,1).We are going to investigate each separately

ARMA(0,3)

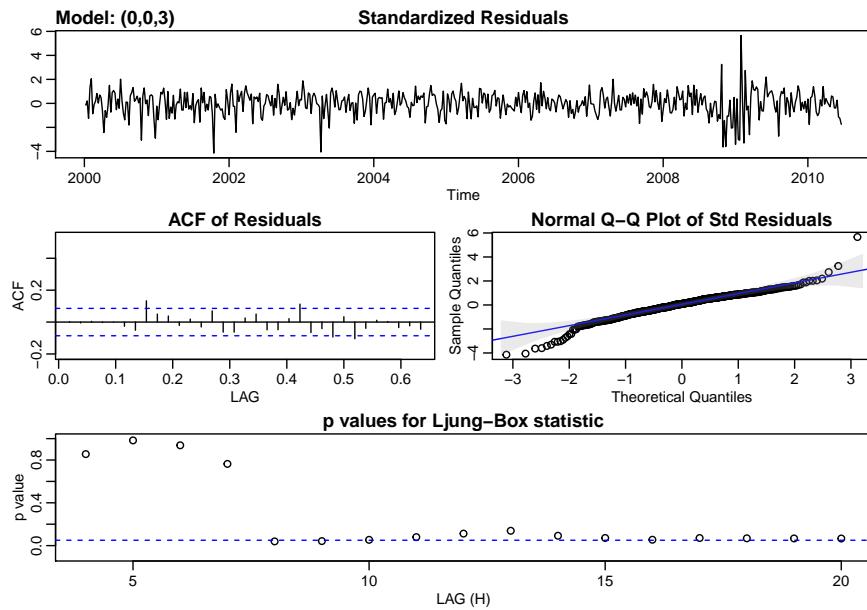
Diagnostic Plots for residuals

```
## initial value -3.058495
## iter  2 value -3.086110
## iter  3 value -3.086980
## iter  4 value -3.087501
## iter  5 value -3.087521
## iter  6 value -3.087521
## iter  7 value -3.087522
## iter  8 value -3.087522
## iter  9 value -3.087522
## iter  9 value -3.087522
## iter  9 value -3.087522
```

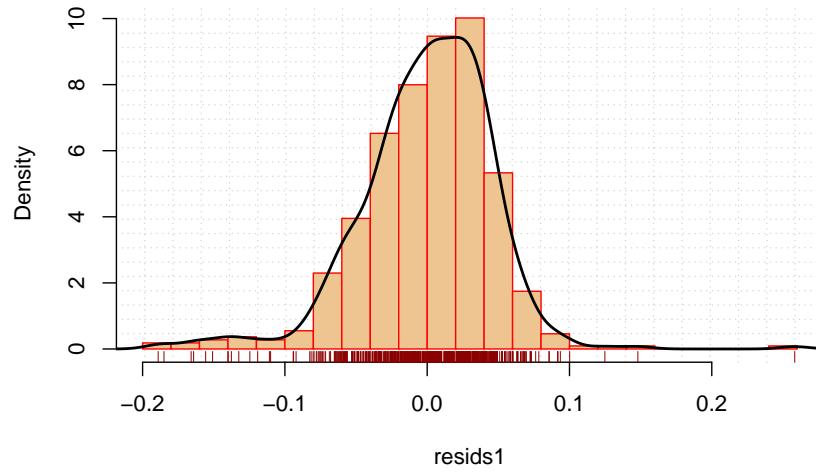
```

## final value -3.087522
## converged
## initial value -3.087448
## iter 2 value -3.087448
## iter 3 value -3.087449
## iter 3 value -3.087449
## iter 3 value -3.087449
## final value -3.087449
## converged

```



Plot of the residuals for ARMA(0,3)



The Ljung-Box p-value is significant until lag 7 and from the Q-Q plot some sample residuals are not in the line of the theoretical ones as we see on the tail of the plot. The histogram of the residuals seems quit normal although the tails are very long.

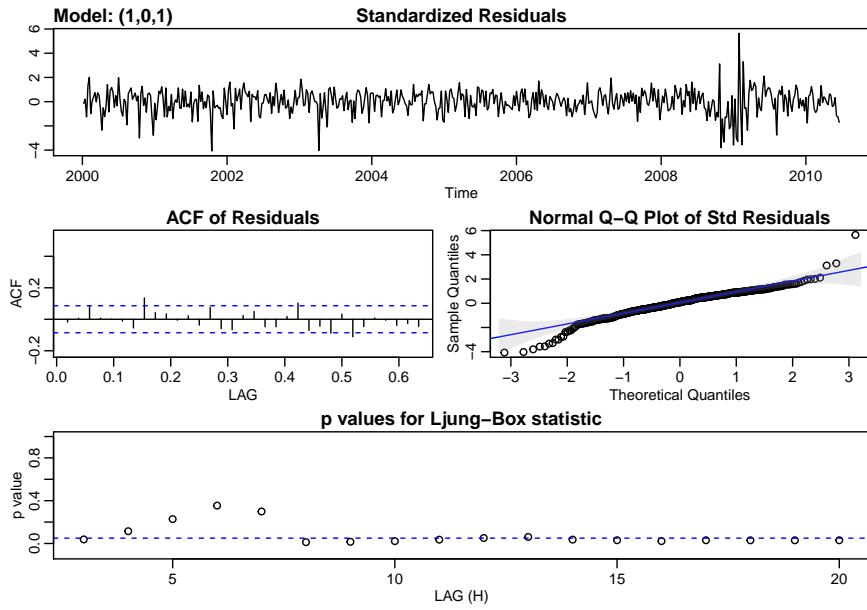
Next we perform runs test for independence

	x
pvalue	0.7940
observed.runs	267.0000
expected.runs	270.5147
n1	246.0000
n2	298.0000
k	0.0000
The p-value is quite high suggesting that the series is dependent	

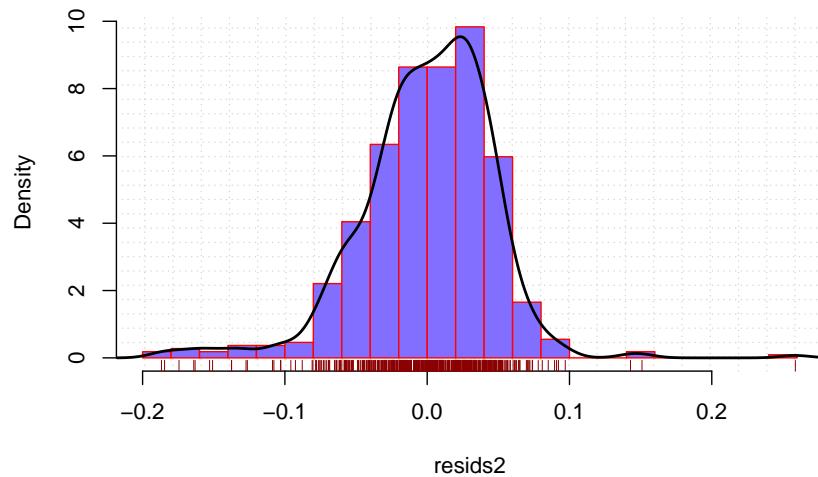
ARMA(1,1)

We proceed now for the next model

```
## initial value -3.057594
## iter  2 value -3.061420
## iter  3 value -3.067360
## iter  4 value -3.067479
## iter  5 value -3.071834
## iter  6 value -3.074359
## iter  7 value -3.074843
## iter  8 value -3.076656
## iter  9 value -3.080467
## iter 10 value -3.081546
## iter 11 value -3.081603
## iter 12 value -3.081615
## iter 13 value -3.081642
## iter 14 value -3.081643
## iter 14 value -3.081643
## iter 14 value -3.081643
## final value -3.081643
## converged
## initial value -3.082345
## iter  2 value -3.082345
## iter  3 value -3.082346
## iter  4 value -3.082346
## iter  5 value -3.082346
## iter  5 value -3.082346
## iter  5 value -3.082346
## final value -3.082346
## converged
```



Plot of the residuals for ARMA(1,1)



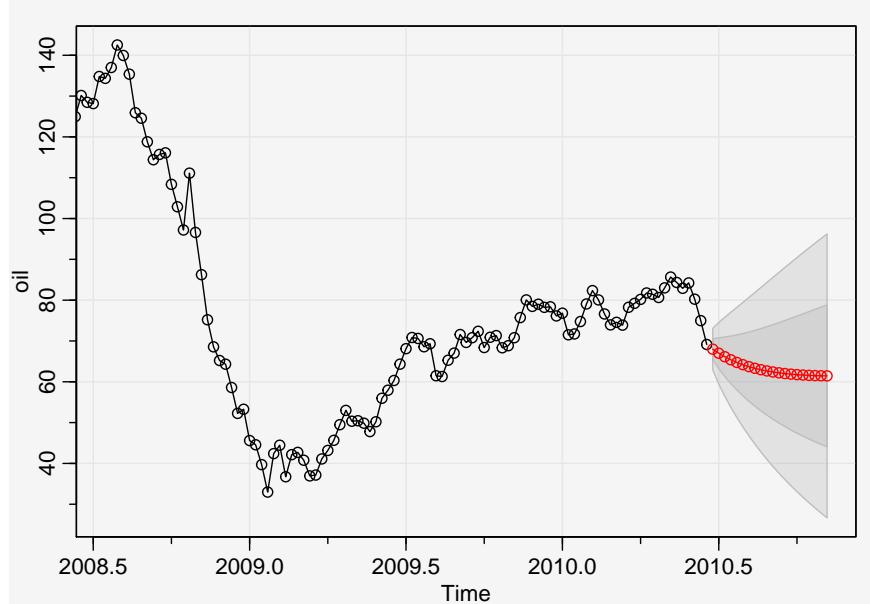
As we can see from the Q-Q plot ,the plot the Ljung-Box and the plot of the histogram of the residuals we obtain quite similar results with the previous model. Testing again for independence again the p-value suggests that we have dependence.

x	
pvalue	0.7880
observed.runs	275.0000
expected.runs	271.3787
n1	251.0000
n2	293.0000
k	0.0000

We proceed comparing the AIC and BIC of the 2 models.

	AIC	BIC
ARMA(0,3)	-3.318638	-3.279125
ARMA(1,1)	-3.312109	-3.280499

From the above table we conclude that the ARMA(1,1,1) seems to perform a little better and we use this for the predictions.



b)

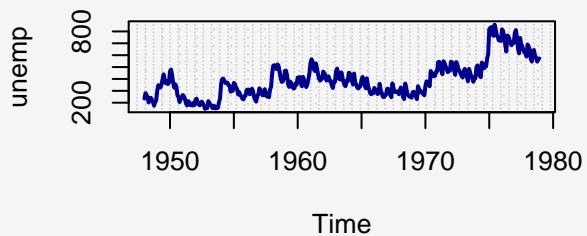
Find a suitable $ARIMA(p, d, q)x(P, D, Q)_s$ model for the data set unemp present in the library astsa. Your modeling should include the following steps in an appropriate order: visualization, detrending by differencing (if necessary), transformations (if necessary), ACF and PACF plots when needed, EACF analysis, Q-Q plots, Box-Ljung test, ARIMA fit analysis, control of the parameter redundancy in the fitted model. When performing these steps, always have 2 tentative models at hand and select one of them in the end. Validate your choice by AIC and BIC and write down the equation of the selected model (write in the backshift operator notation without expanding the brackets). Finally, perform forecasting of the model 20 observations ahead and provide a suitable plot showing the forecast and its uncertainty.

TS -ACF-PACF Plots

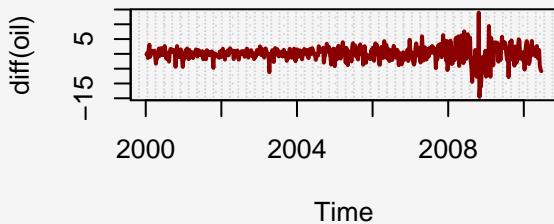
We start by making some diagnostic plots for the original time series ,the first difference and the log first difference.

TS Plots

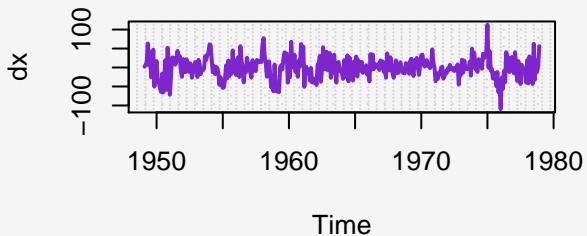
PLot of unemp TS



PLot of diff(unemp) TS

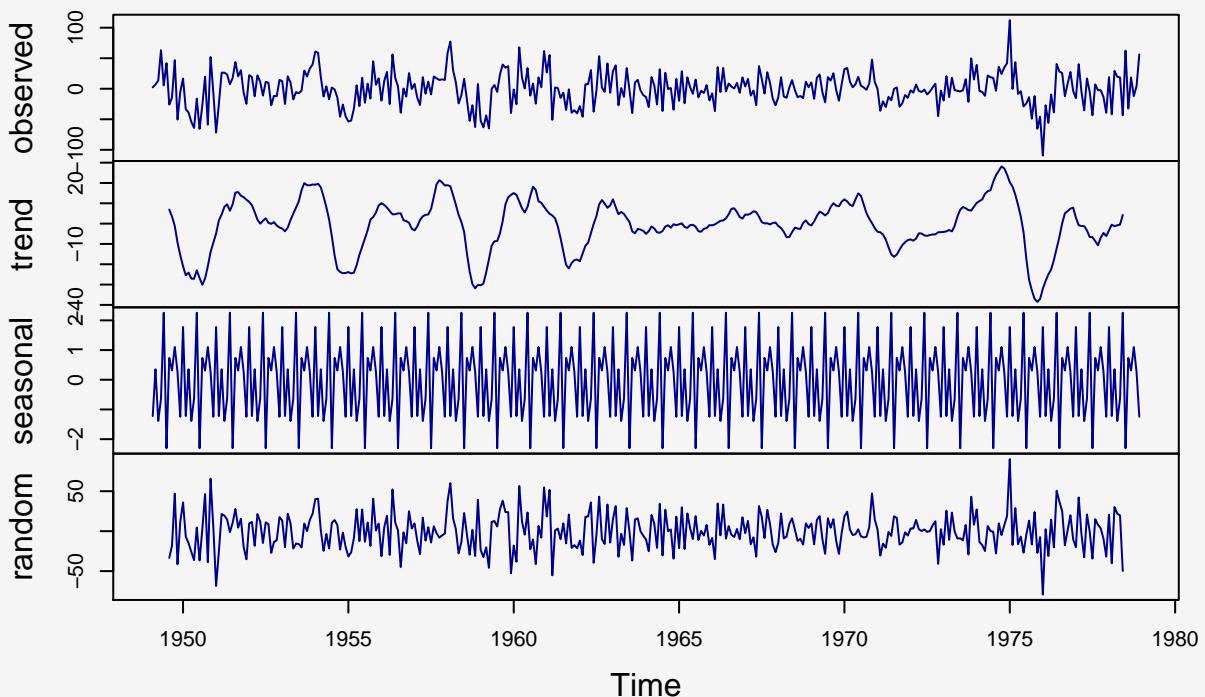


PLot of diff(diff(unemp)),12) TS



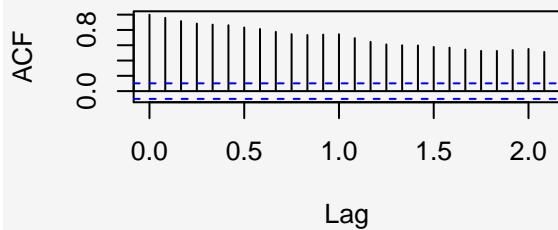
We report also the decomposition of the $\nabla^{12} \nabla unemp$

Decomposition of additive time series

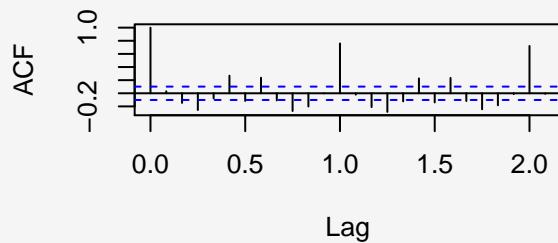


ACF Plots

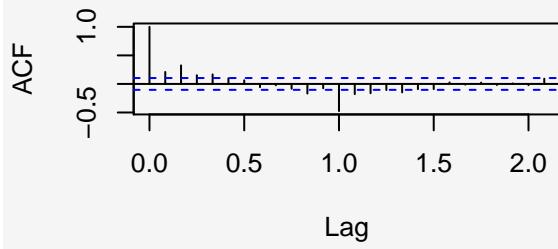
Series unemp

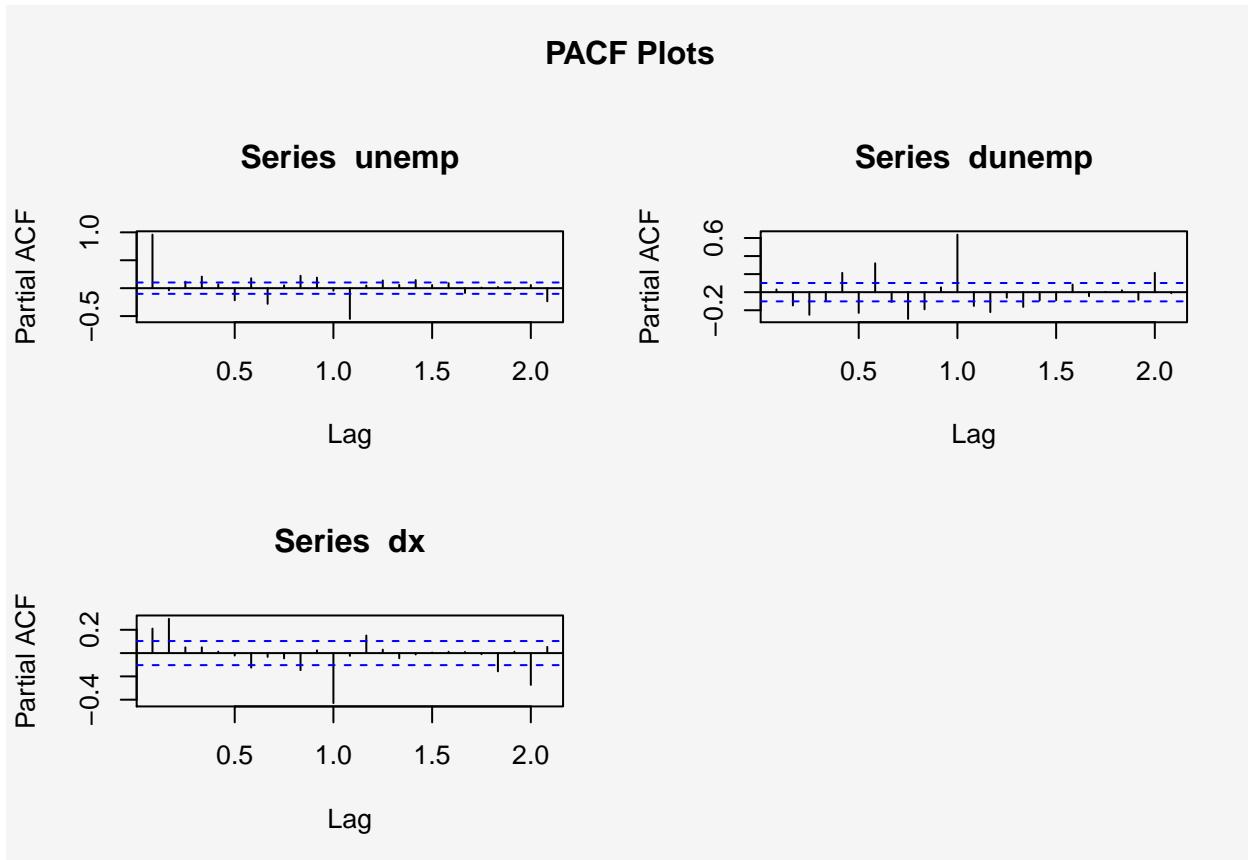


Series dunemp



Series dx





From the plots we conclude that is suggesting to work with the $\nabla^{12}\nabla unemp$ transformation. Also the plot suggest an $ARMA(1, 1, 1)$ for the first difference data. Here we report the Dickey-Fuller test for stationarity and as we can see the p-value suggests that data are non stationary.

```
##  
## Augmented Dickey-Fuller Test  
##  
## data: dx  
## Dickey-Fuller = -6.171, Lag order = 7, p-value = 0.01  
## alternative hypothesis: stationary
```

Now we are going to use the eacf in order to identify best model combinations.

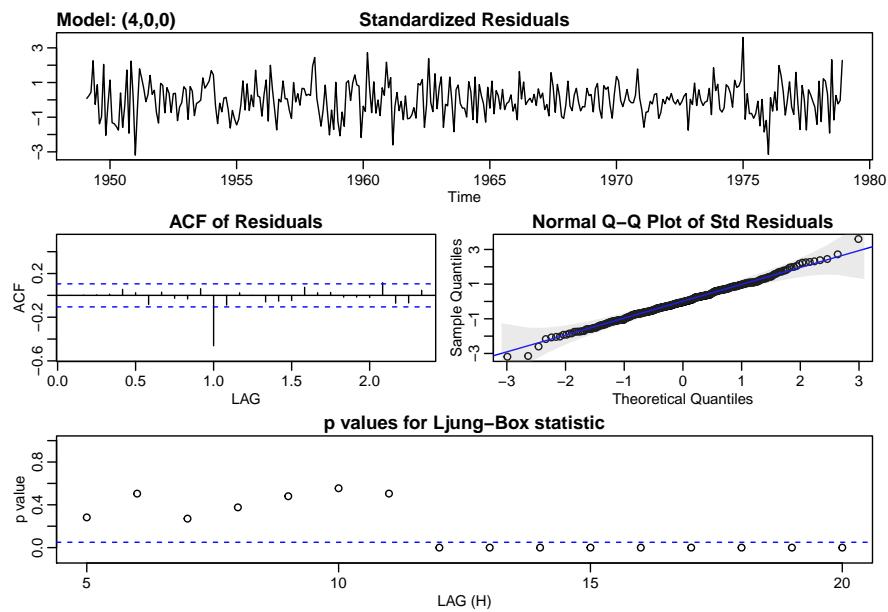
```
## AR/MA  
## 0 1 2 3 4 5 6 7 8 9 10 11 12 13  
## 0 x x x x o o o o o x o x x x  
## 1 x x x o o o o o o o x o x x o  
## 2 x x o o o o o o o o o x o x  
## 3 x x o o o o o o o o o x o x  
## 4 x x o o o o o o o o o x o o  
## 5 x o x o x o o o o o x x x o  
## 6 x x x x x o o o o o x o o  
## 7 x x x x x o o o o o x o o
```

From the matrix we distinguish 2 models 1.ARMA(4,0) and ARMA(2,2). We are going to investigate each separately

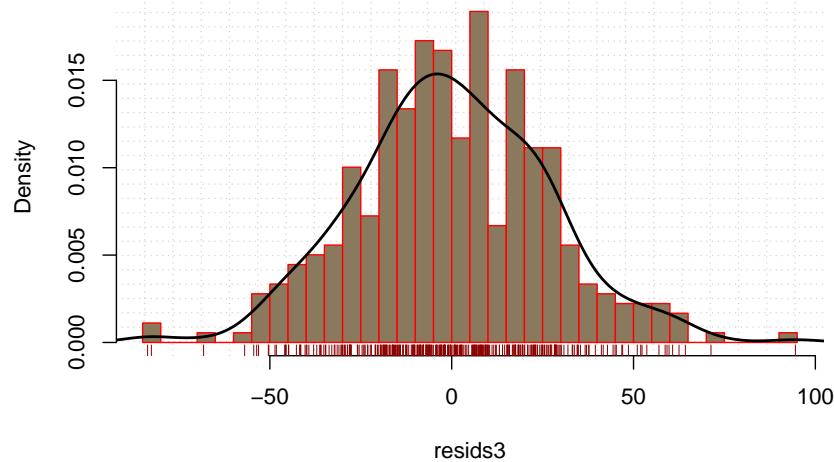
ARMA(4,0)

Diagnostic Plots for residuals

```
## initial value 3.336140
## iter 2 value 3.298515
## iter 3 value 3.265469
## iter 4 value 3.265171
## iter 5 value 3.265162
## iter 6 value 3.265161
## iter 7 value 3.265157
## iter 8 value 3.265157
## iter 9 value 3.265156
## iter 9 value 3.265156
## iter 9 value 3.265156
## final value 3.265156
## converged
## initial value 3.267503
## iter 2 value 3.267493
## iter 3 value 3.267473
## iter 4 value 3.267464
## iter 5 value 3.267456
## iter 6 value 3.267454
## iter 7 value 3.267454
## iter 7 value 3.267454
## iter 7 value 3.267454
## final value 3.267454
## converged
```



Plot of the residuals for ARMA(4,0)



The Ljung-Box p-value is significant until lag 11 and from the Q-Q plot some sample residuals are not in the line of the theoretical ones as we see on the tail of the plot. The histogram of the residuals seems quit normal.

Next we perform runs test for independence

	x
pvalue	0.9800
observed.runs	181.0000
expected.runs	180.2646
n1	186.0000
n2	173.0000
k	0.0000

ARMA(2,2)

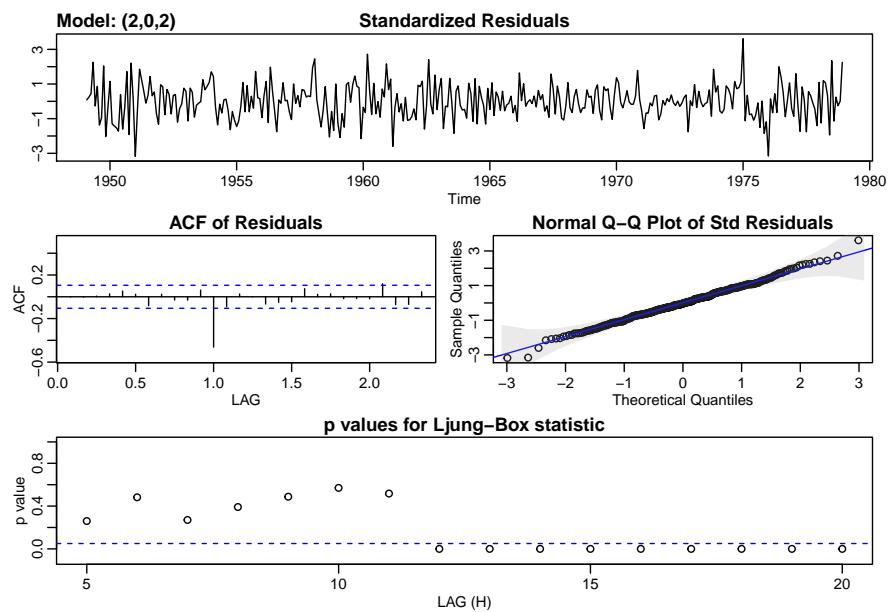
Diagnostic Plots for residuals

```
## initial value 3.340771
## iter  2 value 3.295757
## iter  3 value 3.279401
## iter  4 value 3.276527
## iter  5 value 3.275399
## iter  6 value 3.271435
## iter  7 value 3.270544
## iter  8 value 3.270026
## iter  9 value 3.269973
## iter 10 value 3.269972
## iter 11 value 3.269972
## iter 12 value 3.269971
## iter 12 value 3.269971
## final  value 3.269971
## converged
```

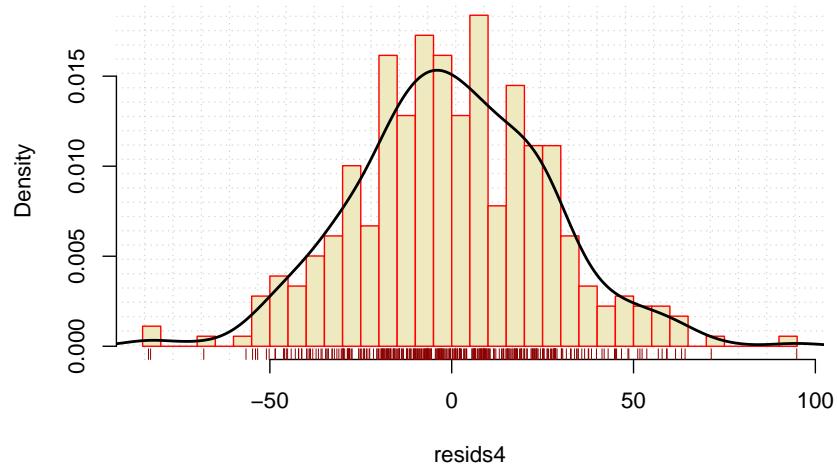
```

## initial value 3.267802
## iter 2 value 3.267800
## iter 3 value 3.267796
## iter 4 value 3.267792
## iter 5 value 3.267791
## iter 6 value 3.267791
## iter 7 value 3.267791
## iter 8 value 3.267791
## iter 9 value 3.267791
## iter 10 value 3.267791
## iter 11 value 3.267791
## iter 12 value 3.267791
## iter 12 value 3.267791
## final value 3.267791
## converged

```



Plot of the residuals for ARMA(0,3)



The results are similar with the previous model

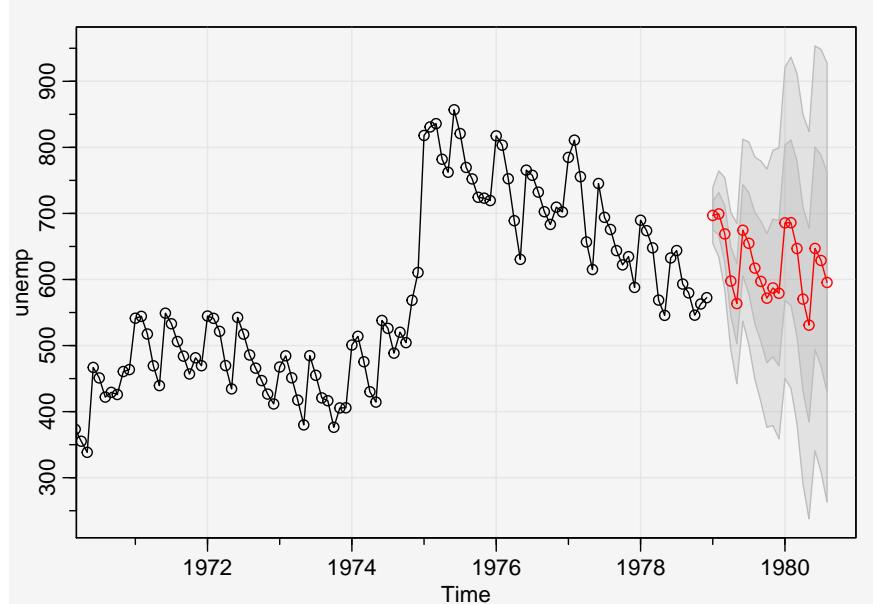
Next we perform runs test for independence

	x
pvalue	0.9900
observed.runs	181.0000
expected.runs	180.3872
n1	184.0000
n2	175.0000
k	0.0000

We proceed comparing the AIC and BIC of the 2 models.

	AIC	BIC
ARMA(4,0)	9.406211	9.471113
ARMA(2,2)	9.406885	9.471787

From the above table we conclude that the $SARMA(4, 1, 0)_{12}$ seems to perform a little better and we use this for the predictions. The final model is $SARMA(1, 1, 1)x(4, 1, 0)_{12}$ and we use this for the prediction shown below.



Appendix

```
1 ## ----setup,
2 ## include=FALSE-----
3 ## knitr::opts_chunk$set(echo = F, message = F,warning =
4 ## F)
5 # Libraries
# -----
6 library(astsa)
7 library(ggplot2)
8 library(knitr)
# -----
9
10 # -----
11 # Assignment 1
# -----
12 # a) -----
13 set.seed(12345)
14 # sim AR(3) n=1000
15 AR3.sim = arima.sim(list(ar = c(0.8, -0.2, 0.1)), n = 1000)
16 # bind ts for up to 3 lags
17 data1 = ts.intersect(x = AR3.sim, x1 = lag(AR3.sim, -1),
18 x2 = lag(AR3.sim, -2), x3 = lag(AR3.sim, -3), dframe = T)
19 # linear regressions
20 res1 = lm(x ~ x1 + x2, data = data1)
21 res2 = lm(x3 ~ x1 + x2, data = data1)
22 # calculate residuals
23 resids1 = residuals(res1)
24 resids2 = residuals(res2)
25 # calculate correlation for the residuals
26 estimatedCorr = cor(cbind(resids1, resids2))
27 # theoretical pacf
28 theoreticalPacf = ARMAacf(ar = c(0.8, -0.2, 0.1), pacf = T)
29 simulatedPacf = pacf(AR3.sim)
30 # make a table with the corr and pacf
31 sum_tab = cbind(estimatedCorr[1, 2], simulatedPacf[3][[1]],
32 theoreticalPacf[3])
33 colnames(sum_tab) = c("Est.Corr", "Sim.PACF", "Theo.PACF")
34 ## -----
35 kable(sum_tab)
# -----
36 # -----
37 ## ----- b)
38 ## ----- simulate AR(2)
39 ## n=100
40 AR2.sim = arima.sim(list(ar = c(0.8, 0.1)), n = 100)
41 # perform each method
42 AR_yw = ar(AR2.sim, order.max = 2, aic = F) # Yule-Walker
43 AR_ols = ar(AR2.sim, method = "ols", order.max = 2, aic = F) # OLS
44 AR_mle = ar(AR2.sim, method = "mle", order.max = 2, aic = F) # ML
45 # make a table with the coefficients
46 d = rbind(AR_yw$ar, AR_ols$ar, AR_mle$ar, c(0.8, 0.1))
47 rownames(d) = c("YW", "OLS", "MLE", "TRUE")
48 colnames(d) = c("phi1", "phi2")
kable(d)
## ----- check if phi2 in CI
```

```

52 ## for ML estimate compute the CI
53 lower_CI = AR_mle$ar[2] - sqrt(AR_mle$asy.var.coef[2, 2]) *
54   1.96 # lower limit
55 upper_CI = AR_mle$ar[2] + sqrt(AR_mle$asy.var.coef[2, 2]) *
56   1.96 # upper limit
57 # check if the phi2=0.1 in the CI
58 cat("The esitmated interval is :", c(lower_CI, upper_CI),
59     "\n")
60 cat("The value of phi_2 is within the estimated interval?\n")
61 0.1 %in% round(seq(as.numeric(lower_CI[1]), as.numeric(upper_CI[1]),
62   length.out = 1000), 2) # returns T,F
63 # PACF = ARMAacf(ar=c(0.8,0.1), pacf=TRUE) ; PACF #
64 # theoretical PACF
65 #
66 # c) -----
67 # ARIMA(0,0,1)x(0,0,1)12 n=200
68 seasonal.sim = arima.sim(list(order = c(0, 0, 13), ma = c(0.3,
69   rep(0, 10), 0.6, (0.6 * 0.3))), n = 200)
70 # plot of the simulated PACF and ACF for simulation
71 par(mfrow = c(2, 2), bg = "whitesmoke")
72 acf(seasonal.sim, main = "Sample ACF", panel.first = grid(25,
73   25), lag.max = 40)
74 pacf(seasonal.sim, main = "Sample PACF", panel.first = grid(25,
75   25), lag.max = 40)
76 # compute theoretical ACF and PACF
77 ACF = ARMAacf(ma = c(0.3, rep(0, 10), 0.6, (0.6 * 0.3)),
78   lag.max = 40)
79 PACF = ARMAacf(ma = c(0.3, rep(0, 10), 0.6, (0.6 * 0.3)),
80   pacf = TRUE, lag.max = 40)
81 # plot of the theoretical ACF and PACF
82 plot(ACF, type = "h", xlab = "Lag", ylim = c(-0.4, 0.8),
83   main = "Theoretical ACF", panel.first = grid(25, 25))
84 abline(h = 0)
85 plot(PACF, type = "h", xlab = "Lag", ylim = c(-0.4, 0.8),
86   main = "Theoretical PACF", panel.first = grid(25, 25))
87 abline(h = 0)
88 #
89 # d) -----
90 # using simulation -----
91 # simulate ARIMA(0,0,1)x(0,0,1)12
92 seasonal.sim1 = arima.sim(list(order = c(0, 0, 13), ma = c(0.3,
93   rep(0, 10), 0.6, (0.6 * 0.3))), n = 200)
94 # forecast using the sarima.for and auto plot
95 # fore.sar=sarima.for(seasonal.sim1,n.ahead=30,p=0,d=0,q=1,P=0,D=0,Q=1,S=12)
96 # the same results but using predict forecast using the
97 # predict and plot of the forecasts
98 fore = predict(arima(seasonal.sim1, order = c(0, 0, 1),
99   seasonal = list(order = c(0, 0, 1), period = 12)), n.ahead = 30)
100 # plot of ts and predictions and band
101 cols = c("mediumpurple3", "darkslateblue")
102 par(mfrow = c(1, 1), oma = c(0, 0, 3, 0))
103 ts.plot(seasonal.sim1, fore$pred, col = cols, lwd = 3)
104 U = fore$pred + fore$se

```

```

105 L = fore$pred - fore$se
106 xx = c(time(U), rev(time(U)))
107 yy = c(L, rev(U))
108 polygon(xx, yy, border = 8, col = gray(0.6, alpha = 0.2))
109 points(fore$pred, pch = 20, col = "red")
110 legend("topright", legend = c("Sim TS", "Predicted 30"),
111       col = c(cols[1], cols[2]), lty = 1, lwd = 2)
112 title(expression("ARMA(0,0,1)x(0,0,1)"[12]))
113 # using the kernelab ----- kernel fit not import
114 # the kernelab because of the predict
115 kernel_fit = kernlab::gausspr(x = c(1:200), seasonal.sim1)
116 # make predictions
117 kernels_preds = kernlab::predict(kernel_fit, c(200:230))
118 # plot the ts and predictions
119 cols1 = c("blue2", "black")
120 par(mfrow = c(1, 1), oma = c(0, 0, 3, 0))
121 ts.plot(seasonal.sim1, ts(kernels_preds, start = 200, end = 230),
122         col = cols1, lwd = 3)
123 points(ts(kernels_preds, start = 200, end = 230), pch = 20,
124         col = "azure3", cex = 0.2)
125 legend("bottomright", legend = c("Sim TS", "Predicted 30"),
126       col = c(cols1[1], cols[2]), lty = 1, lwd = 2)
127 title(expression("ARMA(0,0,1)x(0,0,1)"[12] * " kernelab"))
128 #
129 # e) ----- simulate
130 # ARMA(1,1) n=50
131 arma.sim <- arima.sim(list(order = c(1, 0, 1), ar = 0.7,
132                           ma = 0.5), n = 50)
133 # make predictions with the sarima.for
134 # sarima.for(arma.sim[1:40], n.ahead=10, 1, 0, 1, 0, 0, 0, 0, no.constant
135 # = T)
136 # the same results but using predict forecast using the
137 # predict and plot of the forecasts
138 fore1 = predict(arima(arma.sim[1:40], order = c(1, 0, 1),
139                      include.mean = F), n.ahead = 10)
140 col1 = "forestgreen"
141 col2 = "magenta1"
142 col3 = "orange1"
143 ts.plot(as.ts(arma.sim[1:41]), fore1$pred, col = c(col1,
144           col2), lwd = 2, main = "ARMA(1,1) with predictions")
145 lines(ts(arma.sim[41:50], start = 41, end = 50), col = col3,
146       lwd = 2, type = "o")
147 U1 = fore1$pred + fore1$se
148 L1 = fore1$pred - fore1$se
149 xx1 = c(time(U1), rev(time(U1)))
150 yy1 = c(L1, rev(U1))
151 polygon(xx1, yy1, border = 8, col = gray(0.6, alpha = 0.2))
152 points(fore1$pred, pch = 20, col = "black", cex = 0.5)
153 legend("topleft", legend = c("First 40 values", "Predicted 10",
154       "True 10"), col = c(col1, col2, col3), lty = 1, lwd = 2)
155 ## ----- Note
156 require(dplyr) # we load the library here becaaus it masks the lag and had problems in Ass1.a
157 # count how many points are not in the band

```

```

158 mat <- cbind(as.vector(U1), as.vector(L1), as.vector(arma.sim[41:50]))
159 mat <- as.data.frame(mat) # ; mat
160 mat = mat %>% mutate(res = ifelse(((mat$V3 > mat$V1) | (mat$V3 <
161     mat$V2)), 1, 0))
162 cat("The number of the values outside the prediction band is:",
163     sum(mat$res))
164 #
# -----
165 # Assignment 2
# -----
166 #
# -----
167 # data chicken -----
168 data(chicken)
169 # girst diffrence
170 diff.xt <- diff(chicken)
171 my_colors = c("seagreen4", "red4")
172 # ACF and PACF plots
173 par(mfrow = c(2, 2), bg = "whitesmoke", oma = c(0, 0, 3,
174     0))
175 acf(chicken, 60, col = my_colors[1], main = expression("ACF x"[t]),
176     lwd = 2)
177 pacf(chicken, 60, col = my_colors[1], main = expression("PACFx"[t]),
178     lwd = 2)
179 acf(diff.xt, 60, col = my_colors[2], main = expression("ACF diff x"[t]),
180     lwd = 2)
181 pacf(diff.xt, 60, col = my_colors[2], main = expression("PACF diff x"[t]),
182     lwd = 2)
183 title("\nACF and PACF Plots for chicken \nwith first diffrences",
184     outer = TRUE)
185 # mtext(c('ACF~PACF Plots chicken', 'ACF~PACF PLots for
186 # diff(chicken)'), side = 3, line = c(-2,-18), outer =
187 # TRUE)
188 #
# -----
189 # data so2 -----
190 data(so2)
191 # first diffrence
192 diff.xt <- diff(so2)
193 # ACF and PACF plots
194 par(mfrow = c(2, 2), bg = "whitesmoke", oma = c(0, 0, 3,
195     0))
196 acf(chicken, 40, col = my_colors[1], main = expression("ACF x"[t]),
197     lwd = 2)
198 pacf(chicken, 40, col = my_colors[1], main = expression("PACFx"[t]),
199     lwd = 2)
200 acf(diff.xt, 40, col = my_colors[2], main = expression("ACF diff x"[t]),
201     lwd = 2)
202 pacf(diff.xt, 40, col = my_colors[2], main = expression("PACF diff x"[t]),
203     lwd = 2)
204 title("\nACF and PACF Plots for so2 \nwith first diffrence",
205     outer = TRUE)
206 # mtext(c('ACF~PACF Plots so2', 'ACF~PACF PLots for
207 # diff(so2)'), side = 3, line=c(0,-19),outer = T)
208 #
# -----
209 # data EQcount -----
210 data("EQcount")

```

```

211 # first diffrence
212 diff.xt <- diff(EQcount)
213 # plots
214 par(mfrow = c(2, 2), bg = "whitesmoke", oma = c(0, 0, 3,
215   0))
216 acf(chicken, 40, col = my_colors[1], main = expression("ACF x"[t]),
217   lwd = 2)
218 pacf(chicken, 40, col = my_colors[1], main = expression("PACFx"[t]),
219   lwd = 2)
220 acf(diff.xt, 40, col = my_colors[2], main = expression("ACF diff x"[t]),
221   lwd = 2)
222 pacf(diff.xt, 40, col = my_colors[2], main = expression("PACF diff x"[t]),
223   lwd = 2)
224 title("\nACF and PACF Plots for EQcount \nwith first diffrence",
225   outer = TRUE)
226 # mtext(c('ACF~PACF Plots EQcount', 'ACF~PACF Plots for
227 # diff(EQcount)'), side = 3, line = c(-2,-18), outer =
228 # TRUE)
229 #
230 # HCT -----
231 data(HCT)
232 # first diffrence
233 diff.xt <- diff(HCT)
234 # plots
235 par(mfrow = c(2, 2), bg = "whitesmoke", oma = c(0, 0, 3,
236   0))
237 acf(chicken, 40, col = my_colors[1], main = expression("ACF x"[t]),
238   lwd = 2)
239 pacf(chicken, 40, col = my_colors[1], main = expression("PACFx"[t]),
240   lwd = 2)
241 acf(diff.xt, 40, col = my_colors[2], main = expression("ACF diff x"[t]),
242   lwd = 2)
243 pacf(diff.xt, 40, col = my_colors[2], main = expression("PACF diff x"[t]),
244   lwd = 2)
245 title("\nACF and PACF Plots for HTC \nwith first diffrence",
246   outer = TRUE)
247 # mtext(c('ACF~PACF Plots EQcount', 'ACF~PACF Plots for
248 # diff(EQcount)'), side = 3, line = c(-2,-35), outer =
249 # TRUE)
250 #
251 # Assignment 3
252 #
253 # a) -----
254 data(oil)
255 # plots
256 par(mfrow = c(2, 2), oma = c(0, 0, 3, 0), bg = "whitesmoke")
257 plot.ts(oil, col = "darkblue", lwd = 2, panel.first = grid(25,
258   25), main = "PLot of oil TS")
259 plot.ts(diff(oil), col = "darkred", lwd = 2, panel.first = grid(25,
260   25), main = "PLot of diff(oil) TS")
261 plot.ts(diff(log(oil)), col = "purple3", lwd = 2, panel.first = grid(25,
262   25), main = "PLot of diff(log(oil)) TS")
263 title("TS Plots", outer = T)

```

```

264  ## -----
265  par(mfrow = c(2, 2), bg = "whitesmoke", oma = c(0, 0, 3,
266    0))
267  acf(oil)
268  acf(diff(oil))
269  acf(diff(log(oil)))
270  title("ACF Plots", outer = T)
271  ## -----
272  par(mfrow = c(2, 2), bg = "whitesmoke", oma = c(0, 0, 3,
273    0))
274  pacf(oil)
275  pacf(diff(oil))
276  pacf(diff(log(oil)))
277  title("PACF Plots", outer = T)
278  ## ----- we work
279  ## with diff(log(oil)) for the analysis
280  doil = diff(log(oil))
281  # test the p-value
282  tseries::adf.test(doil)
283  plots = acf2(doil)
284  ## -----
285  ## eacf test
286  TSA::eacf(doil) # 2 choices AR(0,3) ARMA(1,1)
287  # Start with ARMA(0,3) -----
288  # fit the model
289  arma.fit = sarima(doil, p = 0, d = 0, q = 3, details = T)
290  # tsdiag(arma.fit$fit) diagnostic plots
291  resids1 = residuals(arma.fit$fit) # calculate residuals
292  # plot the residuals with the forecast package
293  # resids1%>%forecast::ggtsdisplay(main='TS-ACF-PACF for
294  # residuals', col=sample(colors(),1),theme=theme_gray())
295  # forecast::gg histogram(resids1,add.normal = T,add.kde =
296  # T) # plot of residuals with gg histogram plot the
297  # histogram of the residuals with basic
298  hist(resids1, 30, col = sample(colors(), 1), border = "red",
299    panel.first = grid(25, 25), freq = F, main = "Plot of the residuals for ARMA(0,3)")
300  lines(density(resids1), lwd = 2)
301  rug(resids1, col = "red4")
302  ## ----- Test the
303  ## independence of a sequence of random variables
304  kable(unlist(TSA::runs(resids1)))
305  # Then with ARMA(1,1) ----- fit model
306  arma.fit1 = sarima(doil, 1, 0, 1)
307  resids2 = residuals(arma.fit1$fit)
308  # plot of residuals with the forecast package
309  # resids2%>%forecast::ggtsddisplay(main='TS-ACF-PACF for
310  # residuals', col=sample(colors(),1),theme=theme_gray())
311  # plot of the histogram with the basic
312  hist(resids2, 30, col = sample(colors(), 1), border = "red",
313    panel.first = grid(25, 25), freq = F, main = "Plot of the residuals for ARMA(1,1)")
314  lines(density(resids2), lwd = 2)
315  rug(resids2, col = "red4")
316  ## -----

```

```

317 kable(unlist(TSA::runs(resids2)))
318 ## ----- check the AIC
319 ## and BIC for each model
320 AIC_BIC_mat = cbind(c(arma.fit$AIC, arma.fit1$AIC), c(arma.fit$BIC,
321   arma.fit1$BIC)) # ; AIC_mat
322 colnames(AIC_BIC_mat) = c("AIC", "BIC")
323 rownames(AIC_BIC_mat) = c("ARMA(0,3)", "ARMA(1,1)")
324 kable(AIC_BIC_mat)
325 # we choose the ARMA(1,1) and make predictions
326 par(mfrow = c(1, 1), bg = "whitesmoke")
327 S1 <- sarima.for(oil, n.ahead = 20, p = 1, d = 1, q = 1,
328   P = 0, D = 0, Q = 0, S = 0)
329 # the same results but using predict
330 # fore2=predict(arima(oil,order=c(1,0,1),include.mean =
331 # F),n.ahead = 20)
332 # cols4=sample(colors(),2)
333 # ts.plot(oil,fore2$pred,col=cols4,
334 # lwd=3,main='ARMA(1,1) with predictions') U2 =
335 # fore2$pred+fore2$se; L2 = fore2$pred-fore2$se xx2 =
336 # c(time(U2), rev(time(U2))); yy2 = c(L2, rev(U2))
337 # polygon(xx2, yy2, border = 8, col = gray(.6, alpha =
338 # .2)) points(fore2$pred, pch=19, col=2,cex=0.3)
339 # legend('topleft',legend=c('Oil TS','Predicted Oil 20
340 # ahead'), col=cols1,lty=1,lwd=2)
341 # b) -----
342 ##
343 data(unemp)
344 # first difference
345 dunemp = diff(unemp)
346 # remove seasonal by 12 difference
347 dx = diff(dunemp, 12)
348 # plots
349 par(mfrow = c(2, 2), oma = c(0, 0, 3, 0), bg = "whitesmoke")
350 plot.ts(unemp, col = "darkblue", lwd = 2, panel.first = grid(25,
351   25), main = "PLot of unemp TS")
352 plot.ts(diff(oil), col = "darkred", lwd = 2, panel.first = grid(25,
353   25), main = "PLot of diff(unemp) TS")
354 plot.ts(dx, col = "purple3", lwd = 2, panel.first = grid(25,
355   25), main = "PLot of diff(diff(unemp),12) TS")
356 title("TS Plots ", outer = T)
357 ##
358 par(mfrow = c(1, 1), bg = "whitesmoke")
359 plot(decompose(dx), col = "darkblue")
360 ##
361 par(mfrow = c(2, 2), bg = "whitesmoke", oma = c(0, 0, 3,
362   0))
363 acf(unemp)
364 acf(dunemp)
365 acf(dx)
366 title("ACF Plots", outer = T)
367 ##
368 par(mfrow = c(2, 2), bg = "whitesmoke", oma = c(0, 0, 3,
369   0))

```

```

370 pacf(unemp)
371 pacf(dunemp)
372 pacf(dx)
373 title("PACF Plots", outer = T)
374 ## ----- we work with
375 ## dx for the analysis test the p-value
376 tseries::adf.test(dx)
377 ## ----- eacf test
378 TSA::eacf(dx) # 2 choices AR(4,0) ARMA(2,2)
379 # Start with ARMA(4,0) -----
380 # fit the model
381 arma.fit3 = sarima(dx, p = 4, d = 0, q = 0, details = T)
382 # tsdiag(arma.fit$fit) diagnostic plots
383 resids3 = residuals(arma.fit3$fit) # calculate residuals
384 # plot the residuals with the forecast package
385 # resids1%>%forecast::ggtsdisplay(main='TS-ACF-PACF for
386 # residuals', col=sample(colors(),1),theme=theme_gray())
387 # forecast::gghistogram(resids1,add.normal = T,add.kde =
388 # T) # plot of residuals with gghistogram plot the
389 # histogram of the residuals with basic
390 hist(resids3, 30, col = sample(colors(), 1), border = "red",
391 panel.first = grid(25, 25), freq = F, main = "Plot of the residuals for ARMA(4,0)")
392 lines(density(resids3), lwd = 2)
393 rug(resids3, col = "red4")
394 ##
395 kable(unlist(TSA::runs(resids3)))
396 # Start with ARMA(0,3) -----
397 # fit the model
398 arma.fit4 = sarima(dx, p = 2, d = 0, q = 2, details = T)
399 # tsdiag(arma.fit$fit) diagnostic plots
400 resids4 = residuals(arma.fit4$fit) # calculate residuals
401 # plot the residuals with the forecast package
402 # resids1%>%forecast::ggtsdisplay(main='TS-ACF-PACF for
403 # residuals', col=sample(colors(),1),theme=theme_gray())
404 # forecast::gghistogram(resids1,add.normal = T,add.kde =
405 # T) # plot of residuals with gghistogram plot the
406 # histogram of the residuals with basic
407 hist(resids4, 30, col = sample(colors(), 1), border = "red",
408 panel.first = grid(25, 25), freq = F, main = "Plot of the residuals for ARMA(0,3)")
409 lines(density(resids4), lwd = 2)
410 rug(resids4, col = "red4")
411 ##
412 kable(unlist(TSA::runs(resids4)))
413 ## ----- check the AIC
414 ## and BIC for each model
415 AIC_BIC_mat1 = cbind(c(arma.fit3$AIC, arma.fit4$AIC), c(arma.fit3$BIC,
416 arma.fit4$BIC)) # ; AIC_mat
417 colnames(AIC_BIC_mat1) = c("AIC", "BIC")
418 rownames(AIC_BIC_mat1) = c("ARMA(4,0)", "ARMA(2,2)")
419 kable(AIC_BIC_mat1)
420 # we choose the ARMA(1,1) and make predictions
421 par(mfrow = c(1, 1), bg = "whitesmoke")
422 S2 <- sarima.for(unemp, n.ahead = 20, p = 1, d = 1, q = 1,

```

```

423     P = 4, D = 1, Q = 0, S = 12)
424 # the same results but using predict
425 # fore2=predict(arima(oil,order=c(1,0,1),include.mean =
426 # F),n.ahead = 20)
427 # cols4=sample(colors(),2)
428 # ts.plot(oil,fore2$pred,col=cols4,
429 # lwd=3,main='ARMA(1,1) with predictions') U2 =
430 # fore2$pred+fore2$se; L2 = fore2$pred-fore2$se xx2 =
431 # c(time(U2), rev(time(U2))); yy2 = c(L2, rev(U2))
432 # polygon(xx2, yy2, border = 8, col = gray(.6, alpha =
433 # .2)) points(fore2$pred, pch=19, col=2,cex=0.3)
434 # legend('topleft',legend=c('Oil TS','Predicted Oil 20
435 # ahead'), col=cols1,lty=1,lwd=2)
436 ## ----ref.label=knitr::all_labels(), echo = T, eval =
437 ## F-----

```

732A62 Time Series Analysis

Computer Lab C

Andreas C Charitos (andch552), Ruben Muñoz (rubmu773)

2019-10-10

Contents

1 Instructions	2
2 Implementation of Kalman filter	3
2.1 Assignment 1	3
2.1.1 a)	3
2.1.2 b)	3
2.1.3 c)	4
2.1.4 d)	5
2.1.5 e)	6
2.1.6 f)	7
3 Code Appendix	8

1 Instructions

- The lab is assumed to be done in groups.
- Create a report to the lab solutions in PDF.
- Be concise and do not include unnecessary printouts and figures produced by the software and not required in the assignments.
- **Include all your codes as an appendix into your report.**
- A typical lab report should 2-4 pages of text plus some number of figures plus appendix with codes.
- The group lab report should be submitted via LISAM before the deadline specified in LISAM.
- **Use 12345 as a random seed everywhere where the result of the simulation differs with the run unless stated otherwise.**

2 Implementation of Kalman filter

2.1 Assignment 1

In table 1 a script for generation of data from simulation of the following state space model and implementation of the Kalman filter on the data is given.

$$\begin{aligned} z_t &= A_{t-1}z_{t-1} + e_t, \quad e_t \sim \mathcal{N}(0, Q_t). \\ x_t &= C_t z_t + v_t, \quad v_t \sim \mathcal{N}(0, R_t) \end{aligned}$$

2.1.1 a)

2.1.1.1 Instructions.

Write down the expression for the state space model that is being simulated.

2.1.1.2 Results.

According to the simulation values we have $A_t = 1, C_t = 1, R_t = 1, Q_t = 1$ so the state space model is given by the following expression:

$$\begin{aligned} z_t &= z_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, 1) \\ x_t &= z_t + v_t, \quad v_t \sim \mathcal{N}(0, 1) \end{aligned}$$

2.1.2 b)

2.1.2.1 Instructions.

Run this script and compare the filtering results with a moving average smoother of order 5.

2.1.2.2 Results.

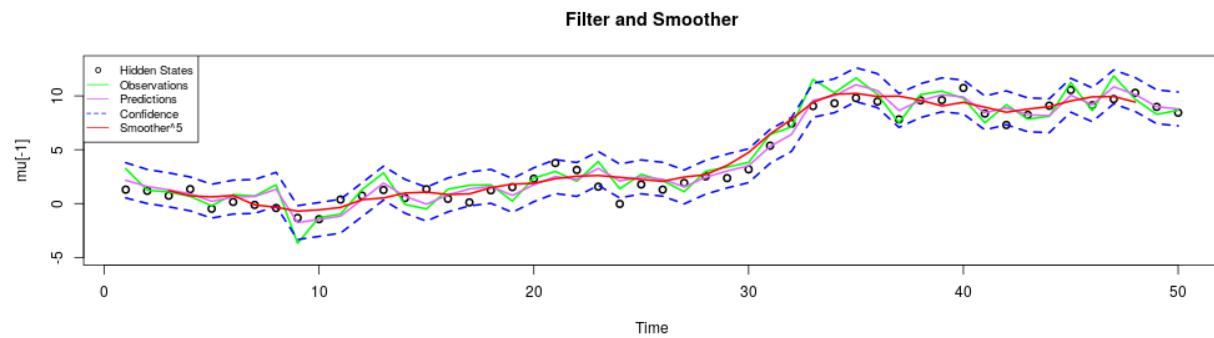


Figure 1: Visualization of the performance of the filter and a simple moving average.

As we can see from the plot the Kalman filter approximates the hidden states pattern quite well. The moving average smoother also manages to capture the overall trend of the signal quite well but as it was expected is much smoother than kalman filter.

2.1.3 c)

2.1.3.1 Instructions.

Also, compare the filtering outcome when R in the filter is 10 times smaller than its actual value while Q in the filter is 10 times larger than its actual value. How does the filtering outcome varies?

2.1.3.2 Results.

The Q and R act as “knobs” for the white noise that might be a inherit part of the model generating the states z_t and x_t respectively. When we in this artificial situation increase Q, this can be translated to the following statement “the error/variance” of the true model generating the states z_t is far from perfect and thus has a greater possibility of error when trying to explain the true hiden state.

As we can see if figure 2 when Q is increased, then the previous situation is also mirrored here, but in this case it means that the error of the observation is big. This causes in our plot for the variance in prediction lines to be very tight and thus the Karman filter model “greatly trusts” the prediction/observations, practically causing a very good filtering so as we can see in the plot the filter predictions and confidence intervals are the same as the observations.

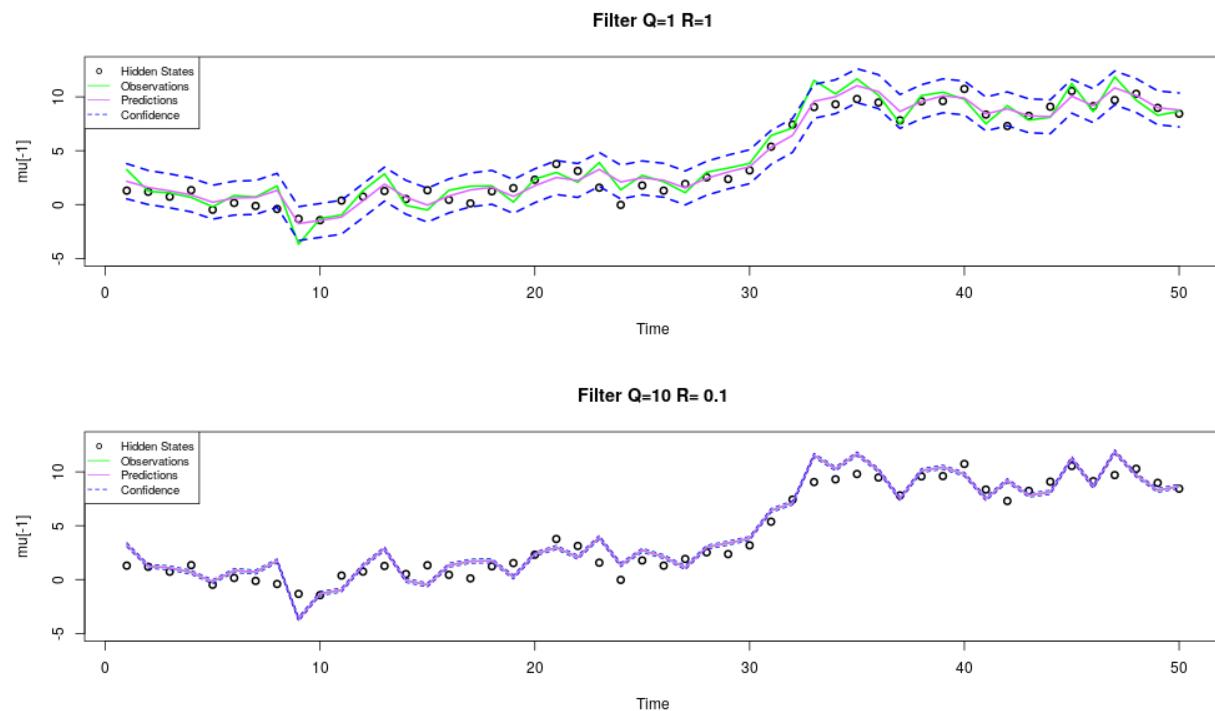


Figure 2: Visualization of the difference in R and Q ratio in the results.

2.1.4 d)

2.1.4.1 Instructions.

Now compare the filtering outcome when R in the filter is 10 times larger than its actual value while Q in the filter is 10 times smaller than its actual value. How does the filtering outcome varies?

2.1.4.2 Results.

As we can see in figure 3, the R is very large now and the Q very small, this is translated as, dont trust the observations because they have great error thus, thus we see that the filter is doing its job according on the parameters we have provided filtering out most of the noise that we told the observations had, but still following the general trend of the data. We can see that the filter predictions are very smooth and don't capture the states pattern.

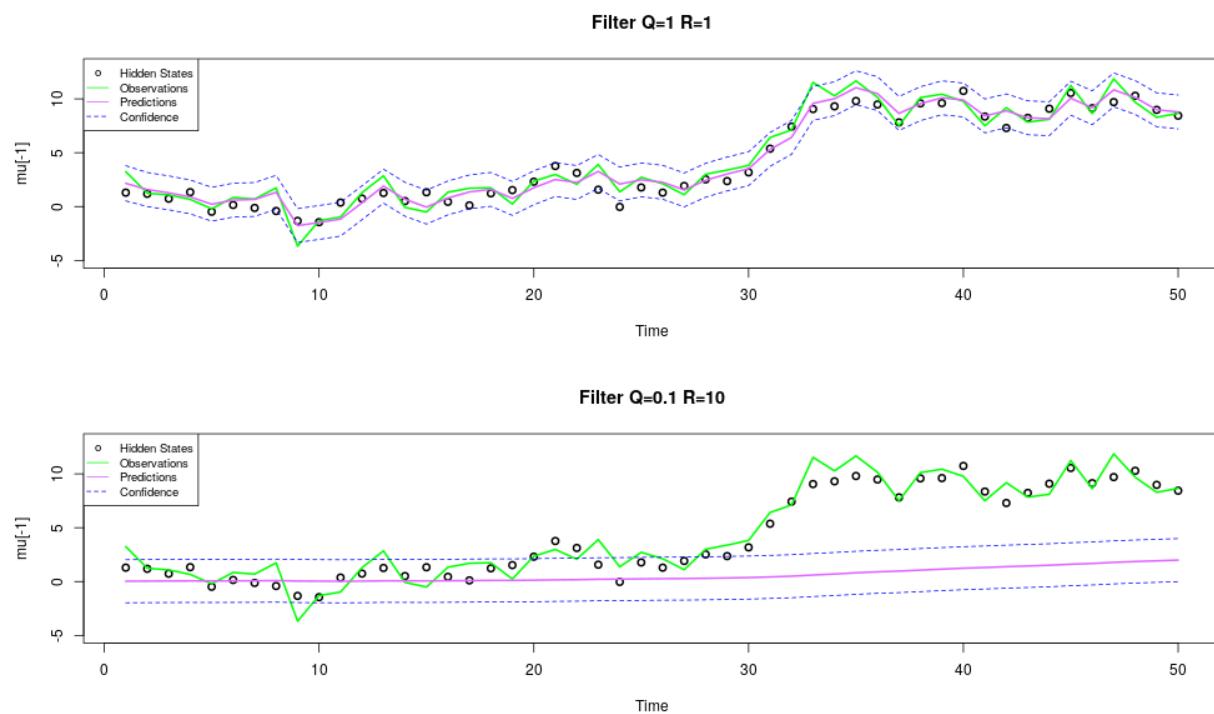


Figure 3: Visualization of the difference in R and Q ratio in the results.

To sum up, Q incorporates the noise between the nearby states, since it affects the change of states. Thus, all the changes among states are acceptable, so do the pattern of observations. The R incorporates the noise between state and observed point at each time.

- if Q is small, the estimated pattern of states will be change slightly.
- if Q is large, all change is acceptable, follow more the observations
- if R large, the confidence intervals of estimated states is large
- if R small, the confidence intervals of estimated staes is small

2.1.5 e)

2.1.5.1 Instructions.

Implement your own Kalman filter and replace ksmooth0 function with your script.

2.1.5.2 Results.

In the plots above we can see the results of our implementation compared with the Ksmooth function with the parameters for R,Q used in the previous questions. As we can see we were able to produce a somehow good approximation to Ksmooth function.

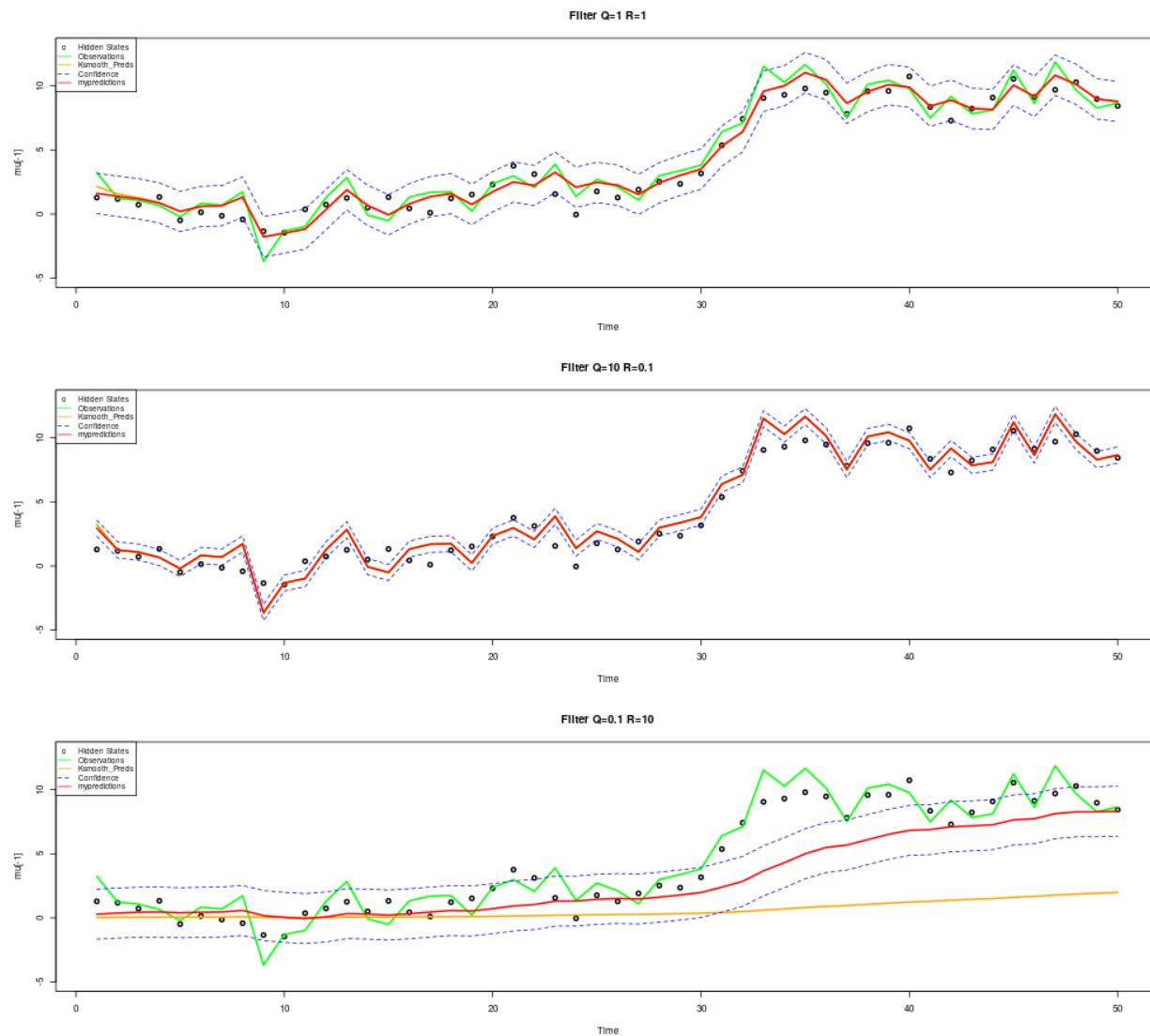


Figure 4: Visualization of the difference in R and Q ratio in the results with mykalmanfilter().

2.1.6 f)**2.1.6.1 Instructions.**

How do you interpret the Kalman gain?

2.1.6.2 Results.

The Kalman gain is the relative weight given to the measurements and current state estimate, and can be “tuned” to achieve a particular performance. With a high gain, the filter places more weight on the most recent measurements, and thus follows them more responsively. With a low gain, the filter follows the model predictions more closely. At the extremes, a high gain close to one will result in a more jumpy estimated trajectory, while a low gain close to zero will smooth out noise but decrease the responsiveness. When performing the actual calculations for the filter (as discussed below), the state estimate and covariances are coded into matrices to handle the multiple dimensions involved in a single set of calculations. This allows for a representation of linear relationships between different state variables (such as position, velocity, and acceleration) in any of the transition models or covariances. source Wikipedia [\[link\]](#)

3 3 Code Appendix

```

1 library(ggplot2)
2 # only use when knitting tables in to png
3 # library(kableExtra)
4 color_palette <- c(rgb(38,50,72,alpha=160, max = 255), # [1] MX robots blue
5                      rgb(255,152,0,alpha=160, max = 255), # [2] MX robots orange
6                      rgb(255,0,0,alpha=100, max = 255), # [3] Red faded
7                      rgb(0,0,0,alpha=160, max = 255), # [4] black
8                      rgb(10,150,10,alpha=160, max = 255), # [5] green
9                      rgb(0,0,0,alpha=40, max = 255), # [6] light Gray
10                     rgb(255,0,0,alpha=255, max = 255), # [7] Red full
11                     rgb(255,152,0,alpha=200, max = 255)) # [8] MX robots orange fulish
12 set.seed(12345)
13 Title <- "732A62 Time Series Analysis"
14 Subtitle <- "Computer Lab C"
15 Author <- "Andreas C Charitos (andch552), Ruben Muñoz (rubmu773)"
16 Date <- Sys.Date()
17 Chapter01 <- "Instructions"
18 Chapter02 <- "Implementation of Kalman filter"
19 ## b) -----
20 library(astsa)
21 ## script given for the lab -----
22 # generate data
23 set.seed(12345); num=50
24 w=rnorm(num + 1, 0, 1);
25 v=rnorm( num, 0, 1)
26 mu=cumsum(w) # state : mu [ 0 ] , mu [ 1 ] ,... , mu [ 5 0 ]
27 y = mu[-1] + v # obs : y [ 1 ] ,... , y [ 5 0 ]
28 Time = 1:num
29 # filter and smooth ( Ksmooth 0 does both )
30 ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 1, cR = 1)
31 # moving average smoother for comparison
32 mysmoother <- function(x, n){filter(as.vector(x),rep(1 / n, n), sides = 2)} # x=vector
  ↪ n=order
33 # start figure
34 png(filename="images/plotA1.png", width = 1000, height = 300)
35 par(mfrow = c( 1, 1)); Time = 1:num
36 # plot(Time, mu[-1], main = 'Predict', ylim = c(-5, 10))
37 # lines(Time ,y , col = 'green' )
38 # lines(ks$xp) # one-step-ahead prediction of the state
39 # lines(ks$xp + 2 * sqrt (ks$Pp), lty = 2, col = 4)
40 # lines(ks$xp - 2 * sqrt (ks$Pp), lty = 2, col = 4)
41 plot(Time, mu[-1], main = 'Filter and Smoother', ylim = c(-5, 13),lwd=2)
42 lines(Time, y, col='green',lwd=2)
43 lines(ks$xf,col="mediumorchid1",lwd=2)
44 lines(ks$xf + 2 * sqrt (ks$Pf) ,lty = 2, col = 4,lwd=2)
45 lines(ks $ xf - 2 * sqrt (ks$Pf) , lty = 2 , col = 4 ,lwd=2)
46 lines(Time, mysmoother(y,5), col='red',lwd=2)
47 legend("topleft",
48        legend=c("Hidden States","Observations","Predictions","Confidence","Smoother^5"),
49        col=c('black','green','mediumorchid1','blue','red'), pch=c("o",NA,NA,NA,NA),
50        lty=c(NA,1, 1, 2, 1), cex=0.8)

```

```

51 # plot(Time, mu[-1] , main = 'Smooth', ylim = c (-5 ,10))
52 # lines(Time, y, col = 'green')
53 # lines(ks$xs) # state smoothers
54 # lines(ks$xs + 2 * sqrt(ks$Ps), lty = 2, col = 4)
55 # lines(ks$xs - 2 * sqrt(ks$Ps), lty = 2, col = 4)
56 # dev.off()
57 # mu[1]; ks$x0n; sqrt(ks$P0n) # initial value info
58 knitr:::include_graphics(c("images/plotA1.png"))
59 ## c) -----
60 # start figure
61 png(filename="images/plotA2.png", width = 1000, height = 600)
62 par(mfrow = c( 2, 1)); Time = 1:num
63 # plot with Q=1,R=1
64 ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 1, cR = 1)
65 plot(Time, mu[-1], main = 'Filter Q=1 R=1', ylim = c(-5, 13),lwd=2)
66 lines(Time ,y , col = 'green',lwd=2)
67 lines(ks$xf,col="mediumorchid1",lwd=2)
68 lines(ks$xf + 2 * sqrt (ks$Pf) ,lty = 2, col = 4,lwd=2)
69 lines(ks $ xf - 2 * sqrt (ks$Pf) , lty = 2 , col = 4 ,lwd=2)
70 legend("topleft",
71         legend=c("Hidden States","Observations","Predictions","Confidence"),
72         col=c('black','green','mediumorchid1','blue','red'), pch=c("o",NA,NA,NA),
73         lty=c(NA,1, 1, 2), cex=0.8)
74 # plot with R=0.1 ,R=10
75 ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 10, cR = 0.1)
76 plot(Time, mu[-1], main ='Filter Q=10 R= 0.1', ylim = c(-5, 13),lwd=2)
77 lines(Time ,y , col = 'green',lwd=2)
78 lines(ks$xf,col="mediumorchid1",lwd=2)
79 lines(ks$xf + 2 * sqrt (ks$Pf) ,lty = 2, col = 4)
80 lines(ks $ xf - 2 * sqrt (ks$Pf) , lty = 2 , col = 4 )
81 legend("topleft",
82         legend=c("Hidden States","Observations","Predictions","Confidence"),
83         col=c('black','green','mediumorchid1','blue','red'), pch=c("o",NA,NA,NA),
84         lty=c(NA,1, 1, 2), cex=0.8)
85 #
86 dev.off()
87 mu[1]; ks$x0n; sqrt(ks$P0n) # initial value info
88 knitr:::include_graphics(c("images/plotA2.png"))
89 ## d) -----
90 # start figure
91 png(filename="images/plotA3.png", width = 1000, height = 600)
92 par(mfrow = c( 2, 1))
93 # plot with Q=1, R=1
94 ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 1, cR = 1)
95 plot(Time, mu[-1], main = 'Filter Q=1 R=1', ylim = c(-5, 13),lwd=2)
96 lines(Time ,y , col = 'green',lwd=2)
97 lines(ks$xf,col="mediumorchid1",lwd=2)
98 lines(ks$xf + 2 * sqrt (ks$Pf) ,lty = 2, col = 4)
99 lines(ks $ xf - 2 * sqrt (ks$Pf) , lty = 2 , col = 4 )
100 legend("topleft",
101         legend=c("Hidden States","Observations","Predictions","Confidence"),
102         col=c('black','green','mediumorchid1','blue'), pch=c("o",NA,NA,NA),
103         lty=c(NA,1, 1, 2), cex=0.8)

```

```

104 # plot with Q=0.1,R=10
105 ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 0.1, cR = 10)
106 plot(Time, mu[-1], main = 'Filter Q=0.1 R=10', ylim = c(-5, 13), lwd=2)
107 lines(Time ,y , col = 'green',lwd=2)
108 lines(ks$xf,col="mediumorchid1",lwd=2)
109 lines(ks$xf + 2 * sqrt (ks$Pf) ,lty = 2, col = 4)
110 lines(ks $ xf - 2 * sqrt ( ks$Pf ) , lty = 2 , col = 4 )
111 legend("topleft",
112     legend=c("Hidden States","Observations","Predictions","Confidence"),
113     col=c('black','green','mediumorchid1','blue'), pch=c("o",NA,NA,NA),
114     lty=c(NA,1, 1, 2), cex=0.8)
115 #
116 dev.off()
117 mu[1]; ks$x0n; sqrt(ks$P0n) # initial value info
118 knitr:::include_graphics(c("images/plotA3.png"))
119 ## e) -----
120 kalman_filter <- function(At, Ct, Qt, Rt, m0, P0, xt) {
121   # initialization
122   bigT=length(xt)
123   mt=rep(0, bigT+1)
124   mt[1]=m0
125   Pt=rep(0, bigT+1)
126   Pt[1]=P0
127   for (t in 1:(bigT)) {
128     Kt=Pt[t]*t(Ct)*solve(Ct*Pt[(t)]*t(Ct)+Rt)
129     mt[t]=mt[(t)] + Kt*(xt[(t)] - Ct*mt[t])
130     Pt[t]=(1 - Kt*Ct)*Pt[t]
131     #
132     mt[(t+1)]= At*mt[t]
133     Pt[(t+1)]= At*Pt[t]*t(At) + Qt
134   }
135   return(list(mt = mt, Pt = Pt))
136 }
137 # start the plot
138 png(filename="images/plotA4.png", width = 1000, height = 900)
139 par(mfrow = c( 3, 1))
140 # Comparison Q and R one to one ratio
141 # our kalman predictions
142 k=kalman_filter(At = 1, Ct = 1, Qt = 1, Rt = 1, m0=0, P0=1, xt =y)
143 ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 1, cR = 1)
144 plot(Time, mu[-1], main = 'Filter Q=1 R=1', ylim = c(-5, 13), lwd=2)
145 lines(Time ,y , col = 'green',lwd=2)
146 lines(ks$xf,col="orange",lwd=2)
147 r=length(k$Pt)-1
148 U=k$mt[-r]+2*sqrt(k$Pt[r])
149 L=k$mt[-r]-2*sqrt(k$Pt[r])
150 lines(U,col="blue",lty=2)
151 lines(L,col="blue",lty=2)
152 lines(k$mt[-length(k$mt)], col="red",lwd=2)
153 legend("topleft",
154     legend=c('Hidden
155       ↳ States','Observations','Ksmooth_Preds','Confidence','mypredictions'),
156     col=c('black','green',"orange",'blue','red'), pch=c("o",NA,NA,NA,NA),

```

```

156     lty=c(NA,1, 1, 2,1), cex=0.8)
157 # comparison greater Q ratio
158 k=kalman_filter(At = 1, Ct = 1, Qt = 10, Rt = 0.1, m0=0, P0=1, xt =y)
159 ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 10, cR = 0.1)
160 plot(Time, mu[-1], main = 'Filter Q=10 R=0.1', ylim = c(-5, 13),lwd=2)
161 lines(Time ,y , col = 'green',lwd=2)
162 lines(ks$xf,col="orange",lwd=2)
163 r=length(k$Pt)-1
164 U=k$mt[-r-1]+2*sqrt(k$Pt[r])
165 L=k$mt[-r-1]-2*sqrt(k$Pt[r])
166 lines(U,col="blue",lty=2)
167 lines(L,col="blue",lty=2)
168 lines(k$mt[-length(k$mt)], col="red",lwd=2)
169 legend("topleft",
170         legend=c('Hidden
171             ↳ States','Observations',"Ksmooth_Preds",'Confidence','mypredictions'),
172             col=c('black','green',"orange",'blue','red'),
173             pch=c("o",NA,NA,NA,NA),lty=c(NA,1, 1, 2,1), cex=0.8)
174 # comparison with grater R ratio
175 k=kalman_filter(At = 1, Ct = 1, Qt = 0.1, Rt = 10, m0=0, P0=1, xt =y)
176 ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 0.1, cR = 10)
177 plot(Time, mu[-1], main = 'Filter Q=0.1 R=10', ylim = c(-5, 13),lwd=2)
178 lines(Time ,y , col = 'green',lwd=2)
179 lines(ks$xf,col="orange",lwd=2)
180 r=length(k$Pt)-1
181 U=k$mt[-r-1]+2*sqrt(k$Pt[r])
182 L=k$mt[-r-1]-2*sqrt(k$Pt[r])
183 lines(U,col="blue",lty=2)
184 lines(L,col="blue",lty=2)
185 lines(k$mt[-length(k$mt)], col="red",lwd=2)
186 legend("topleft",
187         legend=c('Hidden
188             ↳ States','Observations',"Ksmooth_Preds",'Confidence','mypredictions'),
189             col=c('black','green',"orange",'blue','red'),
190             pch=c("o",NA,NA,NA,NA),lty=c(NA,1, 1, 2,1), cex=0.8)
191 dev.off()
192 knitr:::include_graphics(c("images/plotA4.png"))

```

Kalman filter functions

Andreas C Charitos[andch552]

10/16/2019

Contents

Kalman filter with astsa package	1
Kalman filter implementation	2
Kalman filter vectorized	3

Kalman filter with astsa package

```
# -----
library(astsa)
## script given for the lab -----
# generate data
set.seed(12345); num=50
w=rnorm(num + 1, 0, 1); v=rnorm( num, 0, 1)
mu=cumsum(w) # state : mu [ 0 ] , mu [ 1 ] ,... , mu [ 5 0 ]
y = mu[-1] + v # obs : y [ 1 ] ,... , y [ 5 0 ]
# filter and smooth ( Ksmooth 0 does both )
ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ =1, cR = 1)
moving_avg_smooth=filter(y,rep(0.2, 5),sides = 1)
# start figure
par(mfrow = c( 2, 2)); Time = 1:num
plot(Time, mu[-1], main = 'Predict', ylim = c(-5, 10))
lines(Time ,y , col = 'green' )
lines(ks$xp)
lines(ks$xp + 2 * sqrt (ks$Pp), lty = 2, col = 4)
lines(ks$xp - 2 * sqrt (ks$Pp), lty = 2, col = 4)
plot(Time, mu[-1], main = 'Filter', ylim = c(-5, 10))
lines(Time ,y , col = 'green')
lines(ks$xf)
lines(ks$xf + 2 * sqrt (ks$Pf) ,lty = 2, col = 4)
lines(ks $ xf - 2 * sqrt ( ks $ Pf ) , lty = 2 , col = 4 )
plot(Time,moving_avg_smooth,main="Moving Average Smoother",col="black")
lines(Time,moving_avg_smooth,col="green")
plot(Time, mu[-1] , main = 'Smooth', ylim = c (-5 ,10))
lines(Time, y, col = 'green')
lines(ks$xs)
lines(ks$xs + 2 * sqrt(ks$Ps), lty = 2, col = 4)
lines(ks$xs - 2 * sqrt(ks$Ps), lty = 2, col = 4)
mu[1]; ks$x0n; sqrt(ks$P0n) # initial value info
```

Kalman filter implementation

```
# Kalmar Filter ----

kalman_func<-function(A,B,C,Q,R,m0,P0,xt,ut){

  n=length(xt)
  mt=double(n+1) ; mt[1]=m0
  Pt=double(n+1) ; Pt[01]=P0
  #R=chol(R) ; C=chol(C)

  for(t in 1:length(xt)) {
    # odxervation update step
    Kt=Pt[(t)]*t(C)*(1/(C*Pt[(t)]*t(C)+R))
    # print(Kt)
    mt[t]=mt[(t)]+Kt*(xt[t]-C*mt[(t)])
    #
    Pt[t]=Pt[(t)]-Kt*C*Pt[(t)]
    # prediction step
    mt[(t+1)]=A*mt[t]
    Pt[(t+1)]=A*Pt[t]*t(A)+Q
  }
  return(list(predicttions=mt[-(n+1)],Pt=Pt[-(n+1)]))
}

k=kalman_func( A=1,B=1, C=1, Q=0.1,R=10, m0=1,P0=1,y )

# plot(y,type="l")
# lines(k$predicttions)

plot(Time, mu[-1], main = 'Predict', ylim = c(-5, 10))
lines(Time ,y , col = 'green' )
lines(k$predicttions,col="red")
```

Kalman filter vectorized

```
# Vectorized ----

# generate data
set.seed(12345); num=50
w=rnorm(num + 1, 0, 1); v=rnorm( num, 0, 1)
mu=cumsum(w) # state : mu [ 0 ] , mu [ 1 ] ,... , mu [ 5 0 ]
y = mu[-1] + v # obs : y [ 1 ] ,..., y [ 5 0 ]
# filter and smooth ( Ksmooth 0 does both )

kalman_func_vect<-function(A,C,Q,R,m0,P0,xt,verbose){
  #Q=chol(Q,tol=-1) ; R=chol(R,tol = -1) # cholesky decomposition of the covariance matrices
  d1=dim(P0)[1] ; d2=dim(P0)[2] ; Tt=length(xt)
  d3=length(m0)
  mt_t=matrix(NA,nrow=Tt+1,ncol=d3)
  mt_t[1,]=m0
  Pt_t=array(NA,dim=c(d1,d2,Tt+1)) #; print(dim(Pt_t))
  Pt_t[,1]=P0
  for(t in 1:(Tt)){
    if(verbose==1){
      cat("---iteration: ",t,"\\n")
    }
    # odxervation update step
    Kt=Pt_t[,,(t)]%*%t(C)%*%solve(C%*%Pt_t[,,(t)]%*%t(C)+R)
    #
    mt_t[t,]=mt_t[(t),]+Kt%*%(xt[(t)]-C%*%mt_t[(t),])
    #
    Pt_t[,,t]=(diag(d1)-Kt%*%C)%*%Pt_t[,,(t)]
    # prediction step
    mt_t[(t+1),]=A%*%mt_t[t,]
    #
    Pt_t[,,(t+1)]=A%*%Pt_t[,,t]%*%t(A)+Q
    #
  }
  return(list(mt=mt_t,Pt=Pt_t))
}

E=diag(2)*1 ; E

k=kalman_func_vect( A=E, C=E,
                     Q=E*1,R=E*1, m0=c(1,3),P0=E,y,verbose=0)
Time = 1:50
plot(Time, mu[-1], main = 'Predict', ylim = c(-5, 10))
ks=Ksmooth0(num, y, A = 1, mu0 = 0, Sigma0 = 1, Phi = 1, cQ = 1, cR = 1)
lines(Time ,y , col = 'green' )
lines(k$mt[,1],col="red")
U=k$mt[-51,2]+2*sqrt(abs(k$Pt[1,1,50]))
L=k$mt[-51,2]-2*sqrt(abs(k$Pt[1,1,50]))
lines(U,col="blue",lty=2)
lines(L,col="blue",lty=2)
#lines(ks$xf + 2 * sqrt (ks$Pf) ,lty = 2, col = 4) # the same bands with the Ksmooth
```

```
#lines(ks $ xf - 2 * sqrt ( ks$Pf ) , lty = 2 , col = 4 ) # same bands with the Ksmooth
legend("bottomright",legend = c("states","obs","kalman preds","bands"),
       col=c("black","green","red","blue"),pch=c("o",NA,NA,NA),
       lty=c(NA,1,1,2))
```

Time Series Analysis

Lecture 1: Introduction

Tohid Ardeshtiri

Linköping University
Division of Statistics and Machine Learning

September 2, 2019



Course teacher

Tohid Ardeshiri

PhD in 2015 in Bayesian inference



LINKÖPING
UNIVERSITY



UNIVERSITY OF
CAMBRIDGE

Senior Data Scientist at
Qamcom Research & Technology AB



Linköping Studies in Science and Technology. Dissertations.
No. 1710

Analytical
Approximations for
Bayesian Inference

Tohid Ardeshiri



LINKÖPING
UNIVERSITY

Bayesian Inference

Bayesian inference is a means of combining prior beliefs with the data (evidence) to obtain posterior beliefs.

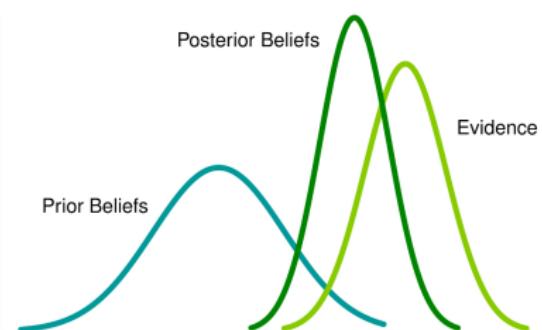
Example: Parameter learning

$$f(\theta|x) \propto f(x|\theta)f(\theta)$$

Probability Calculus

$$f(\theta, x) = f(x|\theta)f(\theta)$$

$$f(\theta, x) = f(\theta|x)f(x)$$



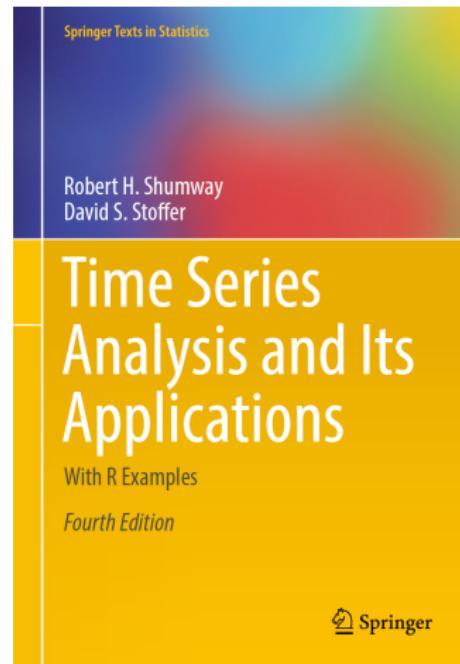
Course literature and software

Course literature:

Time series Analysis and its Applications
Can be downloaded freely here:

<https://www.stat.pitt.edu/stoffer/tsa4/tsa4.pdf>

Software for computer labs is R:



Sequential data



Sequential data: Motion of a ball



Sequential data: A sentence

This is a sequential data type.

Sequential data: A sentence

This is a sequential data type.

This is a sequential data type .

Sequential data: A sentence or a word

This is a sequential data type.

This is a sequential data type .

s e q u e n t i a l

A look at real data

Received signal strength indicator (RSSI) is a common observation (data).



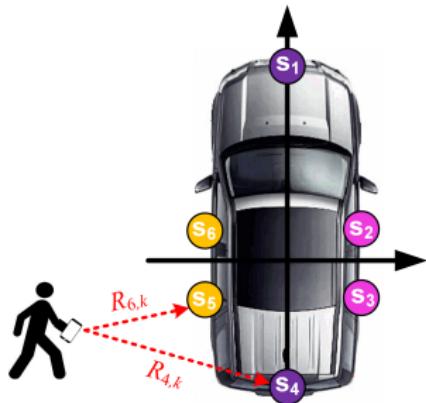
Where is the driver?

A look at real data

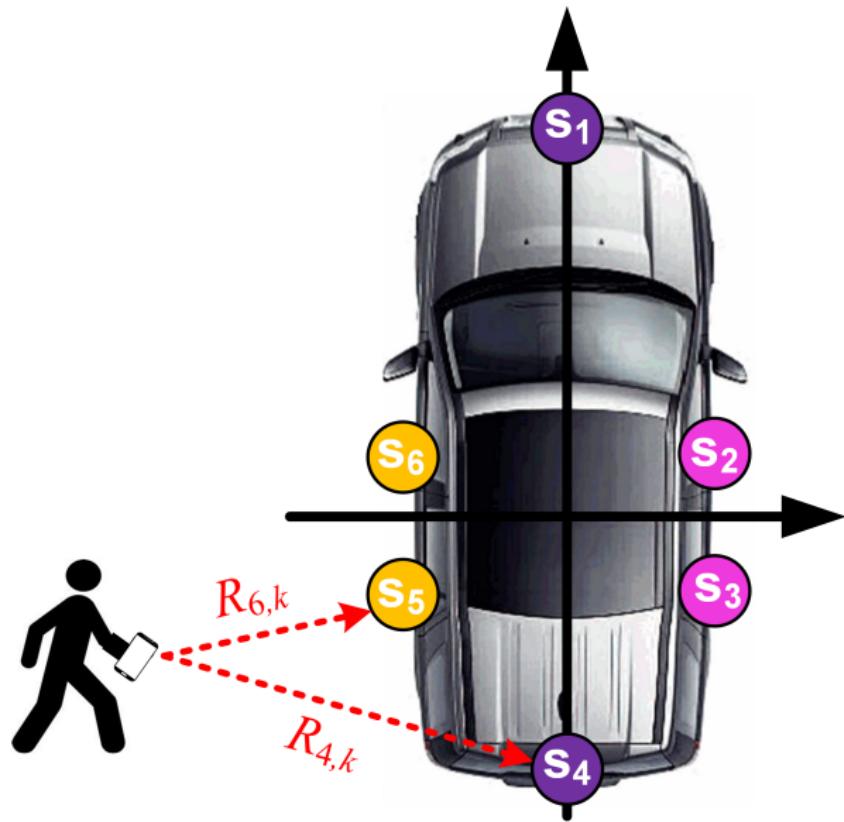
Received signal strength indicator (RSSI) is a common observation (data).



Where is the driver?



Where is the driver?



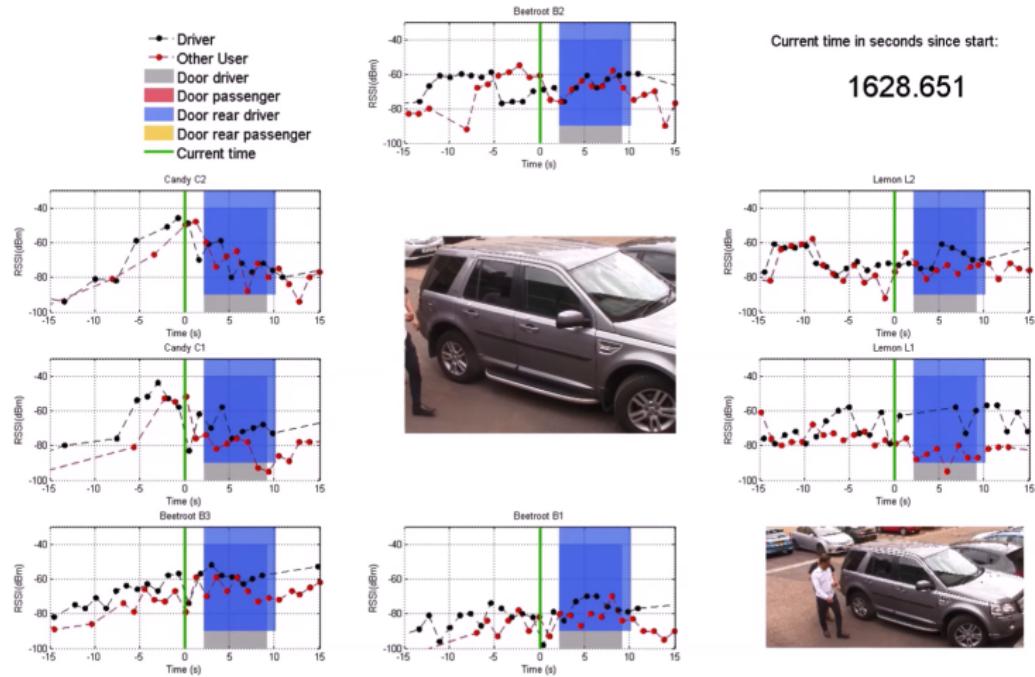
Where is the driver?

Video of data collection



Where is the driver?

Animation of the of signals



Current time in seconds since start:

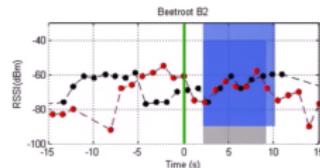
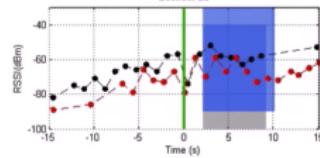
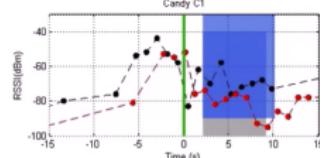
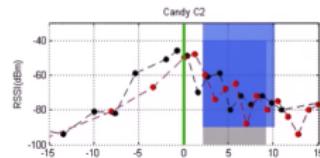
1628.651

Video is Proprietary to Cambridge/Tohid Ardestiri

Time Series Analysis

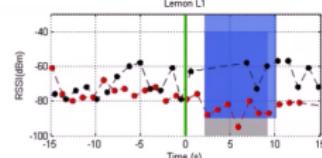
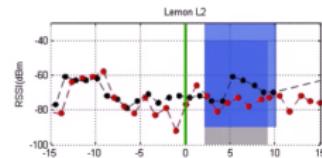
What is a Time Series?

- A sequential data where observations are collected over time
- Observations are typically **correlated!**



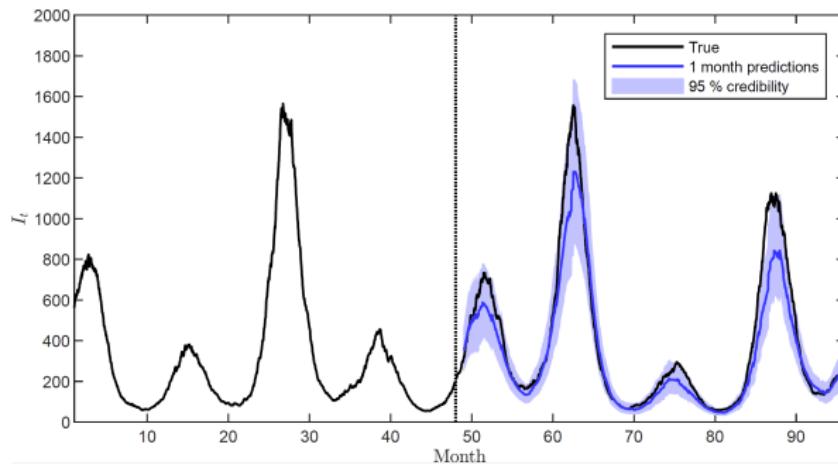
Current time in seconds since start:

1628.651



Time Series Analysis

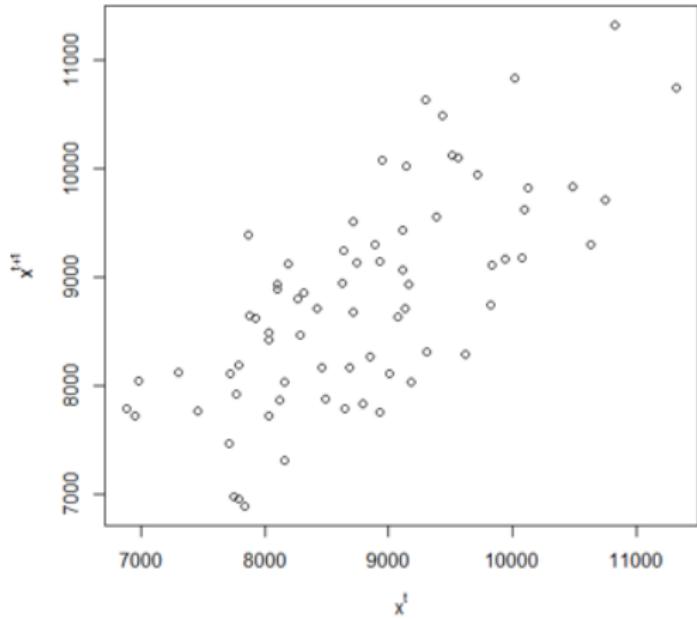
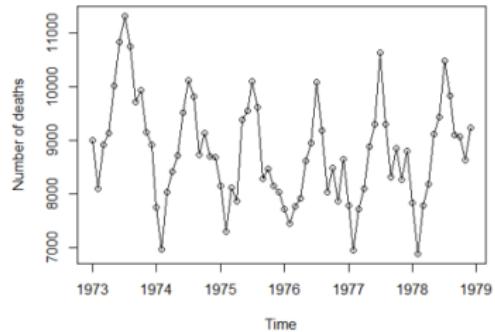
- Understand the properties of the underlying process
- Be able to predict (forecast) possible future values
- Reason about the **uncertainties** in the predictions
requires statistical methods!



Time Series Analysis

Usual regression analysis: observations are often **iid.**

Time Series Analysis: observations are **correlated!**

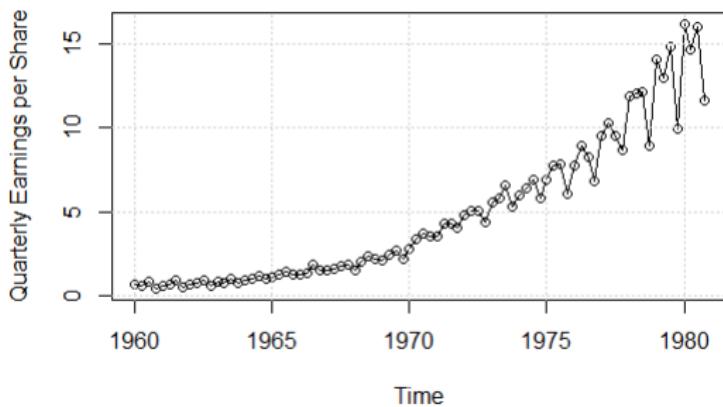


Ex) See connection
between x_t and x_{t+1}

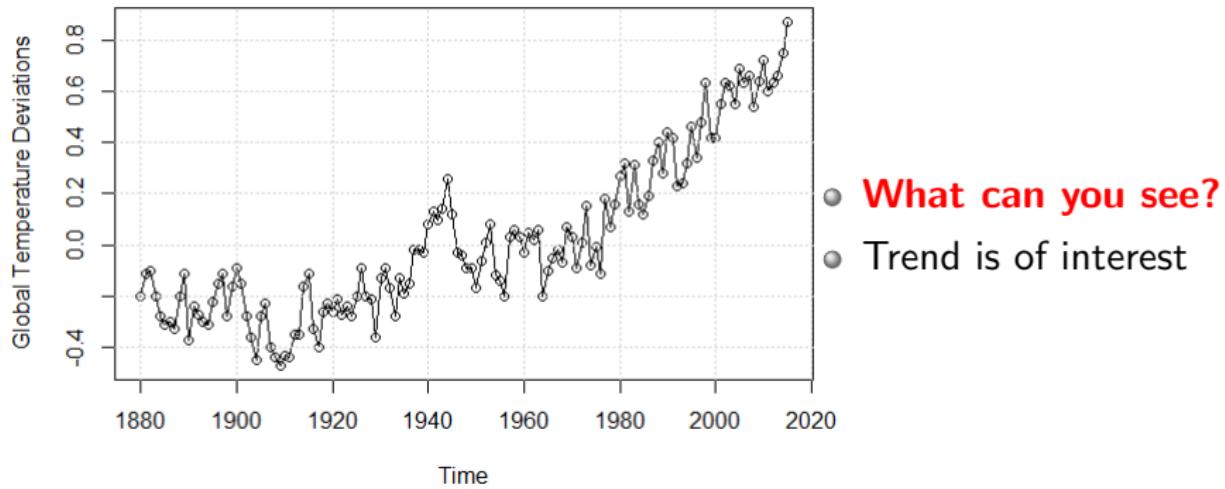
Ex 1: Johnson & Johnson quarterly earnings

- **What can you see?**

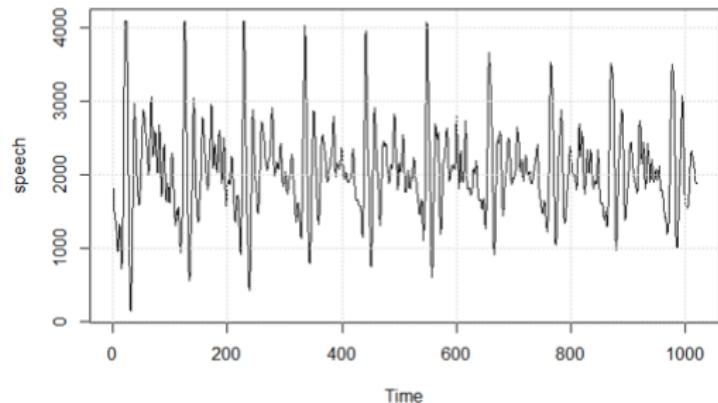
- ▶ Trend?
 - ★ Constant
 - ★ Linear
 - ★ Other
- ▶ Variation?
- ▶ Seasonality?
- ▶ Outliers?



Ex 2: Global warming



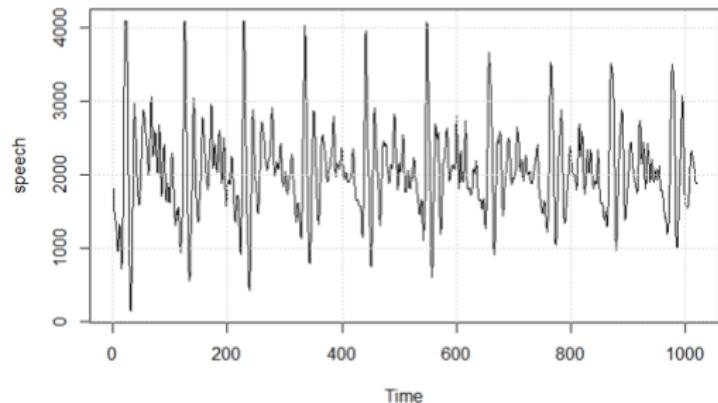
Ex 3: Speech data



- **What can you see?**

Pattern of periodicity is of interest → decompose signal into different frequencies

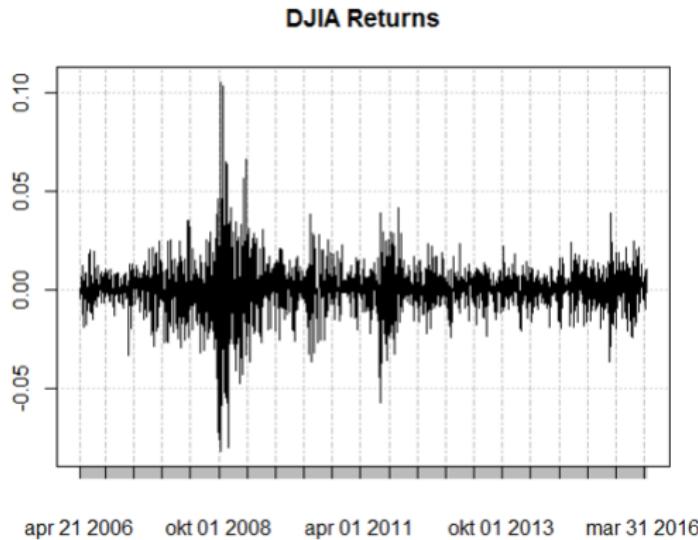
Ex 3: Speech data



• **What can you see?**

Pattern of periodicity is of interest → decompose signal into different frequencies
not covered in this course!

Ex 4: Dow Jones Industrial Average

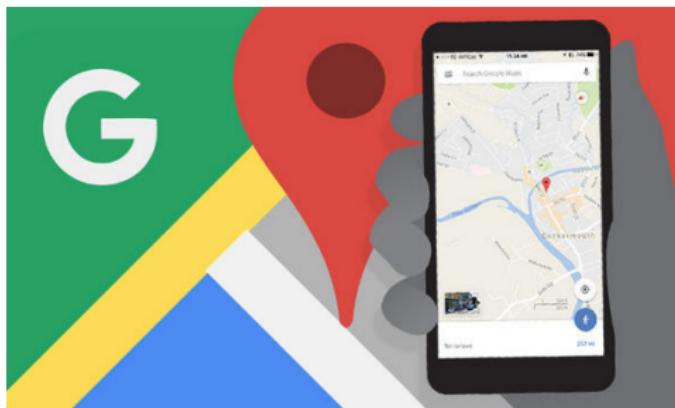
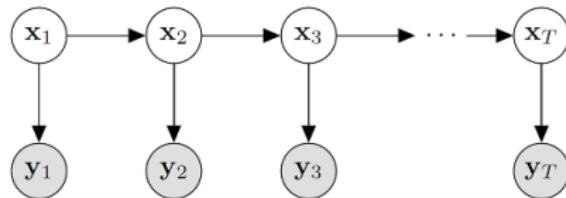


- What can you see here?

Pattern of periodicity is of interest → Stochastic volatility

Ex 5: Dynamical systems

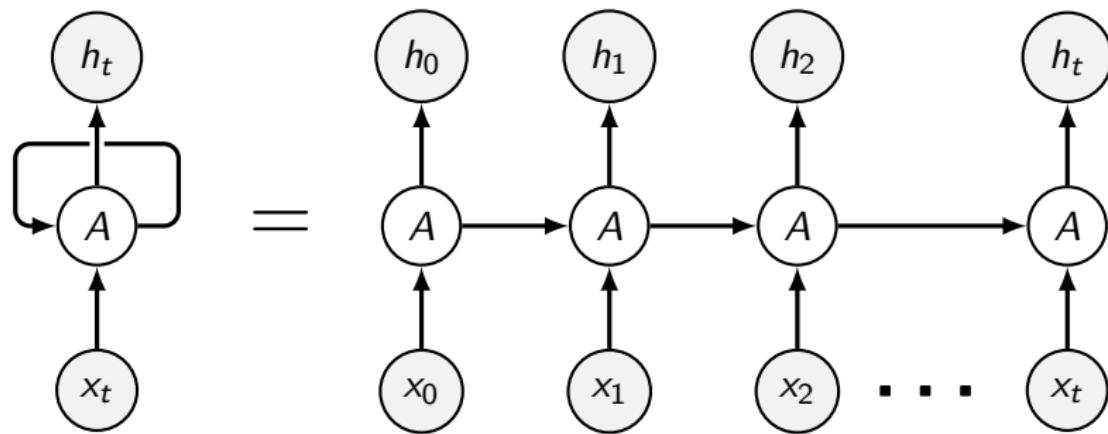
Linear and Gaussian state-space models for tracking objects



Ex 6: Recurrent neural networks

Natural Language Processing

This is a sequential data type .

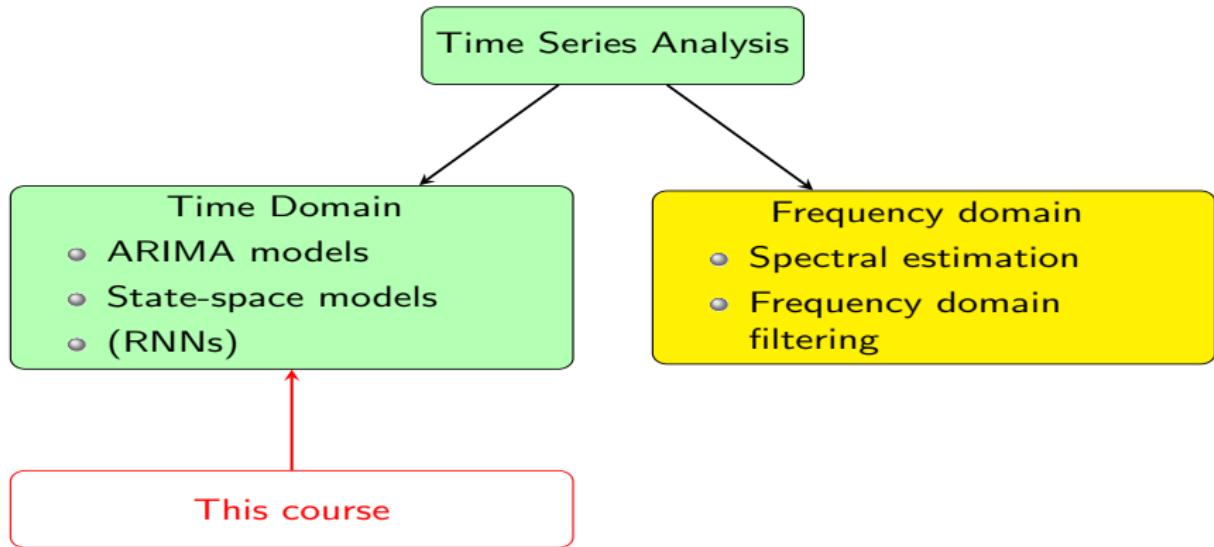


Time Series Analysis

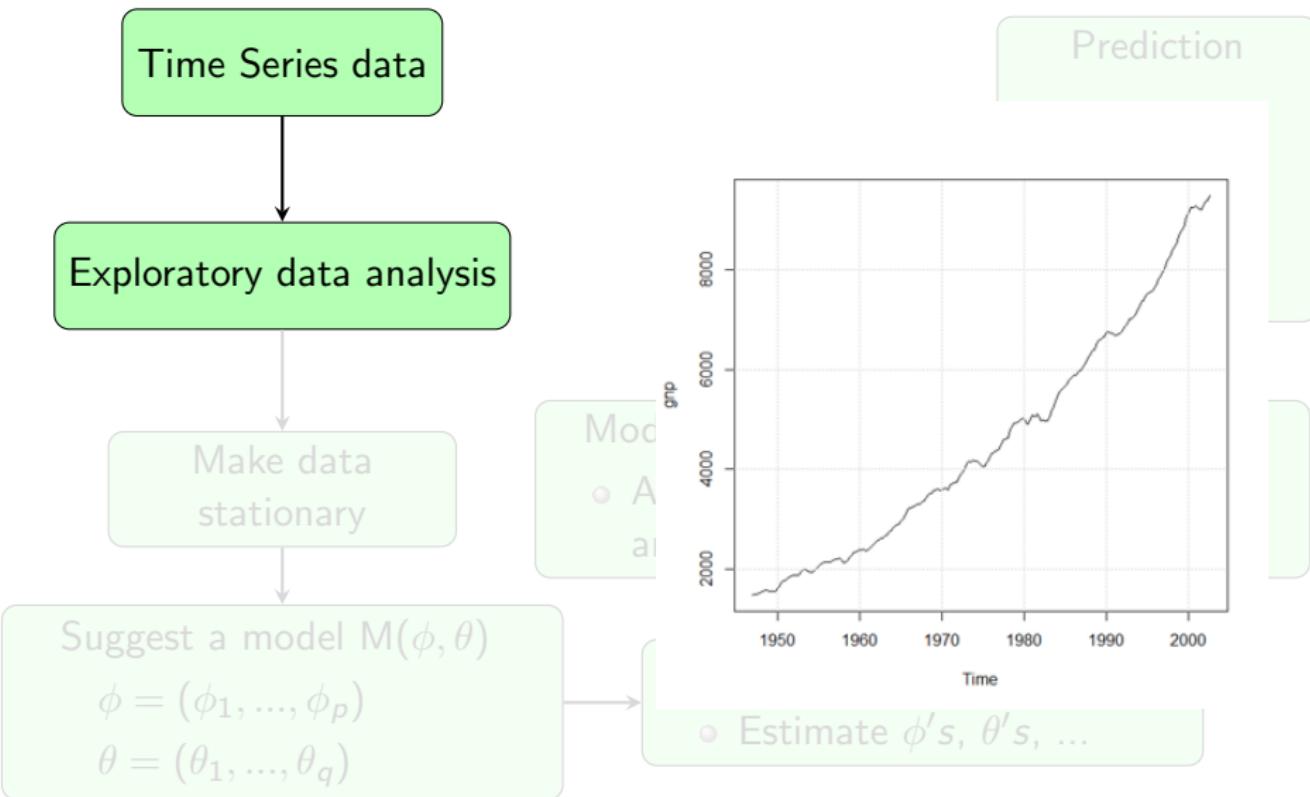
Application areas

- Natural sciences
- Climatology
- Robotics/autonomous systems
- Social sciences
- Medicine
- Economics
- Telecommunications
- ...

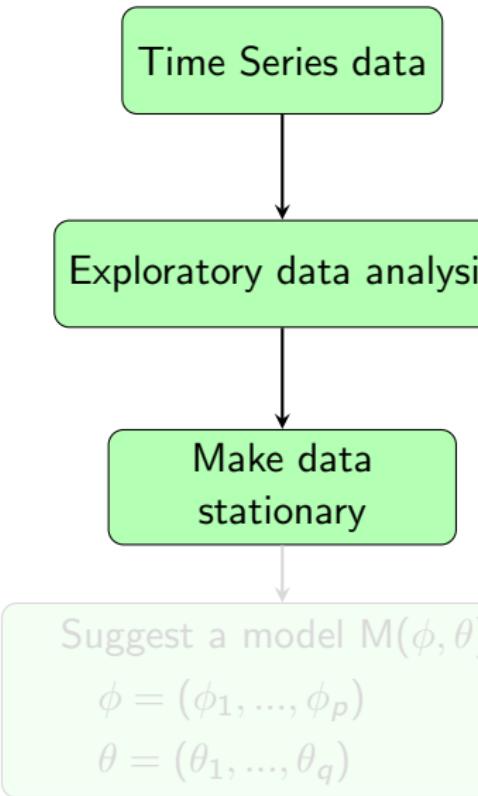
The Big Picture



Time domain: The Big Picture

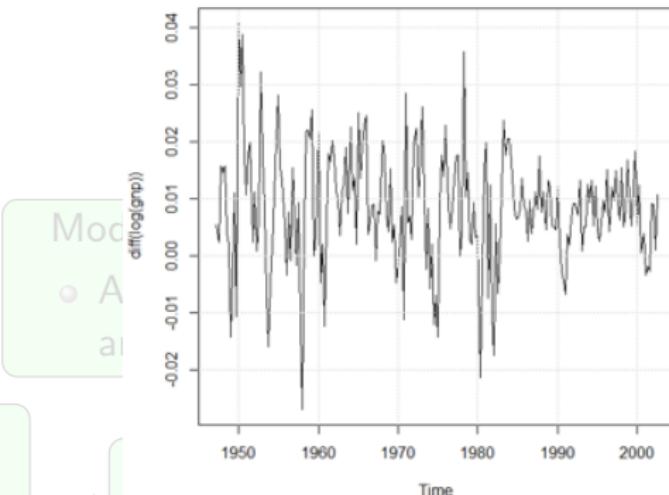


Time domain: The Big Picture



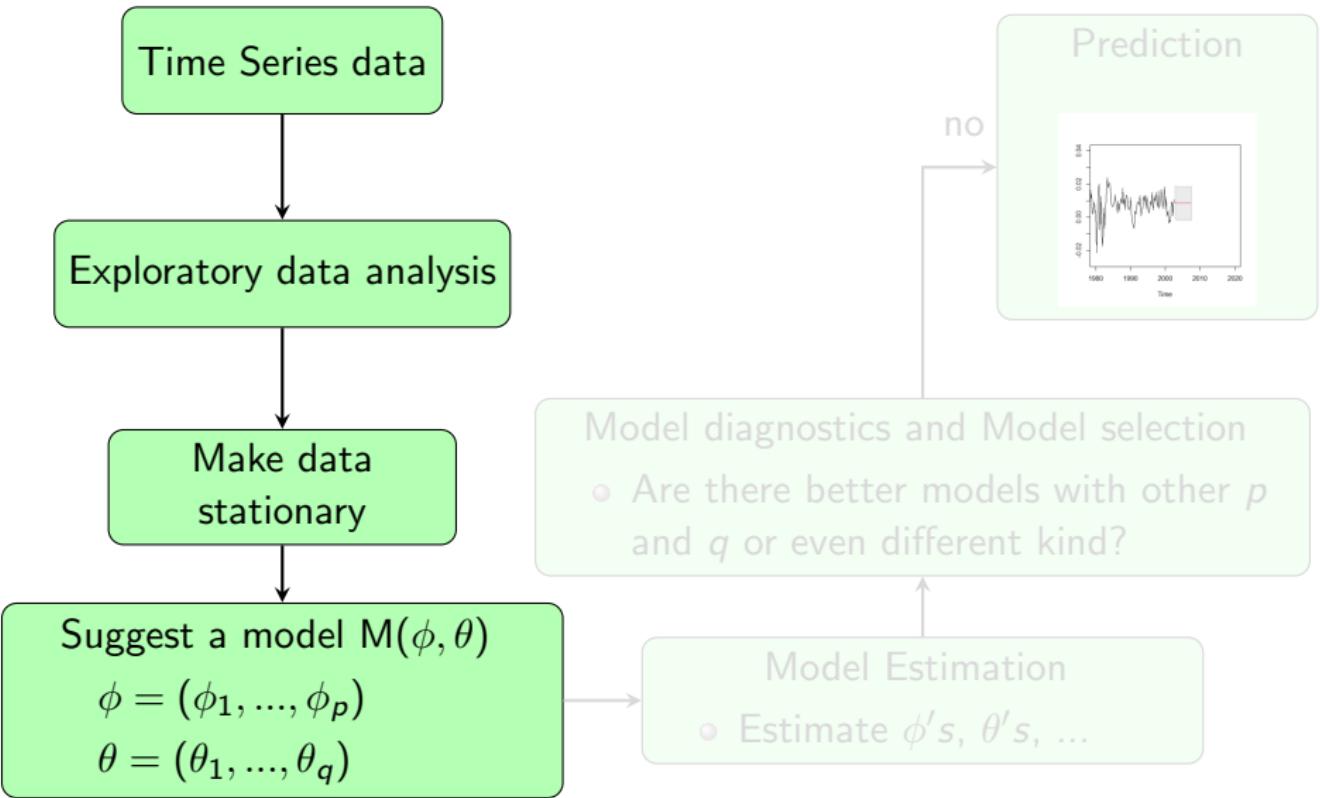
$$Y_t = \nabla(\log(X_t))$$

Prediction

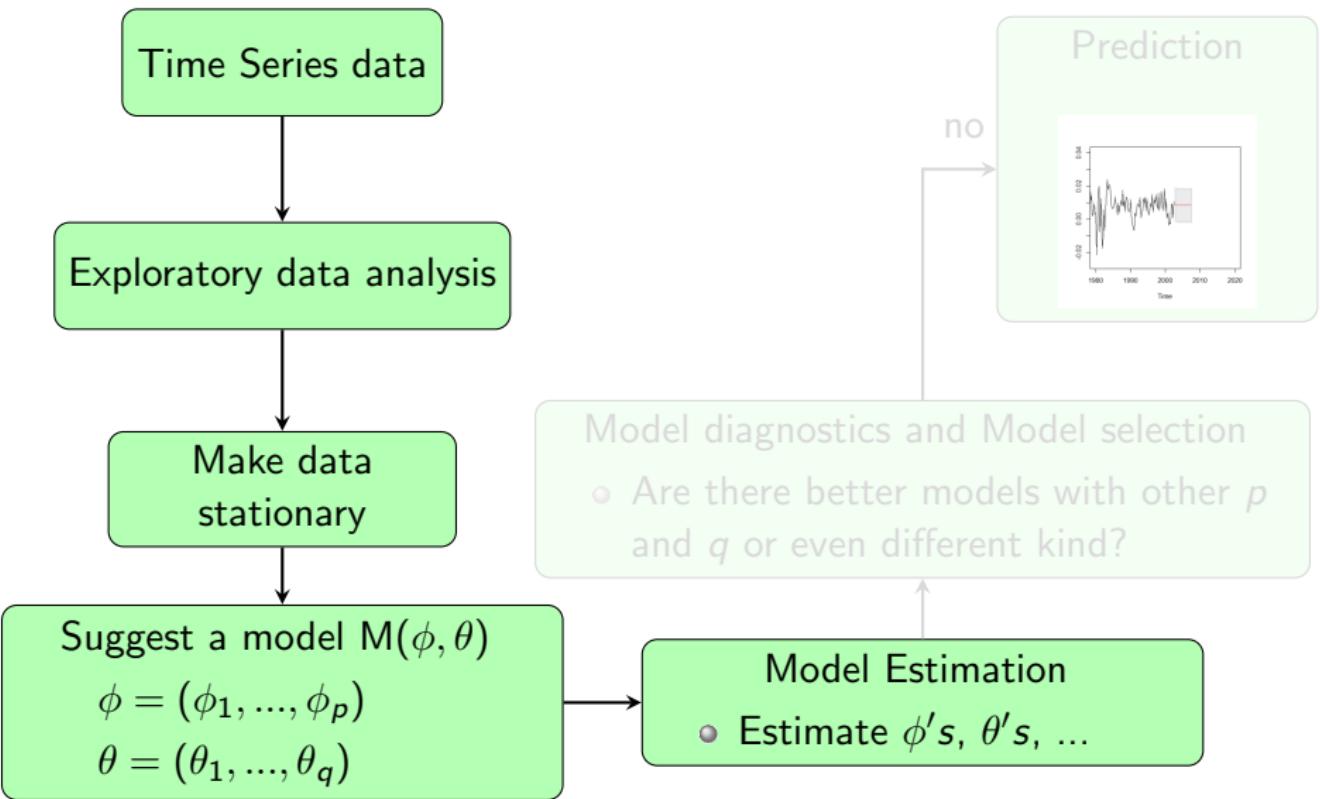


• Estimate ϕ 's, θ 's, ...

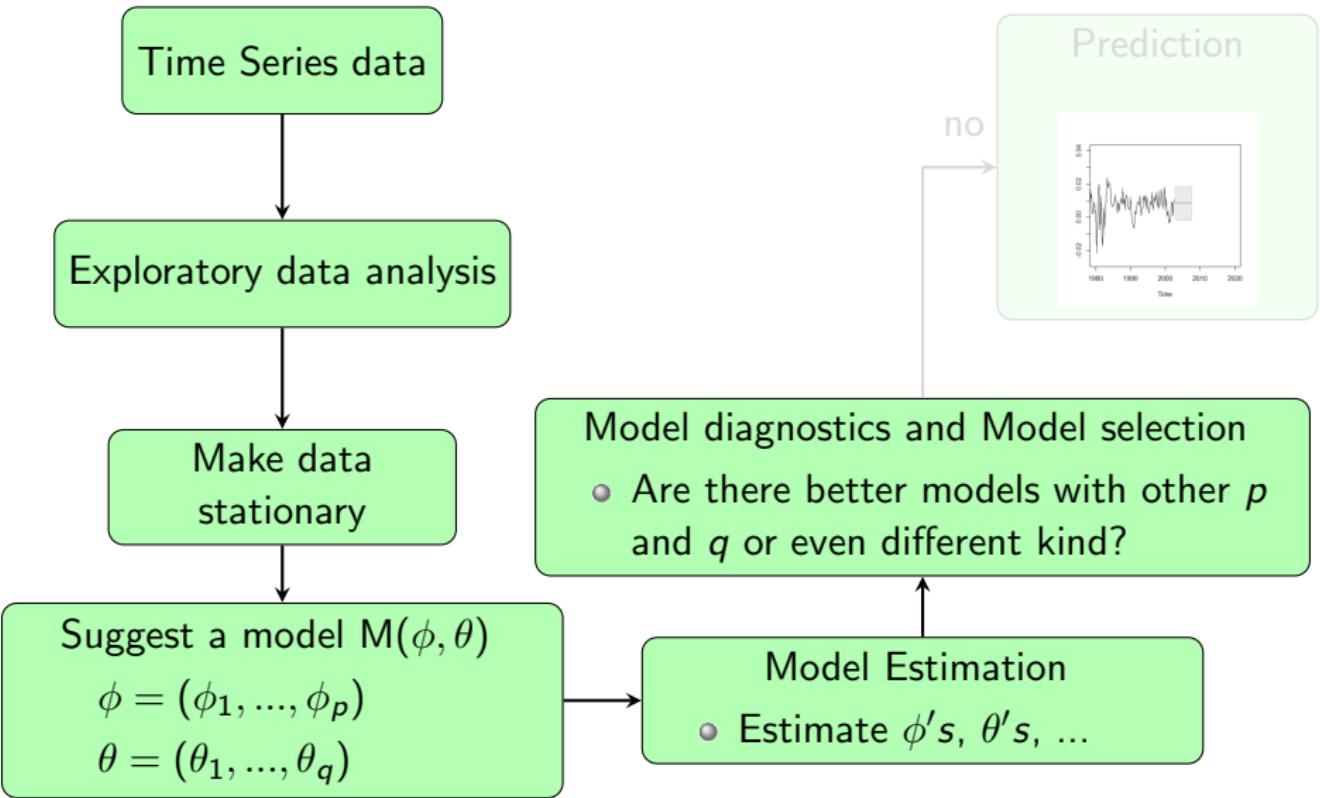
Time domain: The Big Picture



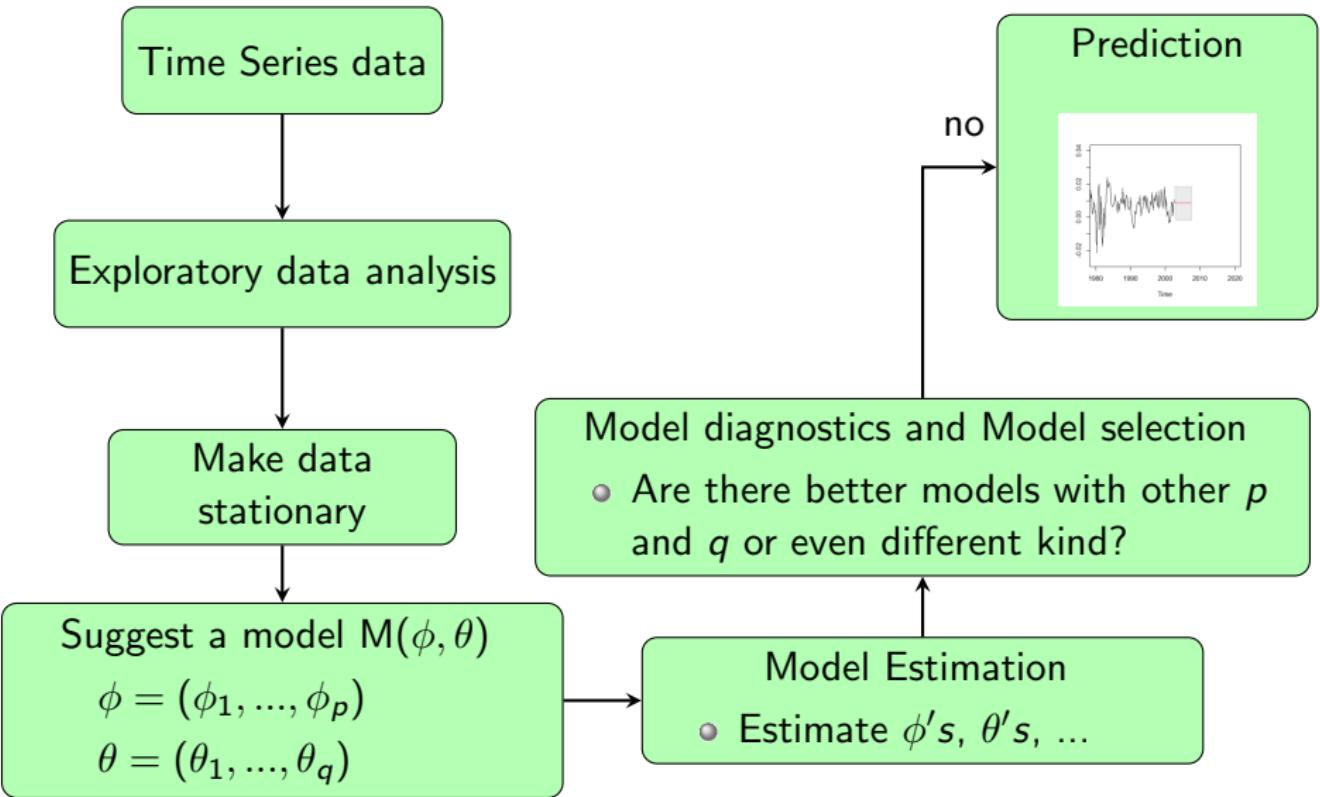
Time domain: The Big Picture



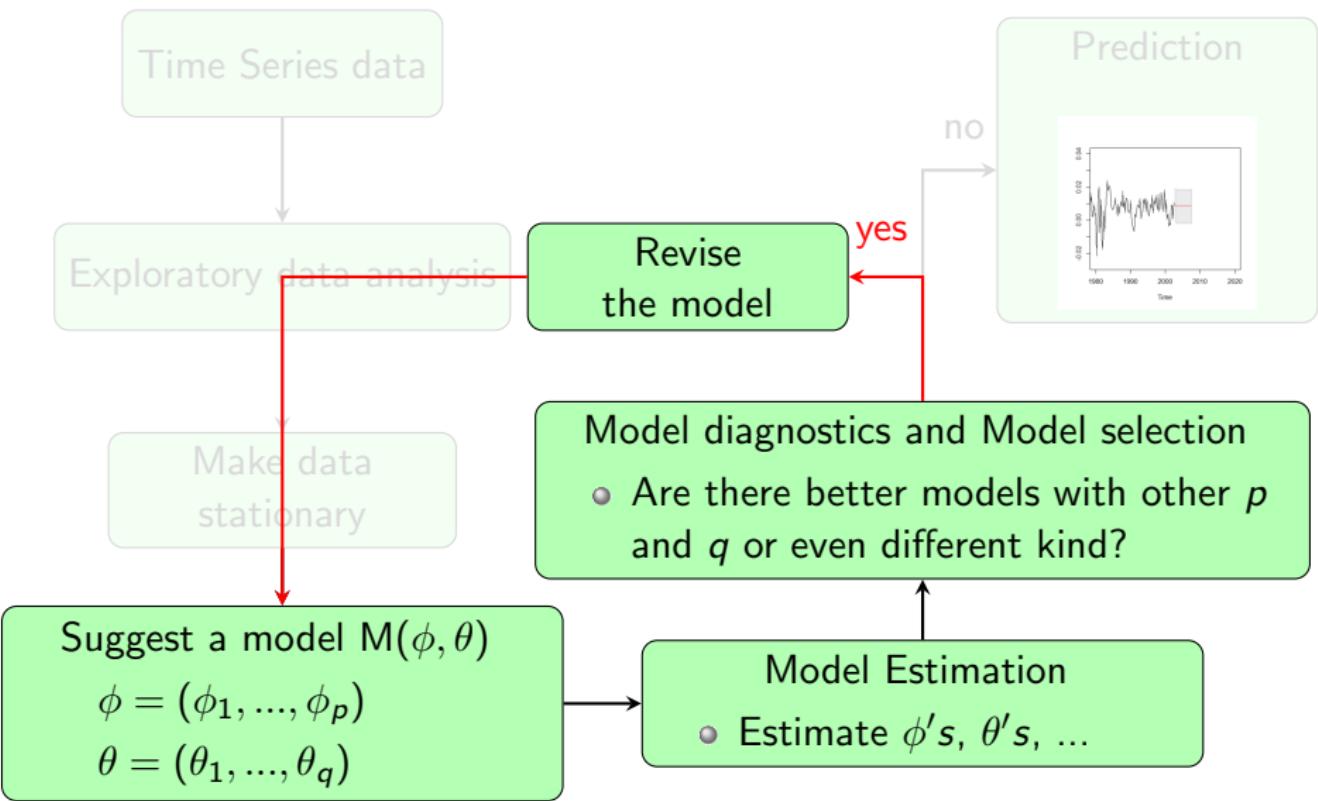
Time domain: The Big Picture



Time domain: The Big Picture



Time domain: The Big Picture



Course topics

- Time series regression and explorative analysis
- ARIMA models
 - ▶ AR, MA, ARMA, ARIMA, seasonal ARIMA
 - ▶ Model selection
 - ▶ Estimation
 - ▶ Forecasting
- State space models
 - ▶ Linear and Gaussian state space models
 - ▶ Kalman filtering and smoothing
- Recurrent Neural Networks (RNNs)

Course organization

- Lectures
 - ▶ Available at LISAM
- Teaching sessions
- Computer labs
 - ▶ Available at LISAM, under Submissions
 - ▶ Work in pairs
 - ▶ Send your report via LISAM
 - ▶ Deadlines
- Written assignments
 - ▶ Submissions needed - keys are given for some assignments
- Examination
 - ▶ Computer based exam
 - ▶ Submission of lab reports and written assignments

Course organization

- Software: R
 - ▶ <https://www.r-project.org/>
 - ▶ <https://www.rstudio.com/>



- Define your groups (2 persons) this week:
 - ▶ <https://docs.google.com/spreadsheets/d/1tzG35WSDWRhHWFA0cNOL1WUzoYqdn0GhZUwvXz3HhII/edit?usp=sharing>
 - ▶ **Difficult to find a group? Put your name in some cell.. I will merge you to someone**

Course organization

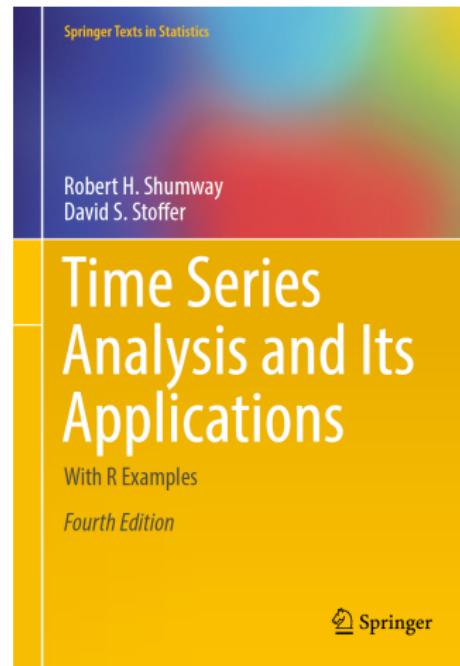
Course literature:

Time series Analysis and its Applications, Fourth Edition (2017), ISBN 978-3-319-52451-1

Can be downloaded freely here:

<https://www.stat.pitt.edu/stoffer/tsa4/tsa4.pdf>

- Do not skip examples when you read!
- First 2 chapters are easy, but don't relax!



Time Series models

- Time series x_t : random variable
 - ▶ A collection of $x_t =$ stochastic process
 - ▶ $t = 0, \pm 1, \pm 2, \dots$
- (probably) Simplest series: white noise
 - ▶ w_t uncorrelated (white: all possible periodic oscillations are present at equal strength)

$$w_t \sim wn(0, \sigma_w^2)$$

- ▶ w_t independent and identically distributed (white independent noise)

$$w_t \sim iid(0, \sigma_w^2)$$

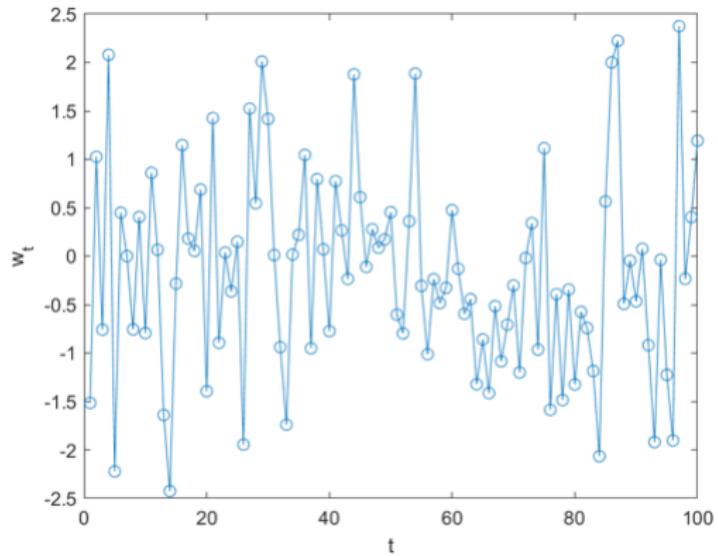
- Reminder:

$$\text{uncorrelated} \iff E(XY) = EX.EY$$

$$\text{independent} \iff f_{X,Y}(x,y) = f_X(x).f_Y(y)$$

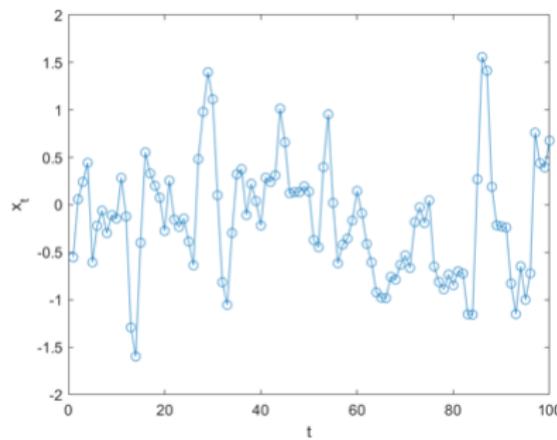
White noise

- Example: $w_t \sim iidN(0, 1)$



Moving average

Example: $x_t = 0.2w_{t-1} + 0.5w_t + 0.2w_{t+1}$



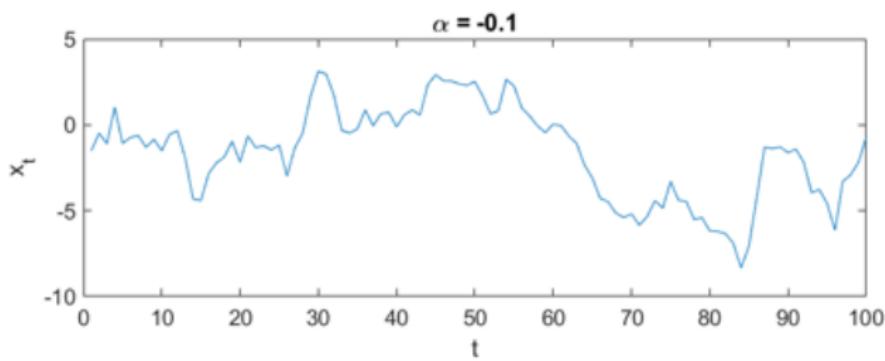
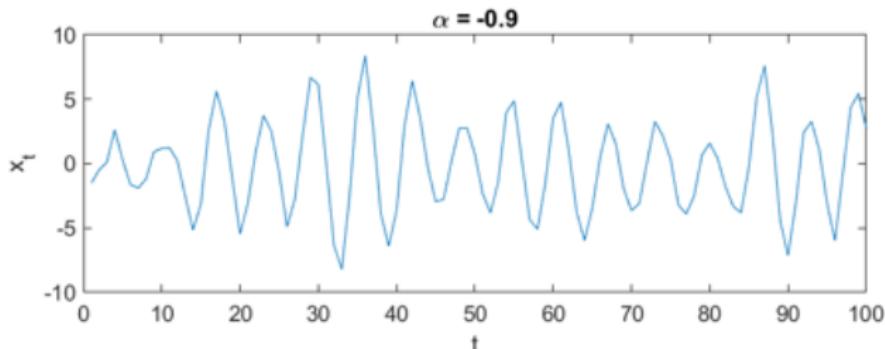
Very Interesting Fact: most stationary processes can be represented as a sum of lagged white noise:

$$x_t = \sum_{j=-\infty}^{\infty} \psi_j w_{t-j}$$

Autoregressive model

Example: AR(2) process (Assume $x_0 = 0, x_{-1} = 0$)

$$x_t = x_{t-1} + \alpha x_{t-2} + w_t$$



Random walk with drift

A simple model for a "drifting" time series

$$x_t = \delta + x_{t-1} + w_t$$

- δ is the drift
- $\delta = 0 \Rightarrow$ random walk

Note: if we assume $x_0 = 0$,

$$x_t = \delta t + \sum_{j=1}^t w_j$$

Random walk with drift

A simple model for a "drifting" time series

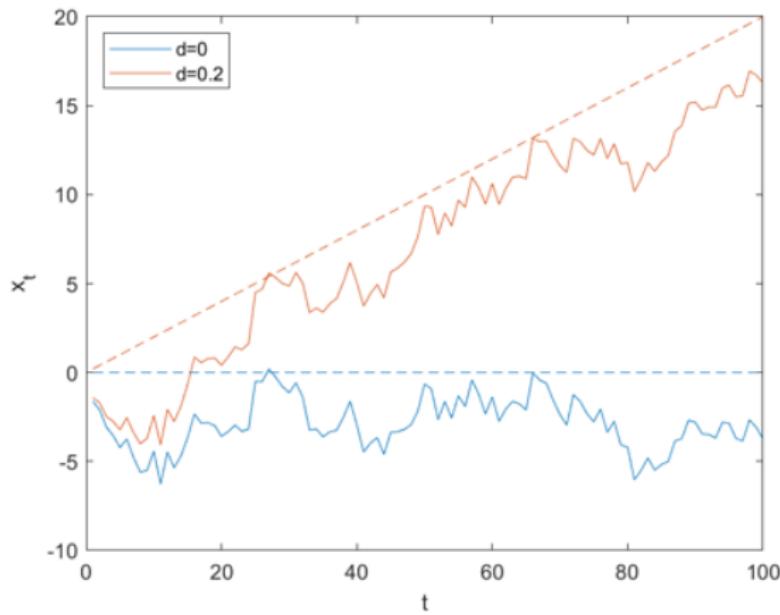
$\delta=0$ and $\delta=0.2$

$$x_t = \delta + x_{t-1} + w_t$$

- δ is the drift
- $\delta = 0 \Rightarrow$ random walk

Note: if we assume $x_0 = 0$,

$$x_t = \delta t + \sum_{j=1}^t w_j$$



Basic statistics - reminder

- Probability density function for x : $f(x)$
- Marginal density $f_i(x_i) = \int f(x) dx_1 \dots dx_{i-1} dx_{i+1} \dots dx_p$
- Expected (mean) value $Ex = \int xf(x)dx$
- Covariance $\text{cov}(x, y) = E\{(x - Ex)(y - Ey)\}$
- Variance $\text{var}(x) = E\{(x - Ex)^2\} = \text{cov}(x, x)$
- Relationships (a is a constant)
 - ▶ $E(x + a) = Ex + a$, $E(ax) = aEx$
 - ▶ $E(x + y) = Ex + Ey$
 - ▶ $\text{cov}(x + a, y) = \text{cov}(x, y)$
 - ▶ $\text{cov}(x + z, y) = \text{cov}(x, y) + \text{cov}(z, y)$
 - ▶ $\text{var}(ax) = a^2 \text{var}(x)$

Statistical representation of a time series

Which measures of dependence exist for time series?

- Theoretical?
- Practical?

Given time series x_1, \dots, x_n measured at fixed t_1, \dots, t_n

- Joint pdf

$$f_{t_1, \dots, t_n}(x_{t_1}, \dots, x_{t_n})$$

- Marginal pdf

$$f_t(x_t)$$

Statistical representation of a time series on whiteboard

Mean function at time t

$$\mu_t = E(x_t) = \int_{-\infty}^{\infty} xf_t(x)dx$$

Examples: Compute mean function for

- Moving average $x_t = 0.2w_{t-1} + 0.5w_t + 0.2w_{t+1}$
- Random walk $x_t = \delta t + \sum_{j=1}^t w_j$

Autocovariance and ACF

How do we measure linear dependence between two variables? → Covariance or Correlation

How do we measure linear dependence between two time-lags in a time series? In the same way!

- Autocovariance function

$$\gamma(s, t) = \text{cov}(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)]$$

Note $\text{var}(x_t) = \gamma(t, t)$

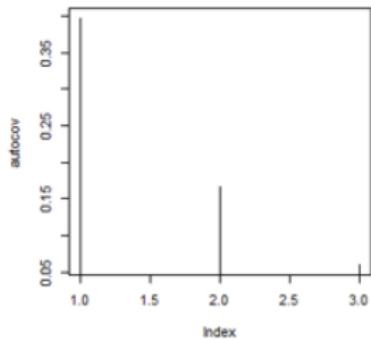
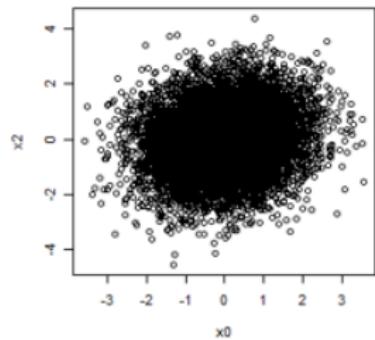
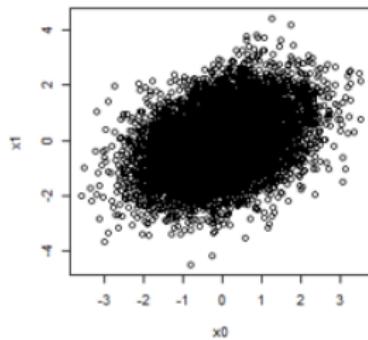
- Autocorrelation function (ACF)

$$\rho(s, t) = \frac{\gamma(s, t)}{\sqrt{\gamma(s, s)\gamma(t, t)}}$$

Autocovariance and ACF

Generate x_0, x_1, x_2 from $x_t = 0.4x_{t-1} + w_t$

- Consider $\gamma(0, 1), \gamma(0, 2)$



Autocovariance and ACF

Useful fact: If $U = \sum_{j=1}^m a_j x_j$ and

$$V = \sum_{k=1}^r b_k y_k$$

$$\text{cov}(U, V) = \sum_{j=1}^m \sum_{k=1}^r a_j b_k \text{cov}(x_j, y_k)$$

Examples: Autocovariance and ACF of on whiteboard

- White noise
- Random walk $x_t = \delta t + \sum_{j=1}^t w_j$
- Moving average $x_t = 0.2w_{t-1} + 0.5w_t + 0.2w_{t+1}$

Home reading

- Shumway and Stoffer, chapters 1.1-1.3
- TS functions in R: ts, plot.ts, acf, ts.intersect, filter, ts.plot

Time Series Analysis

Lecture 2: Exploratory analysis and Time Series Regression

Tohid Ardestiri

Linköping University
Division of Statistics and Machine Learning

September 4, 2019



LINKÖPING
UNIVERSITY

Summary of Lecture 1

- Time series
 - ▶ White noise
 - ▶ Random walk
 - ▶ Moving average filter
- Autocovariance and autocorrelation functions:

$$\gamma(s, t) = \text{cov}(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)]$$

$$\rho(s, t) = \frac{\gamma(s, t)}{\sqrt{\gamma(s, s)\gamma(t, t)}}$$

Autocovariance and ACF

Examples: Autocovariance and ACF of on whiteboard

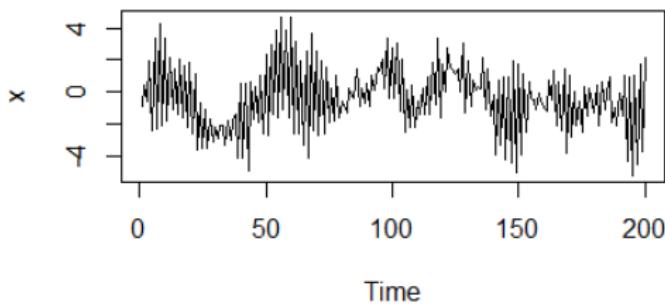
- White noise ✓
- Random walk $x_t = \delta t + \sum_{j=1}^t w_j$
- Moving average $x_t = 0.2w_{t-1} + 0.5w_t + 0.2w_{t+1}$

Autocovariance

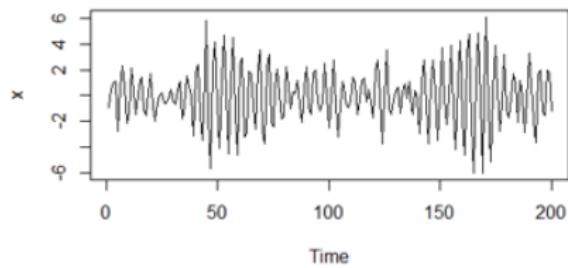
- Intuition:

$$x_t = \phi x_{t-1} + w_t$$

$$\alpha = 0.9$$



$$\alpha = -0.9$$



- when $x_0 = 0$ and $w_t \sim wn(0, 1)$:

$$\text{cov}(x_t, x_{t-1}) = \phi$$

Autocovariance (read at home)

$$x_t = \phi x_{t-1} + w_t$$

Mean function:

$$Ex_t = \phi Ex_{t-1} + Ew_t = \phi Ex_{t-1} = \phi(\phi Ex_{t-2}) = \dots = \phi^t Ex_0$$

for $Ex_0 = 0$, $Ex_t = 0$ for all t .

Variance $\text{var}(x_t)$ when $Ex_0 = 0$ and w_t is uncorrelated with x_0 for all t :

$$\begin{aligned}\text{var}(x_t) &= E\{(x_t - 0)^2\} = E\{\phi^2 x_{t-1}^2 + 2\phi x_{t-1} w_t + w_t^2\} = \\ \phi^2 \text{var}(x_{t-1}) + 2\phi \text{cov}(x_{t-1}, w_t) + \text{var}(w_t) &= \phi^2 \text{var}(x_{t-1}) + \text{var}(w_t) = \\ \phi^2 \text{var}(x_{t-1}) + \sigma_w^2 &= \phi^2(\phi^2 \text{var}(x_{t-2}) + \sigma_w^2) + \sigma_w^2 = \\ \phi^{2t} \text{var}(x_0) + \sigma_w^2 \sum_{k=0}^{t-1} (\phi^{2k}) &= \phi^{2t} \text{var}(x_0) + \frac{\sigma_w^2(1-\phi^{2t})}{1-\phi^2}\end{aligned}$$

When $\text{var}(x_0) = \frac{\sigma_w^2}{1-\phi^2}$ then $\text{var}(x_t) = \frac{\sigma_w^2}{1-\phi^2}$ and time independent.

Autocovariance (read at home)

$$x_t = \phi x_{t-1} + w_t$$

$$x_t = \phi(\phi x_{t-2} + w_{t-1}) + w_t = \dots = \phi^h x_{t-h} + \sum_{j=0}^{h-1} \phi^j w_{t-j}$$

$$\begin{aligned}\gamma(x_t, x_{t-h}) &= \text{cov}(x_t, x_{t-h}) = E(x_t x_{t-h}) = \\ E\{(\phi^h x_{t-h} + \sum_{j=0}^{h-1} \phi^j w_{t-j}) x_{t-h}\} &= \phi^h \text{var}(x_{t-h}) = \frac{\phi^h \sigma_w^2}{1-\phi^2}\end{aligned}$$

Hence,

$$\gamma(h) = \frac{\phi^h \sigma_w^2}{1 - \phi^2}$$

Also,

$$\rho(h) = \phi^h$$

Stationarity

Fact: sometimes $\rho(s, t)$ depends on lag $|s - t|$ only

Time series is **strictly stationary** if distributions of $\{x_{t1}, \dots, x_{tn}\}$ and $\{x_{t1+h}, \dots, x_{tn+h}\}$ are identical for any $\{t_1, \dots, t_n\}$ and all lags $h = 0, \pm 1, \pm 2, \dots$

$$P(x_{t1} \leq c_1, \dots, x_{tn} \leq c_n) = P(x_{t1+h} \leq c_1, \dots, x_{tn+h} \leq c_n)$$

Note: This means

- Mean function $\mu_t = E x_t = \text{const.}$
- Autocovariance $\gamma(t, t + h) = \text{function only of lag } h$

Stationarity

Strict stationarity is often too strong!

- Time series x_t is **weakly stationary (stationary)** if
 - ▶ $E x_t = \text{const}$
 - ▶ $\gamma(s, t) = \gamma(|s - t|)$
 - ▶ $\text{var}(x_t) < \infty$
- $\gamma(t, t + h) = \gamma(|t + h - t|) = \gamma(h)$
 - ▶ Autocovariance depends on lag only!
- Autocovariance for stationary process $\gamma(h) = \text{cov}(x_t, x_{t+h})$
- ACF for stationary process $\rho(h) = \frac{\gamma(h)}{\gamma(0)}$

Stationarity

Properties of stationary process:

$$\gamma(h) = \gamma(-h) \quad \rho(h) = \rho(-h)$$

$$|\gamma(h)| \leq \gamma(0) \quad \rho(h) \leq 1, \rho(0) = 1$$

Reflect: Are these processes stationary?

- White noise
- Moving average, $x_t = 0.2w_{t-1} + 0.5w_t + 0.2w_{t+1}$
- Random walk, $x_t = \delta t + \sum_{j=1}^t w_j$

Sample autocovariance and ACF

Dependence measures for samples?

- Idea: replace mean and covariance with sample estimates

If x_t is stationary,

- Sample mean

$$Ex \approx \bar{x} = \frac{1}{n} \sum_{t=1}^n x_t$$

- Sample autocovariance function

$$\hat{\gamma}(h) = \frac{1}{n} \sum_{t=1}^{n-h} (x_{t+h} - \bar{x})(x_t - \bar{x})$$

Sample autocovariance and ACF

Example: n=6, h=2

		X1	X2	X3	X4	X5	X6
X1	X2	X3	X4	X5	x6		

Sample autocorrelation function (sample ACF)

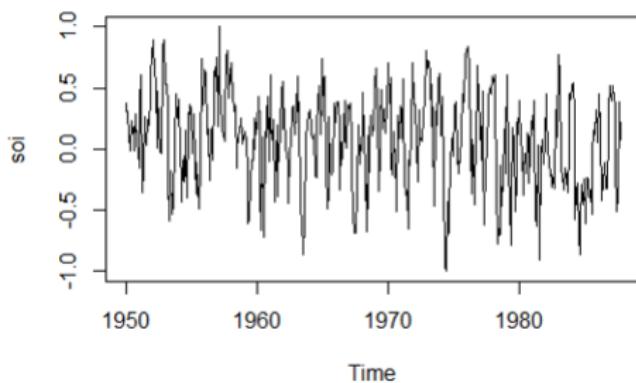
$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}$$

Sample ACF

In R: `acf()`

Example: southern oscillation index (SOI)

- `rho=acf(soi, 5, type="correlation", plot=T)`



```
> print(rho)
```

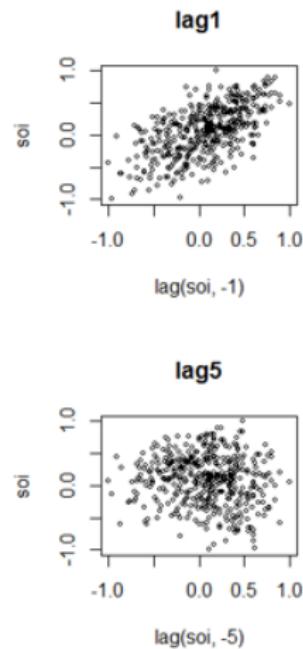
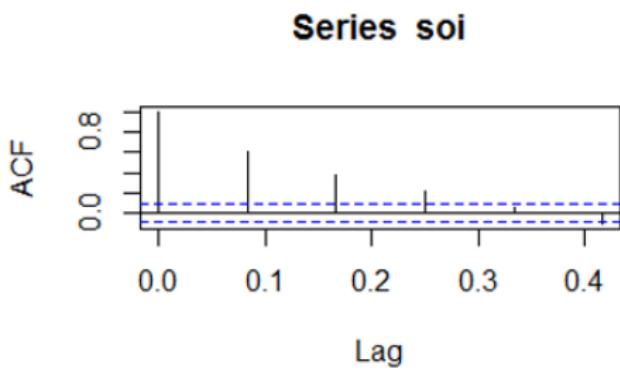
```
Autocorrelations of series 'soi', by lag
```

Lag	Autocorrelation
0	0.0000
1	0.0833
2	0.1667
3	0.2500
4	0.3333
5	0.4167

```
0.0000 0.0833 0.1667 0.2500 0.3333 0.4167  
1.0000 0.6040 0.3740 0.2140 0.0500 -0.1070
```

Why is sample ACF '1' for h=0?

Sample ACF



Sample ACF

What are these blue lines?

Theorem: Under weak conditions, if x_t is white noise and $n \rightarrow \infty$ then $\hat{\rho}(h)$ is approximately $N(0, \frac{1}{n})$

Consequence: If some $|\hat{\rho}(h)| > \frac{2}{\sqrt{n}}$ then the time series is not a white noise (with approximately 95 % confidence).

Typical modeling strategy:

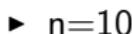
- Fit a model
- Compute residuals
- Check ACF within $\pm \frac{2}{\sqrt{n}}$

Sample ACF vs theoretical

- Moving average $x_t = 0.2w_{t-1} + 0.5w_t + 0.2w_{t+1}$



$$ACF\gamma(h) = \begin{cases} 1 & h = 0 \\ 0.61 & h = 1 \\ 0.12 & h = 2 \\ 0 & other \end{cases}$$



Autocorrelations of series 'y1', by lag

0	1	2	3	4	5
1.000	0.236	-0.399	-0.187	-0.008	-0.118



Autocorrelations of series 'y1', by lag

0	1	2	3	4	5
1.000	0.609	0.129	-0.007	0.001	0.044

⋮

Vector-valued time series

If $x_t = (x_{t1}, x_{t2}, \dots, x_{tp})'$ is stationary,

- mean vector is $\mu = E(x_t)$ and sample mean is its approximation

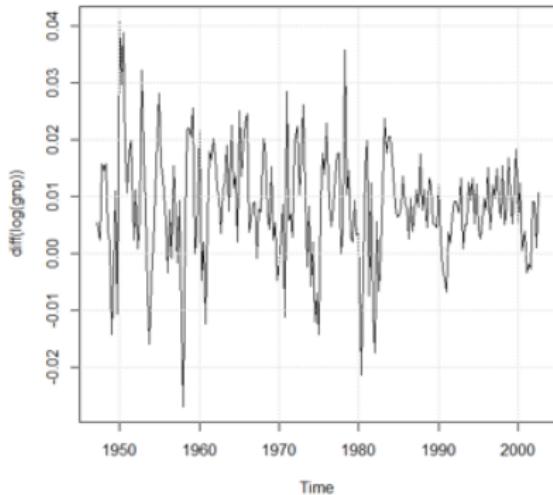
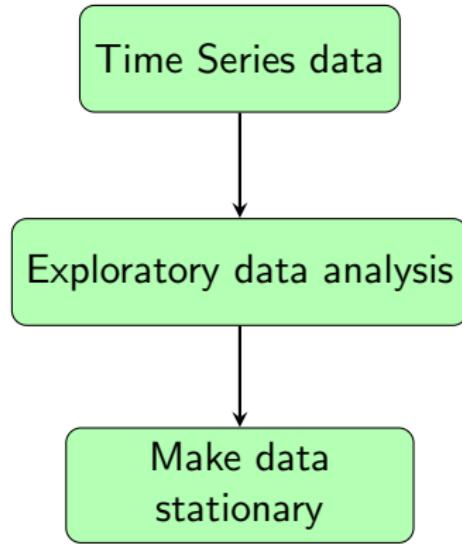
$$\mu = E(x_t) \approx \bar{x} = \frac{1}{n} \sum_{t=1}^n x_t$$

- Autocovariance function is $\Gamma(h) = E[(x_{t+h} - \mu)(x_t - \mu)']$ and sample autocovariance matrix

$$\hat{\Gamma}(h) = \frac{1}{n} \sum_{t=1}^{n-h} (x_{t+h} - \bar{x})(x_t - \bar{x})'$$

Recap: time domain modeling

$$Y_t = \nabla(\log(X_t))$$



Stationarity

- Why do we need stationarity?
 - ▶ Sample ACF becomes consistent
 - ▶ ARIMA models require stationarity

- Tools
 - ▶ Detrending (trend removal)
 - ▶ Differencing
 - ▶ Transformations

whiteboard

- Introduce linear regression/least squares
- Trend removal, simple drift

Trend removal by regression

Regressing on covariates

Given x_t (dependent series) and z_{t1}, \dots, z_{t2} (independent series) we model

$$x_t = \beta_0 + \beta_1 z_{t1} + \dots + \beta_q z_{tq} + w_t$$

where w_t is assumed white noise.

Note: w_t is seldom white noise in practice, used as a tool for detrending!

Trend removal by regression

Still a linear regression in β

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad Z = \begin{pmatrix} 1 & z_{11} & \dots & z_{1q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & z_{n1} & \dots & z_{nq} \end{pmatrix}$$

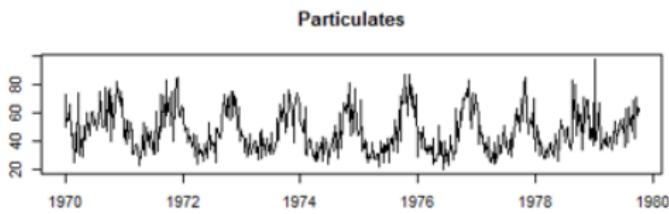
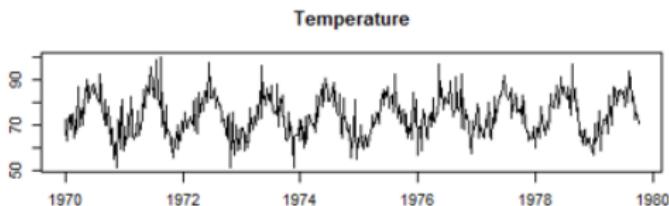
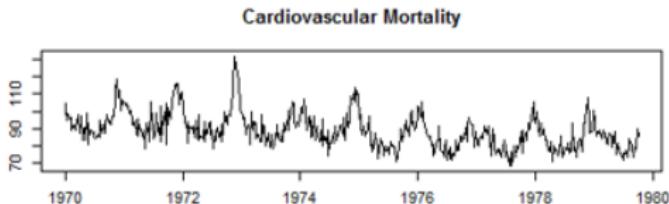
Least squares estimate is computed as

$$\hat{\beta} = (Z^T Z)^{-1} Z^T X$$

Trend removal

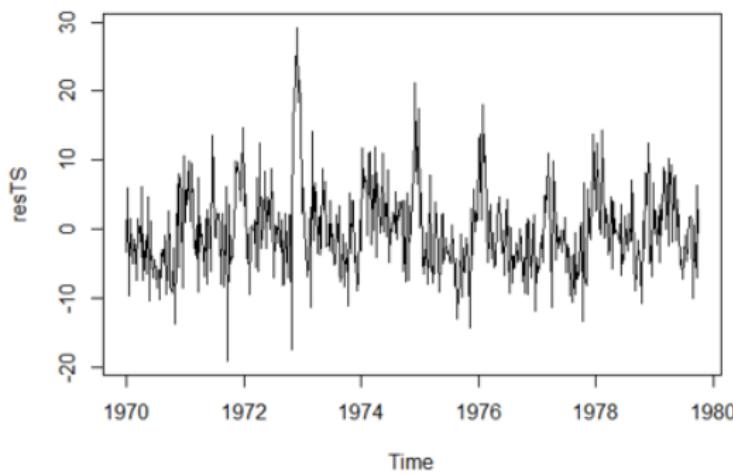
Example: Mortality

- x_t : Cardiovascular mortality
- z_{t1} : Temp (centered)
- z_{t2} : Temp (centered, squared)
- z_{t3} : Time
- z_{t4} : Levels of particles



Trend removal

- Residuals
 - ▶ Stationary?
 - ▶ Independent?
 - ▶ Some additional modeling of the residuals (ARIMA) can be done



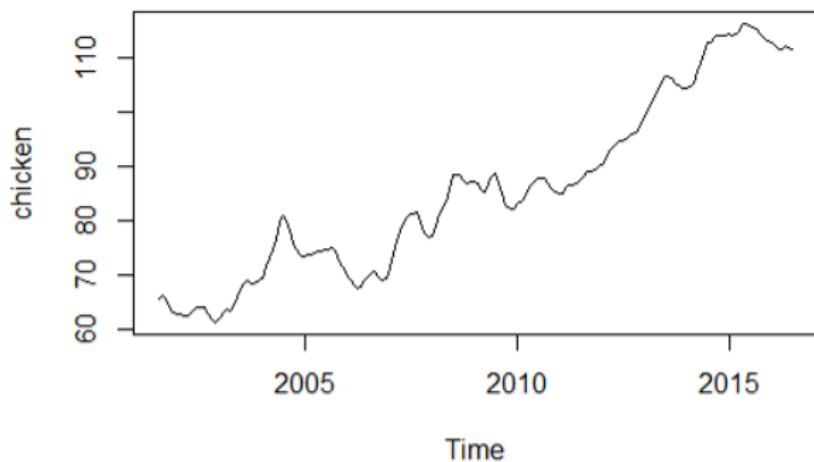
Differencing

Assume $x_t = \mu_t + y_t$, y_t stationary

Differencing gives $z_t = \nabla x_t = x_t - x_{t-1}$

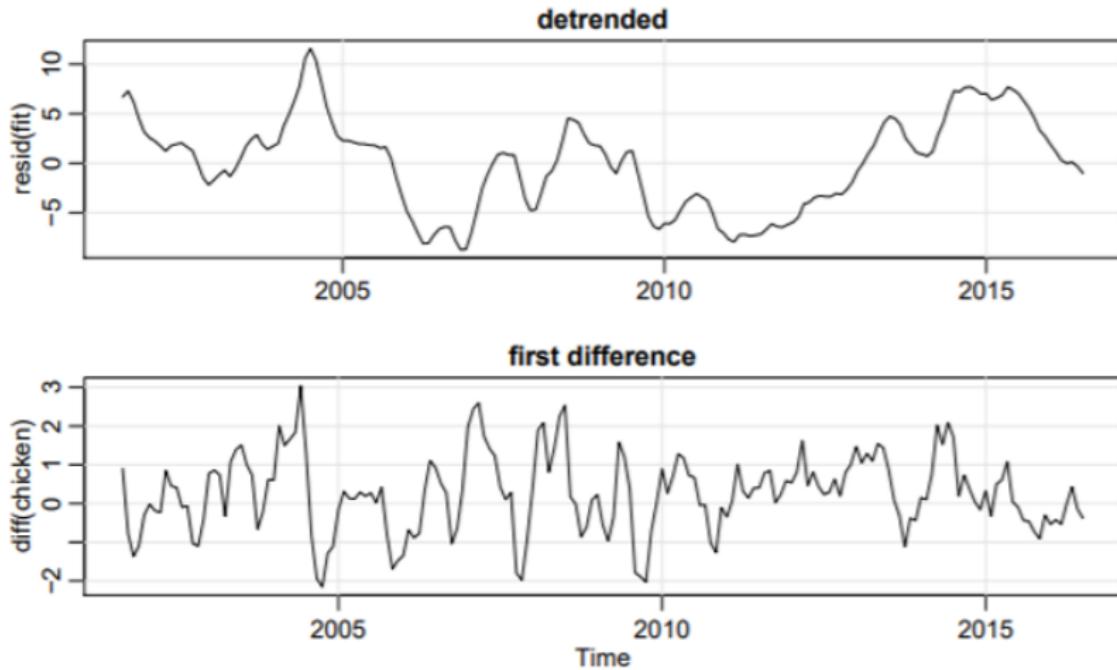
- **Property 1:** If $\mu_t = \alpha_0 + \alpha_1 t$ then z_t is stationary
- **Property 2:** If μ_t is random walk with a drift then z_t is stationary

Example:
Chicken prices

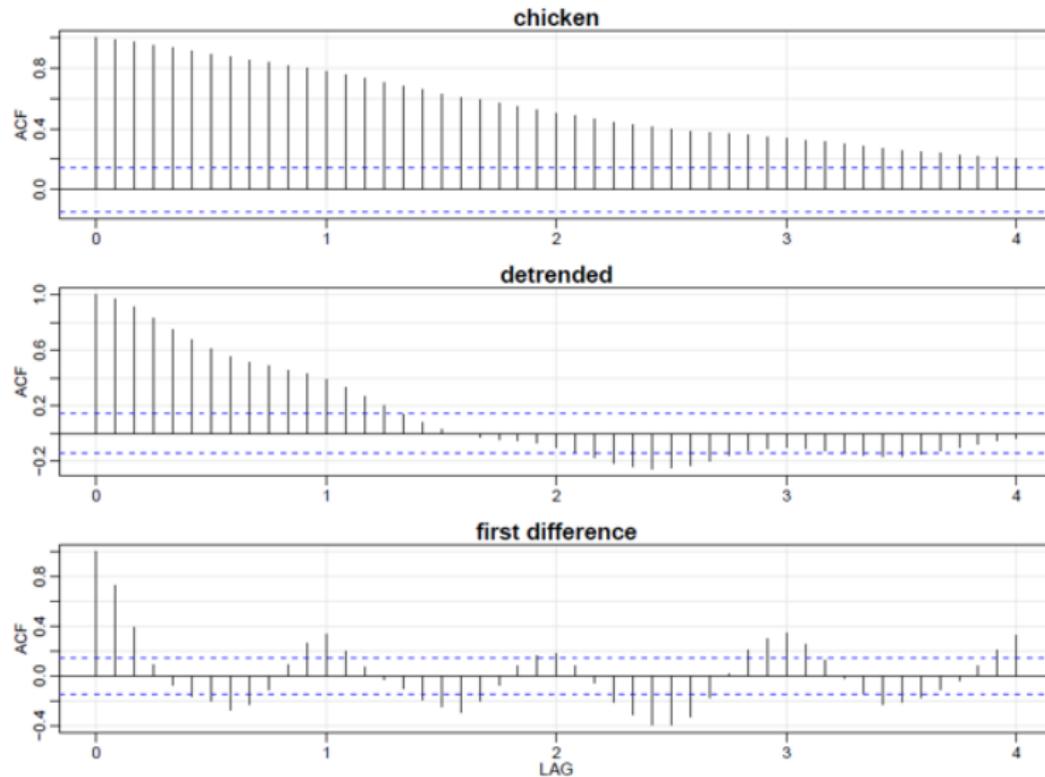


Differencing

Which looks **most random**? Other differences?



Differencing



Detrending vs differencing

- Differencing is more flexible than linear detrending
- Differencing does not require model estimation
- If trend is complex, detrending with a flexible (machine learning) model can be better
- Differencing does not give us the trend

Backshift operator

- Backshift operator $Bx_t = x_{t-1}$, Powers $B^k x_t = x_{t-k}$
- Forward-shift operator $B^{-1}x_t = x_{t+1}$
- Note $BB^{-1}x_t = x_t$ (i.e. $BB^{-1} = 1$)
- Differencing $\nabla x_t = (1 - B)x_t$
- Differences of order d : $\nabla^d = (1 - B)^d$
- Property: Operators can be manipulated as polynomials
- Example Check that $\nabla^2 x_t = x_t - 2x_{t-1} + x_{t-2}$
- Property: Differencing of order p can remove polynomial trend of order p

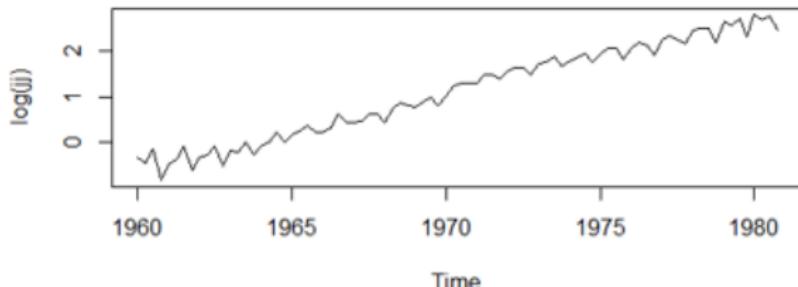
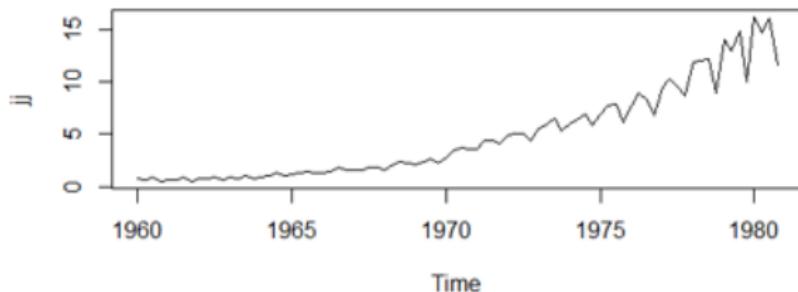
Transformations

- Often used to stabilize variance
 - ▶ If for $\text{ex.var}(x_t) \neq \text{var}(x_s)$ then time series is non-stationary ...
- Sometimes makes data more similar to normal distr.
- Common transforms:
 - ▶ $z_t = \log(x_t)$
 - ▶ Power transformation

$$z_t = \begin{cases} \frac{(x_t^\lambda - 1)}{\lambda} & \lambda \neq 0 \\ \log(x_t) & \lambda = 0 \end{cases}$$

Transformations

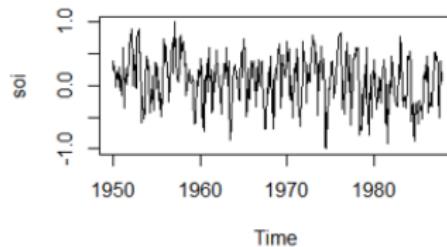
- Johnson & Johnson quarterly earnings



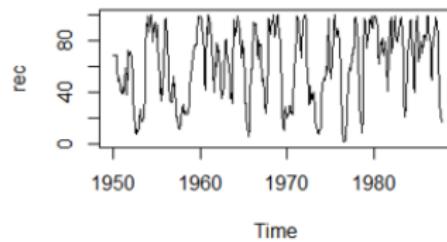
Scatterplots

- Plot x_t vs z_{t_i} or z_{t_i} vs z_{t_j}
- Exploratory tool: indicates which relationship to model

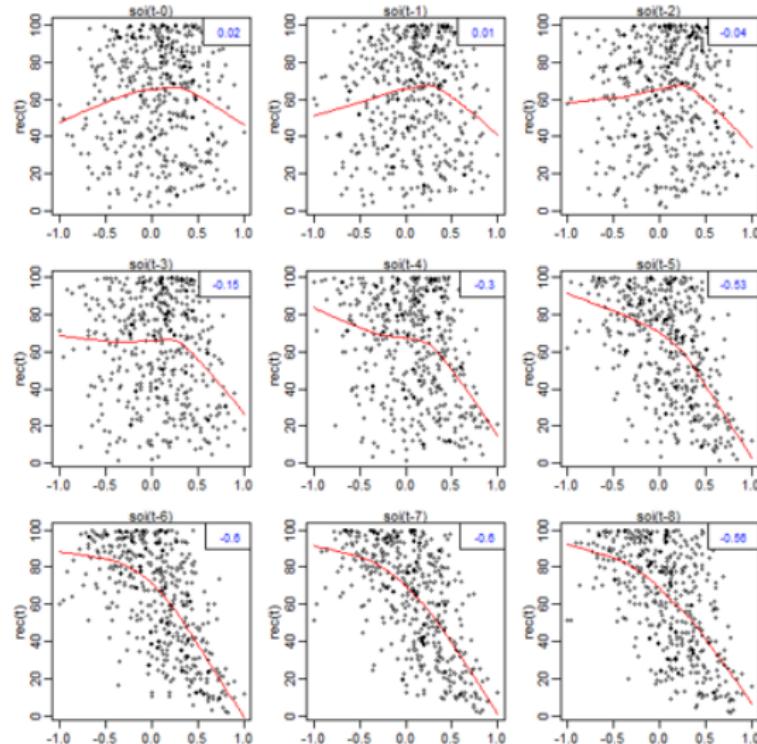
$$x_t = f(z_{t_1}, z_{t_2}, \dots, z_{t_q}) + w_t$$



- Example: SOI and Recruitment



Scatterplots



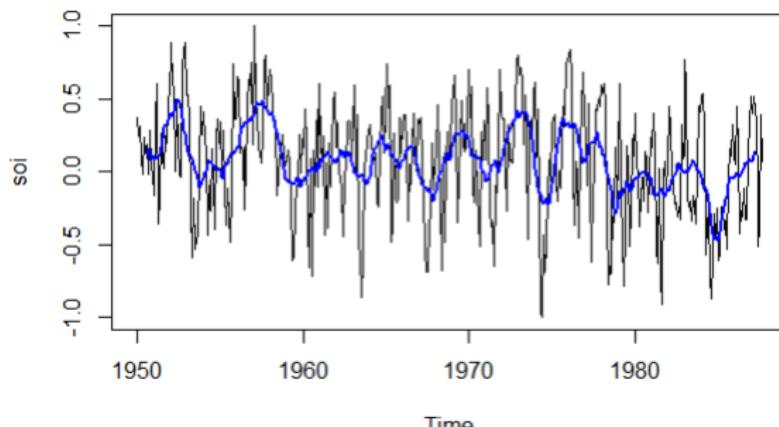
- Which relationships are nonlinear?
- Conclusion:
include dummy variables
 $I(\text{soil}(t - j) > 0)$ in the linear model

Smoothing

- Moving average smoother

$$m_t = \sum_{j=-k}^{j=k} a_j x_{t-j}$$

- Where $\sum_{j=-k}^{j=k} a_j = 1$ and $a_j = a_{-j} \geq 0$,
- Example: SOI data Disadvantage?



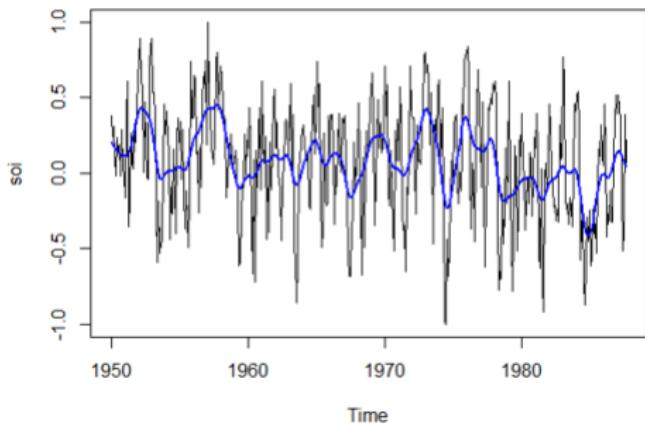
Smoothing

More flexible models?

- Splines
- Kernel smoothers
- Gaussian Process
- Neural networks
- ...

Welcome to ML courses!!

Example: kernel smoothers



Home reading

- Shumway and Stoffer, sections 1.4-1.6 and chapter 2
- TS functions: lag, ksmooth, lm, diff, lag1.plot, lag2.plot

Time Series Analysis

Lecture 3: Introduction to ARIMA

Tohid Ardeshtiri

Linköping University
Division of Statistics and Machine Learning

September 6, 2019



Recap

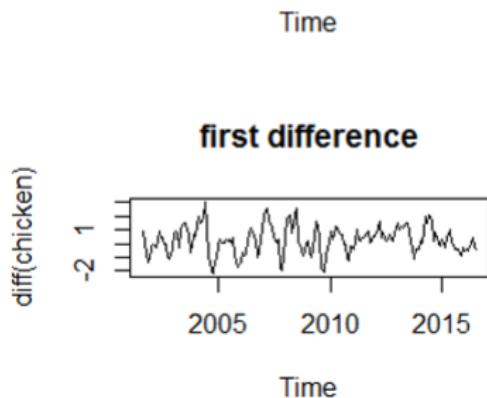
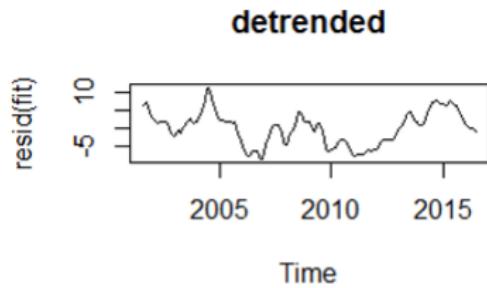
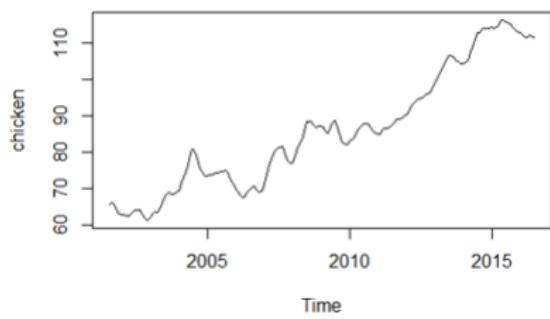
How to make data stationary?

- Transformations (log, other)
- Detrending
 - ▶ Differencing
 - ▶ Linear regression
 - ▶ Kernel smoother
 - ▶ ...

How shall we model the data after detrending and transformations
(residuals)? →?ARIMA models!

ARIMA models

- Why ARIMA models?
 - ▶ Removing trend is not sufficient



Moving average models

- Moving average model of order q, MA(q)

$$\begin{aligned}x_t &= w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q} \\&= \sum_{j=0}^q \theta_j w_{t-j}\end{aligned}$$

- ▶ $w_t \sim wn(0, \sigma_w^2)$
- ▶ $\theta_1, \dots, \theta_q$ constants, $\theta_q \neq 0$ and $\theta_0 = 1$

- Moving average operator

$$\theta(B) = \sum_{j=0}^q \theta_j B^j$$

- MA(q): $x_t = \theta(B)w_t$

Linear process

x_t is a **linear process** if

$$x_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j w_{t-j}$$

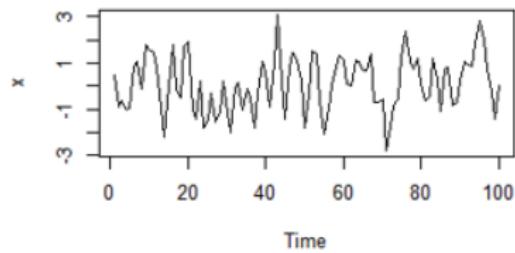
Property: It can be shown that

$$\gamma_x(h) = \sigma_w^2 \sum_{j=-\infty}^{\infty} \psi_{j+h} \psi_j$$

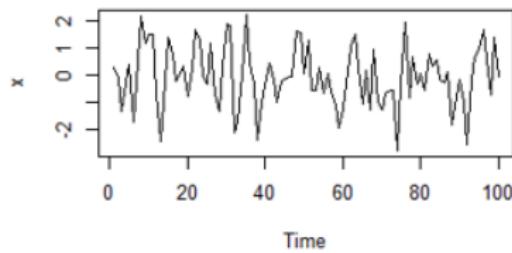
Example: MA(1)

$$x_t = w_t + \theta w_{t-1}$$

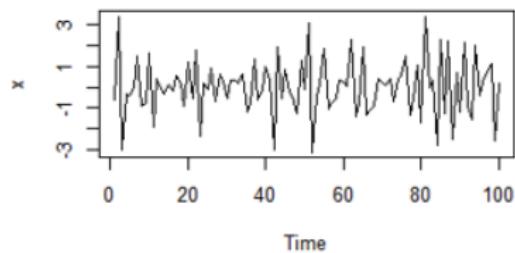
$$\theta = 0.9$$



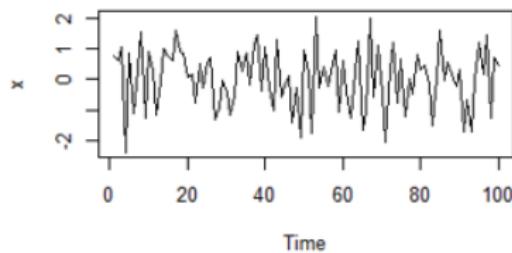
$$\theta = 0.2$$



$$\theta = -0.9$$



$$\theta = -0.2$$



Example: MA(1)

$$x_t = w_t + \theta w_{t-1}$$

- Autocovariance and ACF

$$\gamma(h) = \begin{cases} (1 + \theta^2)\sigma_w^2 & h = 0 \\ \theta\sigma_w^2 & h = 1 \\ 0 & h > 1 \end{cases}$$

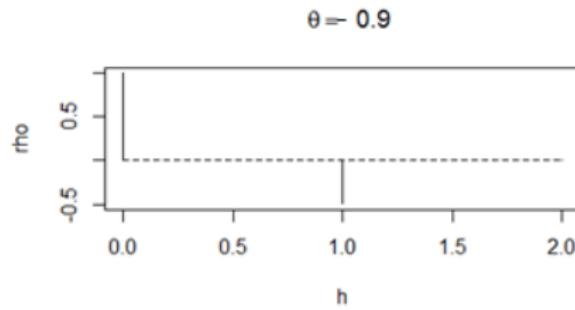
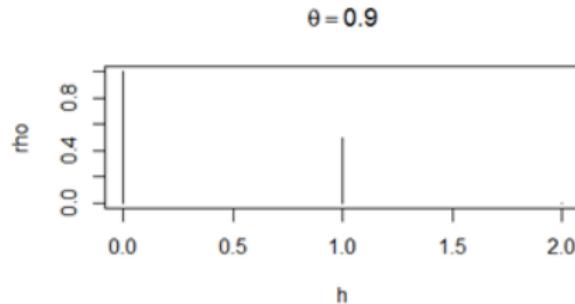
$$\rho(h) = \begin{cases} \frac{\theta}{1+\theta^2} & h = 1 \\ 0 & h > 1 \end{cases}$$

Note: $\rho(0) = 1$ is often not written as it is trivial.

- Process is stationary

Example: MA(1)

- Note: $\rho(0) = 1$ is often not shown \rightarrow only 1 bar



AR models

- Autoregressive model of order p , $AR(p)$

$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t$$

- ▶ x_t is stationary if x_0 is sampled from the stationary distribution
 - ▶ $w_t \sim \text{wn}(0, \sigma_w^2)$
 - ▶ ϕ_1, \dots, ϕ_p constants, $\phi_p \neq 0$
 - ▶ $E x_t = 0$
-
- Note: if $E x_t = \mu \neq 0$, model $x'_t = x_t - \mu$

AR models

Another form

- **Autoregressive operator**

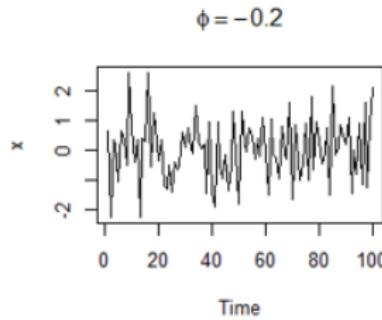
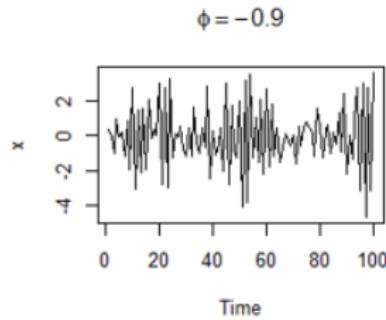
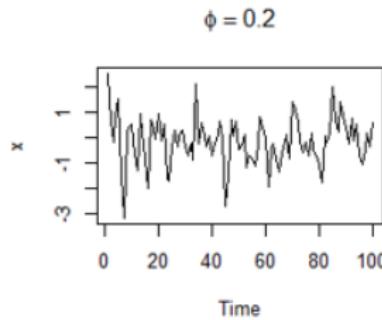
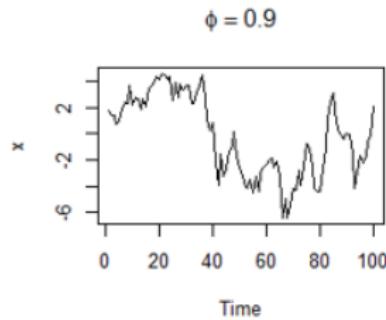
$$\phi(B) = 1 - \phi_1 B - \dots - \phi_p B^p$$

- AR(p) model

$$\boxed{\phi(B)x_t = w_t}$$

Example: AR(1)

- How do these plots differ? $x_t = \phi x_{t-1} + w_t$



Ar(1) (read at home)

$$x_t = \phi x_{t-1} + w_t$$

Mean function:

$$Ex_t = \phi Ex_{t-1} + Ew_t = \phi Ex_{t-1} = \phi(\phi Ex_{t-2}) = \dots = \phi^t Ex_0$$

for $Ex_0 = 0$, $Ex_t = 0$ for all t .

Variance $\text{var}(x_t)$ when $Ex_0 = 0$ and w_t is uncorrelated with x_0 for all t :

$$\begin{aligned}\text{var}(x_t) &= E\{(x_t - 0)^2\} = E\{\phi^2 x_{t-1}^2 + 2\phi x_{t-1} w_t + w_t^2\} = \\ \phi^2 \text{var}(x_{t-1}) + 2\phi \text{cov}(x_{t-1}, w_t) + \text{var}(w_t) &= \phi^2 \text{var}(x_{t-1}) + \text{var}(w_t) = \\ \phi^2 \text{var}(x_{t-1}) + \sigma_w^2 &= \phi^2(\phi^2 \text{var}(x_{t-2}) + \sigma_w^2) + \sigma_w^2 = \\ \phi^{2t} \text{var}(x_0) + \sigma_w^2 \sum_{k=0}^{t-1} (\phi^{2k}) &= \phi^{2t} \text{var}(x_0) + \frac{\sigma_w^2(1-\phi^{2t})}{1-\phi^2}\end{aligned}$$

When $\text{var}(x_0) = \frac{\sigma_w^2}{1-\phi^2}$ then $\text{var}(x_t) = \frac{\sigma_w^2}{1-\phi^2}$ and time independent.

A(1) (read at home)

$$x_t = \phi x_{t-1} + w_t$$

$$x_t = \phi(\phi x_{t-2} + w_{t-1}) + w_t = \dots = \phi^h x_{t-h} + \sum_{j=0}^{h-1} \phi^j w_{t-j}$$

$$\begin{aligned}\gamma(x_t, x_{t-h}) &= \text{cov}(x_t, x_{t-h}) = E(x_t x_{t-h}) = \\ E\{(\phi^h x_{t-h} + \sum_{j=0}^{h-1} \phi^j w_{t-j}) x_{t-h}\} &= \phi^h \text{var}(x_{t-h}) = \frac{\phi^h \sigma_w^2}{1-\phi^2}\end{aligned}$$

Hence,

$$\gamma(h) = \frac{\phi^h \sigma_w^2}{1 - \phi^2}$$

Also,

$$\rho(h) = \phi^h$$

- **Property:** If $|\phi| < 1$ and $\sup \text{var}(x_t) < \infty$

$$x_t = \sum_{j=0}^{\infty} \phi^j w_{t-j}$$

- Show it by
 - ▶ Substitution
 - ▶ Taylor expansion
 - ▶ Coefficient matching
- Autocovariance and ACF

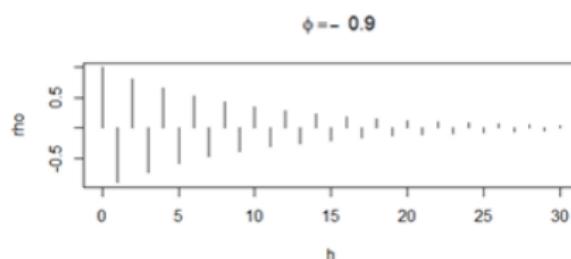
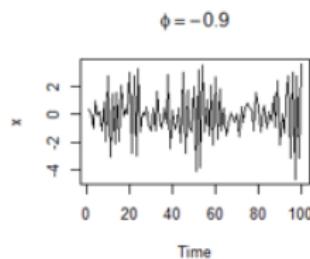
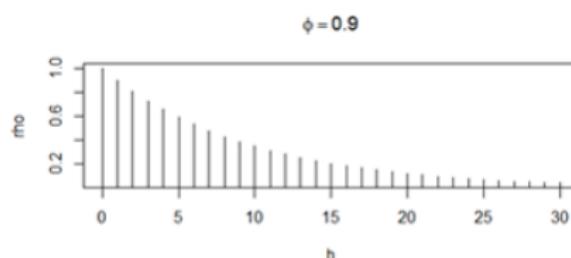
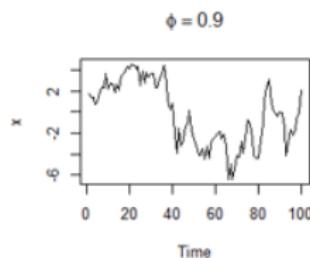
$$\gamma(h) = \frac{\sigma_w^2 \phi^h}{1 - \phi^2} \quad \rho(h) = \phi^h$$

for $h \geq 0$.

Example: AR(1)

Autocovariance and ACF (for $h \geq 0$)

$$\gamma(h) = \frac{\sigma_w^2 \phi^h}{1 - \phi^2} \quad \rho(h) = \phi^h$$



Explosive AR models

- **Explosive** =series become arbitrarily large in magnitude
- AR(1): What if $|\phi| > 1$?
 - ▶ $x_t = \phi^p x_{t-p} + \sum_{j=0}^{p-1} \phi^p w_{t-j} \rightarrow$ grows exponentially
 - ▶ **Stationary?** Check variance
- Can we make it stationary?
$$x_t = \phi^{-1} x_{t+1} - \phi^{-1} w_{t+1} = \phi' x_{t+1} + w'_t$$
 - ▶ Stationary, but dependent on the future
 - ▶ $w'_t \sim N(0, \phi^{-2} \sigma_w^2)$
 - ▶ $x_t = - \sum_{j=1}^{\infty} \phi^{-j} w_{t+j}$

Causal process

A stationary process is **causal** if it is only dependent on the past values of the process

Def: A linear process is **nonexplosive** and **causal** if it can be written as a one-sided sum:

$$x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j} = \psi(B)w_t$$

where $\psi(B) = \sum_{j=0}^{\infty} \psi_j B^j$ and $\sum_{j=0}^{\infty} |\psi_j| < \infty$.

$$\rho(h) = \begin{cases} \frac{\theta}{1+\theta^2} & h = 1 \\ 0 & h > 1 \end{cases}$$

Note: MA(1) gives equivalent models for $\theta = s$ and $\theta = \frac{1}{s}$

Probabilistic expressions equivalent: ACF identical

→ we can not distinguish between these models

Invertibility of MA

Def: An MA process is **invertible** if it has a causal AR representation,

$$w_t = \sum_{j=0}^{\infty} \pi_j x_{t-j}$$

Example: MA(1) with $\theta = 1/5$ is invertible, $\theta = 5$ not.

ARMA models

- Autoregressive moving average ARMA(p,q)

$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q}$$

- ▶ $\phi_p \neq 0, \theta_q \neq 0$
- ▶ Is stationary
- ▶ $E x_t = 0$
- p -autoregressive order, q -moving average order
- Alternative form

$$\phi(B)x_t = \theta(B)w_t$$

- Note: $x_t = \phi^{-1}(B)\theta(B)w_t = \sum_{j=-\infty}^{\infty} \psi_j w_{t-j}$
 - ▶ But series might be non-convergent

Parameter redundancy

Note: we can multiply both sides with $\eta(B)$

$$\eta(B)\phi(B)x_t = \eta(B)\theta(B)w_t$$

- The resulting model looks different (higher orders)
- Underlying model is actually the same

Example: $x_t = w_t$, white noise. Let $\eta(B) = 1 - 0.5B$.

We get

$$x_t - 0.5x_{t-1} = w_t - 0.5w_{t-1}$$

Looks like ARMA(1,1)!

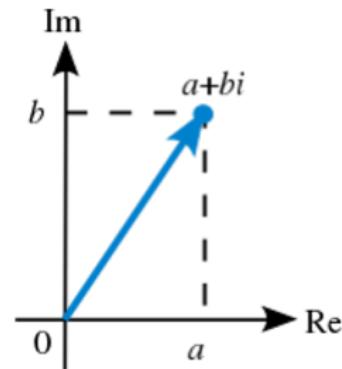
Reminder: complex numbers

- Imaginary unit $i^2 = -1$
- Complex number $z = a + ib$
- Conjugate $\bar{z} = a - ib$

- Absolute value $|z|^2 = z\bar{z} = a^2 + b^2$
- Trigonometric form
$$z = r(\cos(\theta) + i \sin(\theta))$$
- Eulers formula $e^{i\theta} = \cos(\theta) + i \sin(\theta)$

- Therefore

$$\cos(\theta) = \frac{e^{i\theta} + e^{-i\theta}}{2}$$



$$\sin(\theta) = \frac{e^{i\theta} - e^{-i\theta}}{2i}$$

Reminder: polynomials

- Any polynomial $P_r(x)$ of degree r can be written as

$$P_r(x) = a(x - z_1)\dots(x - z_r)$$

- where z_i are roots (real or complex)
- If z_i is a root, \bar{z}_i is also a root

Causal ARMA

Def: Linear process is **causal** and **nonexplosive** if

- $x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j}$ (depends on the past only)
- $\sum_{j=0}^{\infty} |\psi_j| < \infty$
- We set $\psi_0 = 1$ by convention.

Property: ARMA(p,q) is **causal** iff roots $\phi(z') = 0$ are outside unit circle,
i.e. $|z'| > 1$

$$\phi(B)x_t = \theta(B)w_t$$

Causal ARMA

Example: Is the ARMA process below causal?

$$x_t = 0.4x_{t-1} + 0.3x_{t-2} + 0.2x_{t-3} + w_t - 0.1w_{t-1}$$
$$\Rightarrow \phi(B) = 1 - 0.4B - 0.3B^2 - 0.2B^3$$

```
> z=c(1, -0.4,-0.3,-0.2)
> polyroot(z)
[1] 1.060419-0.000000i -1.280210+1.753904i -1.280210-1.753904i
>
```

Invertible ARMA

Def: ARMA(p,q) is **invertible** if

- $w_t = \sum_{j=0}^{\infty} \pi_j x_{t-j}$ (depends on the past only)
- $\sum_{j=0}^{\infty} |\pi_j| < \infty$

Property: ARMA(p,q) is **invertible** iff roots $\theta(z') = 0$ are outside unit circle, i.e. $|z'| > 1$

$$\phi(B)x_t = \theta(B)w_t$$

- $x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j} \rightarrow x_t = \psi(B)w_t$
- $w_t = \sum_{j=0}^{\infty} \pi_j w_{t-j} \rightarrow w_t = \pi(B)x_t$
- How to find coefficients in ψ and π → coefficient matching

$$\phi(z)\psi(z) = \theta(z) \quad \pi(z)\theta(z) = \phi(z)$$

- Example: $x_t = 0.4x_{t-1} + 0.45x_{t-2} + w_t + w_{t-1} + 0.25w_{t-2}$

```
> ARMAtoMA(ar=.9,ma=0.5, 6)
[1] 1.400000 1.260000 1.134000 1.020600 0.918540 0.826686
```

Home reading

- Shumway and Stoffer, section 3.1
- R code: arima.sim, arima, polyroot, ARMAtoMA, ARMAacf
 - ▶ Check carefully arima() docs to see how ar and ma coefficients are specified in the software

Time Series Analysis

Lecture 4: ARIMA models-1, Estimation

Tohid Ardestiri

Linköping University
Division of Statistics and Machine Learning

September 13, 2019



White noise

Simplest and most random time series: **white noise**

- w_t uncorrelated $E(w_t w_{t-h}) = 0$ for all $h \neq 0$

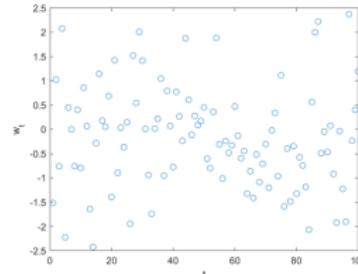
$$w_t \sim wn(0, \sigma_w^2)$$

- w_t white independent noise: independent and identically distributed
independence: $f(w_t, w_{t-h}) = f(w_t)f(w_{t-h})$

$$w_t \sim iid(0, \sigma_w^2)$$

- w_t white normal noise: independent and identically normal distributed

$$w_t \sim iidN(0, \sigma_w^2)$$



Autocovariance and ACF

- Autocovariance function

$$\gamma(s, t) = \text{cov}(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)]$$

Note $\text{var}(x_t) = \gamma(t, t)$

- Autocorrelation function (ACF)

$$\rho(s, t) = \frac{\gamma(s, t)}{\sqrt{\gamma(s, s)\gamma(t, t)}}$$

Useful fact: If $U = \sum_{j=1}^m a_j x_j$ and

$$V = \sum_{k=1}^r b_k y_k$$

$$\text{cov}(U, V) = \sum_{j=1}^m \sum_{k=1}^r a_j b_k \text{cov}(x_j, y_k)$$

Stationarity

- Time series x_t is **weakly stationary (stationary)** if
 - ▶ $E x_t = \text{const}$
 - ▶ $\gamma(s, t) = \gamma(|s - t|)$
 - ▶ $\text{var}(x_t) < \infty$
- $\gamma(t, t + h) = \gamma(|t + h - t|) = \gamma(h)$
 - ▶ Autocovariance depends on lag only!
- Autocovariance for stationary process $\gamma(h) = \text{cov}(x_t, x_{t+h})$
- ACF for stationary process $\rho(h) = \frac{\gamma(h)}{\gamma(0)}$

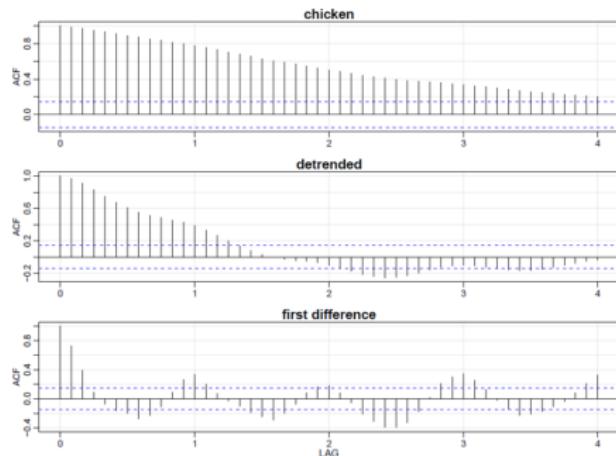
Sample ACF

Theorem: Under weak conditions, if x_t is white noise and $n \rightarrow \infty$ then $\hat{\rho}(h)$ is approximately $N(0, \frac{1}{n})$

Consequence: If some $|\hat{\rho}(h)| > \frac{2}{\sqrt{n}}$ then the time series is not a white noise (with approximately 95 % confidence).

Typical modeling strategy:

- Propose a model
- Fit a model
- Compute residuals
- Check ACF within $\pm \frac{2}{\sqrt{n}}$



Moving average models

- Moving average model of order q, MA(q)

$$1 \quad x_t = 1 w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q}$$

$$x_t = \sum_{j=0}^q \theta_j w_{t-j}$$

- ▶ $w_t \sim wn(0, \sigma_w^2)$
- ▶ $\theta_1, \dots, \theta_q$ constants, $\theta_q \neq 0$ and $\theta_0 = 1$

- Moving average operator

$$\theta(B) = \sum_{j=0}^q \theta_j B^j$$

- MA(q):

$$x_t = \theta(B)w_t$$

Autoregressive models

- Autoregressive model of order p , $AR(p)$

$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t$$

$$x_t - \sum_{j=1}^p \phi_j x_{t-j} = w_t$$

- ▶ x_t is stationary if x_0 is sampled from the stationary distribution
- ▶ $w_t \sim \text{wn}(0, \sigma_w^2)$
- ▶ ϕ_1, \dots, ϕ_p constants, $\phi_p \neq 0$
- ▶ $E x_t = 0$ if $E x_0 = 0$

- Autoregressive operator

$$\phi(B) = 1 - \phi_1 B - \dots - \phi_p B^p$$

- AR(p) model

$$\boxed{\phi(B)x_t = w_t}$$

ARMA models

- Autoregressive moving average ARMA(p,q)

$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q}$$

- ▶ $\phi_p \neq 0, \theta_q \neq 0$
- ▶ Is stationary
- ▶ $E x_t = 0$ if $E x_0 = 0$

- p -autoregressive order, q -moving average order
- Alternative form

$$\phi(B)x_t = \theta(B)w_t$$

- Criteria for **causality** and **invertibility**

- ▶ Check roots of the characteristic polynomials $\phi(\cdot)$ and $\theta(\cdot)$

Property: ARMA(p,q) is **causal** iff **ALL** roots $\phi(z') = 0$ are outside unit circle, i.e. $|z'| > 1$

Property: ARMA(p,q) is **invertible** iff **ALL** roots $\theta(z') = 0$ are outside unit circle, i.e. $|z'| > 1$

Linear process

For a **linear process** x_t : $x_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j w_{t-j} = \mu + \psi(B)w_t$
where $\psi(B) = \sum_{j=0}^{\infty} \psi_j B^j$,

$$\gamma_x(h) = \sigma_w^2 \sum_{j=-\infty}^{\infty} \psi_{j+h} \psi_j$$

Note: $x_t = \phi^{-1}(B)\theta(B)w_t = \sum_{j=-\infty}^{\infty} \psi_j w_{t-j}$ But series might be non-convergent

- Coefficient matching **whiteboard**
- How to find coefficients in $\psi(B)$ → **coefficient matching**
- **Example:** $x_t = 0.4x_{t-1} + 0.45x_{t-2} + w_t + w_{t-1} + 0.25w_{t-2}$

```
> ARMAtoMA(ar=.9,ma=0.5, 6)
[1] 1.400000 1.260000 1.134000 1.020600 0.918540 0.826686
```

Differencing

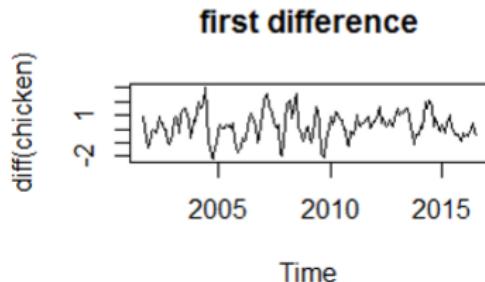
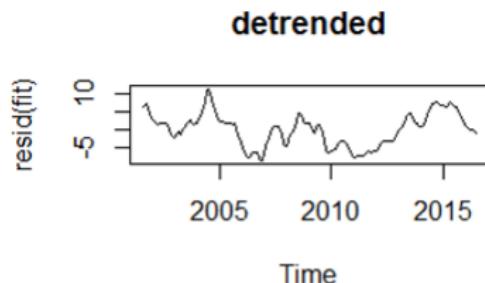
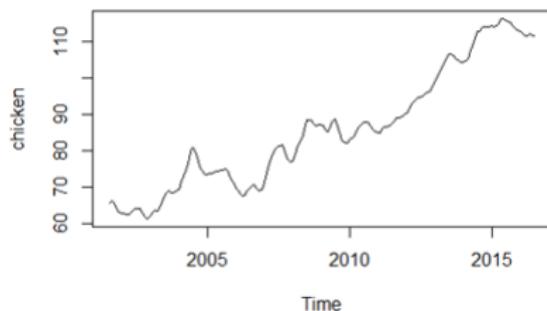
Assume $x_t = \mu_t + y_t$ and y_t stationary

Differencing gives

$$z_t = \nabla x_t = x_t - x_{t-1}$$

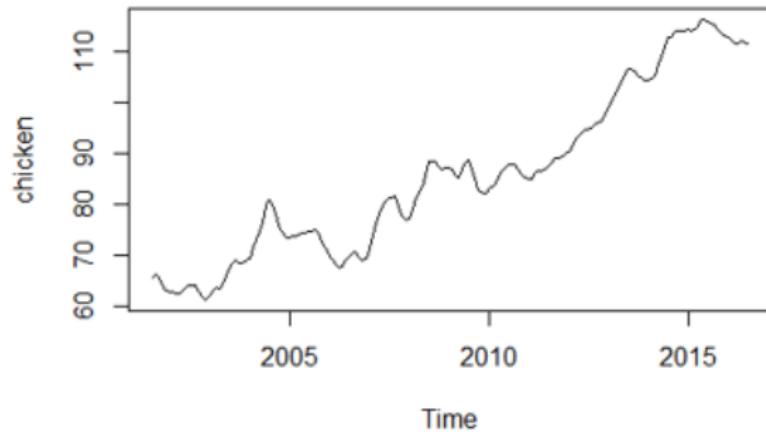
Also,

- $\nabla x_t = (1 - B)x_t$
- $\nabla^d = (1 - B)^d$



ARIMA models

- ARMA for stationary models
 - ▶ What if not stationary?



ARIMA models

- Differencing helps (lecture 2)
 - ▶ $\nabla x_t = x_t - x_{t-1}$ removes linear trend and random walk
 - ▶ $\nabla^d x_t$ removes polynomial of order d and some stochastic trends
 - ▶ → differencing is important modeling instrument!
- **Def:** x_t is ARIMA(p,d,q) if $\nabla^d x_t$ is ARMA(p,q), i.e.

$$\phi(B)(1 - B)^d x_t = \theta(B)w_t$$

- For nonzero mean $E(\nabla^d x_t) = \mu$,

$$\phi(B)(1 - B)^d x_t = \theta(B)w_t + \delta$$

$$\delta = \mu(1 - \phi_1 - \dots - \phi_p)$$

ARIMA models

- Notation: $p=0 \rightarrow \text{IMA}(d,q)$, $q=0 \rightarrow \text{ARI}(p,d)$
- Estimation: Differentiate + fit ARMA
- Forecasting:
 - ▶ Transform data $y_t = \nabla^d x_t$ and forecast ARMA(p,q)
 - ▶ Solve $(1 - B)^d x_t^n = y_t^n$

Estimation

Consider **ARIMA(p,d,q)**

$$\phi(B)(1 - B)^d x_t = \theta(B)w_t$$

- What are the unknowns?
 - ▶ Orders p , d and q
 - ▶ Parameters ϕ_1, \dots, ϕ_p and $\theta_1, \dots, \theta_q$
 - ▶ variance σ_w^2 where $w_t \sim N(0, \sigma_w^2)$
- How to estimate these?
- Assumption: Let us assume for now that we know p , d and q
 - ▶ Maximum likelihood (ML) estimate
 - ▶ Least squares

Maximum likelihood estimation: reminder

Let $x \sim f(x|\alpha)$

- Likelihood of α given observations x_1, \dots, x_t is

$$L(\alpha) = f(x_1, \dots, x_t | \alpha)$$

- Maximum likelihood: Optimal α

$$\hat{\alpha} = \arg \max_{\alpha} L(\alpha)$$

- Independent observations: $x_i \stackrel{iid}{\sim} f(x_i | \alpha)$
- $L(\alpha) = \prod_i f(x_i | \alpha)$
- Negative log-likelihood $I(\alpha) = -\sum_i \log(f(x_i | \alpha))$
- Maximum likelihood α can be obtained from negative log-likelihood

$$\max_{\alpha} L(\alpha) = \min_{\alpha} I(\alpha)$$

Maximum likelihood estimation: reminder

Time series data are NOT independent

- Likelihood of α given observations x_1, \dots, x_t is

$$L(\alpha) = f(x_1, \dots, x_t | \alpha)$$

- Maximum likelihood: Optimal α

$$\hat{\alpha} = \arg \max_{\alpha} L(\alpha)$$

- Dependent data (time series): chain rule

$$L(\alpha) = f(x_1 | \alpha) f(x_2 | \alpha, x_1) f(x_3 | \alpha, x_2, x_1) \dots$$

- Negative log-likelihood $I(\alpha) = - \sum_i \log(f(x_i | \alpha, x_{i-1}, \dots))$
- Maximum likelihood: Optimal α

$$\max_{\alpha} L(\alpha) = \min_{\alpha} I(\alpha)$$

Maximum likelihood estimation: reminder

- Normal distributions: if $x_i \sim N(\mu, \sigma^2)$, iid.

$$L(\theta) = \frac{1}{(\sqrt{2\pi}\sigma)^n} e^{-\frac{\sum_i(x_i-\mu)^2}{2\sigma^2}}$$

- Maximum likelihood

$$\hat{\mu} = \bar{x}$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_i (x_i - \bar{x})^2$$

- For ARMA models, assume normality of w_t !
- Negative log-likelihood

$$I(\mu, \phi, \sigma_w^2) = \frac{S(\mu, \phi)}{2\sigma_w^2} + \frac{n}{2} \log(2\pi\sigma_w^2) - \frac{1}{2} \log(1 - \phi^2)$$

$$S(\mu, \phi) = (1 - \phi^2)(x_1 - \mu)^2 + \sum_{t=2}^n [(x_t - \mu) - \phi(x_{t-1} - \mu)]^2$$

- How to find optimum?

- For σ^2 explicit

$$\hat{\sigma}_w^2 = \frac{1}{n} S(\hat{\mu}, \hat{\phi})$$

- Otherwise numerical optimization (unconstrained optimization)

Optimization methods

- Examples:
 - ▶ Steepest descent
 - ▶ Newtons Methods
 - ▶ Gauss-Newton methods
 - ▶ (least squares)
 - ▶ ...

Least squares

- **Unconditional least squares**

- Estimate by numerical methods or sometimes analytically

$$\min_{\mu, \phi} S(\mu, \phi)$$

- **Conditional least squares:** assume x_1 given (constant)

$$\min \sum_{i=1}^t w_i^2$$

- For AR(1), $\sum_{i=1}^t w_i^2 = S_c(\mu, \phi)$

$$S_c(\mu, \phi) = \sum_{t=2}^n [(x_t - \mu) - \phi(x_{t-1} - \mu)]^2 = \sum_{t=2}^n [x_t - \alpha - \phi x_{t-1}]^2$$

- **Note:** Minimize by doing regression $Y = x_t, X = \text{lag}(x_t)$

Home reading

- Shumway and Stoffer, parts of sections 3.5, 3.6, 3.7
- R code: arima.sim, arima, polyroot, ARMAtoMA, ARMAacf

Time Series Analysis

Lecture 5: ARIMA models-2

Estimation, PACF, Forecasting

Tohid Ardesthiri

Linköping University
Division of Statistics and Machine Learning

September 16, 2019



Maximum likelihood estimation: reminder

Time series data are NOT independent

- Likelihood of α given observations x_1, \dots, x_t is

$$L(\alpha) = f(x_1, \dots, x_t | \alpha)$$

- Maximum likelihood:** Optimal α

$$\hat{\alpha} = \arg \max_{\alpha} L(\alpha)$$

- Dependent data (time series):** chain rule

$$L(\alpha) = f(x_1 | \alpha) f(x_2 | \alpha, x_1) f(x_3 | \alpha, x_2, x_1) \dots$$

- Negative log-likelihood $I(\alpha) = - \sum_i \log(f(x_i | \alpha, x_{i-1}, \dots))$
- Maximum likelihood:** Optimal α

$$\max_{\alpha} L(\alpha) = \min_{\alpha} I(\alpha)$$

Maximum likelihood estimation: reminder

- Normal distributions: if $x_i \sim N(\mu, \sigma^2)$, iid.

$$L(\theta) = \frac{1}{(\sqrt{2\pi}\sigma)^n} e^{-\frac{\sum_i(x_i-\mu)^2}{2\sigma^2}}$$

- Maximum likelihood

$$\hat{\mu} = \bar{x}$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_i (x_i - \bar{x})^2$$

- For ARMA models, assume normality of w_t !
- Negative log-likelihood

$$I(\mu, \phi, \sigma_w^2) = \frac{S(\mu, \phi)}{2\sigma_w^2} + \frac{n}{2} \log(2\pi\sigma_w^2) - \frac{1}{2} \log(1 - \phi^2)$$

$$S(\mu, \phi) = (1 - \phi^2)(x_1 - \mu)^2 + \sum_{t=2}^n [(x_t - \mu) - \phi(x_{t-1} - \mu)]^2$$

- How to find optimum?

- For σ^2 explicit

$$\hat{\sigma}_w^2 = \frac{1}{n} S(\hat{\mu}, \hat{\phi})$$

- Otherwise numerical optimization (unconstrained optimization)

ARMA

- **Autoregressive moving average ARMA(p, q)**

$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q}$$

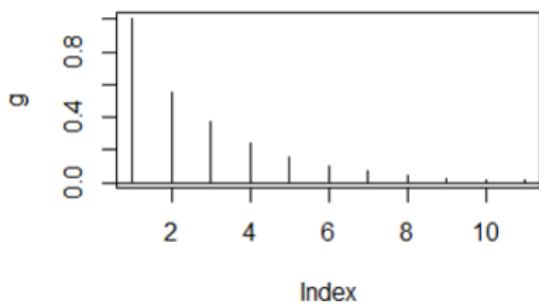
- ▶ $\phi_p \neq 0, \theta_q \neq 0$
- ▶ Is stationary
- ▶ $E x_t = 0$

- ACF for AR(1), MA(1), MA(2)

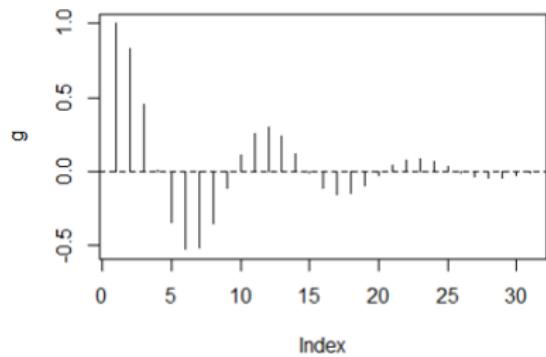
→ how to compute ACF for a general ARMA?

ACF for AR(2)

$$\phi^1 = 0.5 \quad \phi^2 = 0.1$$



$$\phi^1 = 1.5 \quad \phi^2 = -0.8$$



ACF for AR(p), MA(p)

- AR(p): using difference equations
- MA(q): using difference equations

$$\rho(h) = \begin{cases} \frac{\sum_{j=0}^{q-h} \theta_j \theta_{j+h}}{1+\theta^2+\dots+\theta_q^2} & 0 \leq h \leq q \\ 0 & h > q \end{cases}$$

ACF for ARMA(p,q)

- ARMA(p,q):

$$\phi(B)x_t = \theta(B)w_t$$

- Causal ARMA: $x_t = \phi^{-1}(B)\theta(B)w_t = \psi(B)w_t = \sum_{j=0}^{\infty} \psi_j w_{t-j}$
 - ▶ How to find ψ_j in practice? Coefficient matching
- **Theorem:** ACF of ARMA(p,q) can be found by solving general homogeneous equations:

$$\gamma(h) - \phi_1\gamma(h-1) - \dots - \phi_p\gamma(h-p) = 0, \quad h \geq \max(p, q+1)$$

- ▶ Initial conditions

$$\gamma(h) - \phi_1\gamma(h-1) - \dots - \phi_p\gamma(h-p) = \sigma_w^2 \sum_{j=h}^q \theta_j \psi_{j-h}, \quad 0 \leq h < \max(p, q+1)$$

ACF for ARMA(1,1)

- Show for ARMA(1,1)

$$\rho(h) = \frac{(1 + \theta\phi)(\phi + \theta)}{1 + 2\theta\phi + \theta^2} \phi^{h-1}, h \geq 1$$

- **Note:** pattern similar to AR(1) → hard to differentiate
- **Note:** ACF is 0 for $h > q$ from MA(q) → MA(q) is identifiable from ACF
- **How to differentiate between AR(p)? ARMA(p)?**

Partial correlation

A Gaussian intuition:

- Conditional density: $f(x, y|z) = \frac{f(x, y, z)}{f(z)}$
- if x , y and z were jointly normal then

$$f(x, y|z) = N\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right)$$

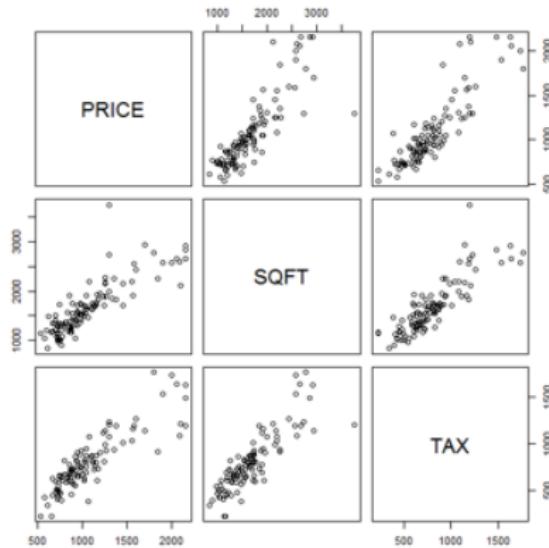
- Also,

$$\rho_{xy|z} = \frac{\text{cov}(x, y|z)}{\sqrt{\text{var}(x|z) \text{var}(y|z)}} = \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}}$$

- **What if $\Sigma_{12} = 0$?**

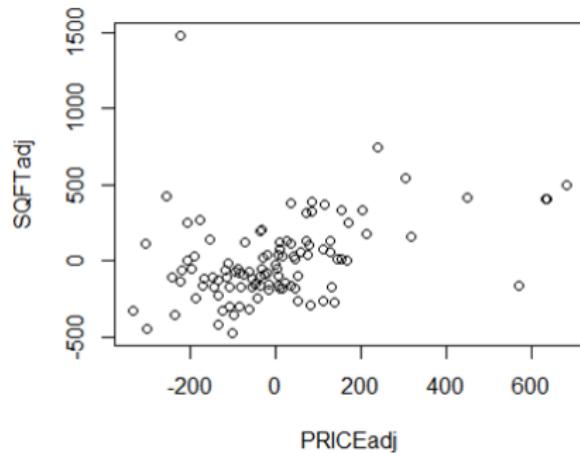
Partial autocorrelation

- **Example:** Albuquerque home prices
 - ▶ What if we remove information stored in TAX from PRICE and SQFT?



Partial autocorrelation

- $\hat{y} = \hat{\alpha}_0 + \hat{\alpha}_1 z$
- $\hat{x} = \hat{\beta}_0 + \hat{\beta}_1 z$
- $x' = x - \hat{x}$
- $y' = y - \hat{y}$
- **Partial autocorrelation**



- If we know α , β and z , we can reduce connection between x and y

```
> corr(cbind(PRICEadj,SQFTadj))  
[1] 0.3675204
```

PACF

- Partial autocorrelation function (PACF) for a stationary process

$$\phi_{11} = \text{corr}(x_{t+1}, x_t)$$

$$\phi_{hh} = \text{corr}(x'_{t+h}, x''_t), \quad h > 1$$

- ▶ where $x'_{t+h} = x_{t+h} - \sum_{j=1}^{h-1} \hat{\beta}_j x_{t+h-j}$
- ▶ and $x''_t = x_t - \sum_{j=1}^{h-1} \hat{\beta}_j x_{t+j}$
- ▶ **Note:** coefficients in x''_{t+h} and x'_{t+h} are the same (stationarity)
- **Example:** AR(1) $\phi_{11} = \phi, \phi_{22} = 0$

PACF for AR(p)

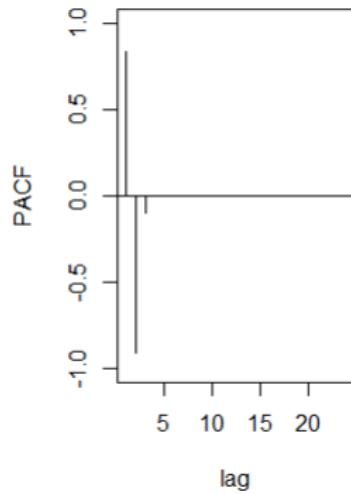
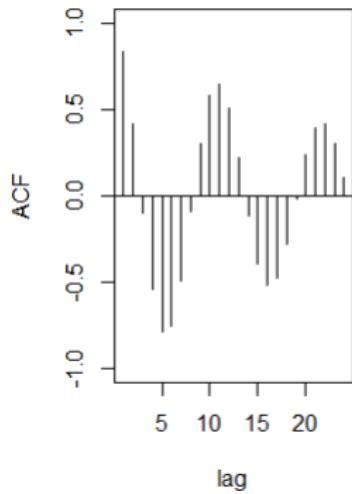
$$x_t = \sum_{j=1}^p \phi_j x_{t-j} + w_t$$

- It can be shown:
 - ▶ $\phi_{pp} = \phi_p$
 - ▶ $\hat{\beta}_1 = \phi_1, \dots, \hat{\beta}_p = \phi_p, \hat{\beta}_{p+1} = 0, \dots, \hat{\beta}_h = 0$ for $h > p$
- It means

$$\begin{aligned}\phi_{hh} &= \text{cov}(x_{t+h} - \sum_{j=1}^p \phi_j x_{t+h-j}, x_t - \sum_{j=1}^p \phi_j x_{t+j}) \\ &= \text{cov}(w_{t+h}, x_t - \sum_{j=1}^p \phi_j x_{t+j}) = 0, \quad \text{when } h > p\end{aligned}$$

PACF for AR(p)

- Example: AR(3) $\phi_1 = 1.5$, $\phi_2 = -0.75$, $\phi_3 = -0.1$

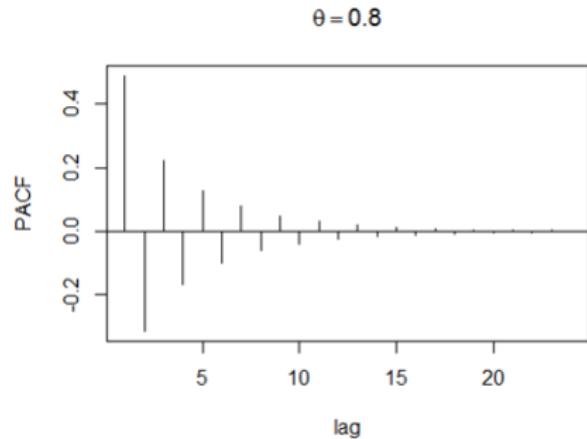


PACF for MA(1)

- If invertible,

$$\phi_{hh} = -\frac{(-\theta)^h(1-\theta^2)}{1-\theta^{2h+2}}, \quad h \geq 1$$

Decreases exponentially with h



ACF and PACF

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

How to differentiate between ARMA(p, q)?

Empirical ACF (EACF)

Idea:

- ARMA(p,q): $x_t = \sum_{j=1}^p \phi_j x_{t-j} + \sum_{j=1}^q \theta_j w_{t-j} + w_t$
- If we can estimate $\phi_j \rightarrow x'_t = x_t - \sum_{j=1}^p \phi_j x_{t-j}$ is linear function in w_t, \dots, w_{t-q}
- If we run regression x'_t against $w_t \dots w_{t-j}$:
 - ▶ Residuals are white noise, $j \geq q \rightarrow$ ACFs not significant
 - ★ Some of the coefficients will be 0
 - ▶ Residuals are not white noise, $j < q \rightarrow$ ACFs significant
 - ▶ Note: w_t s substituted by lagged residuals from a series of regressions
- If $x'_t = x_t - \sum_{j=1}^k \phi_j x_{t-j}, k < p \rightarrow$ white noise will never be achieved
 \rightarrow ACFs are not zero

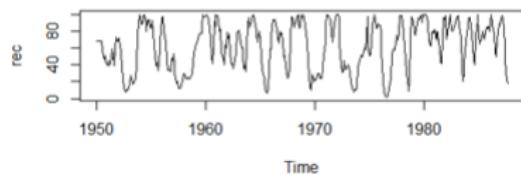
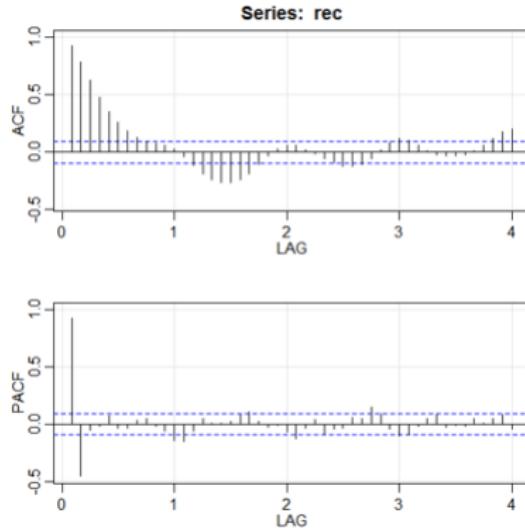
Empirical ACF (EACF)

- $k > p$ General result: ACFs are 0 for $j > q + (k - p)$
 - ▶ Example: ARMA(0,1)
- General conclusion for AR,MA = (k,j):
 - ▶ This is theoretical one! → not exactly the same for the samples

AR/MA	0	1	2
0	X	X	X	X	X	X	X
1	X	X	X	X	X	X	X
2	X	X	X	X	X	X	X
...	X	X	X	X	X	X	X
...	X	X	X	X	X	X	X
...	X	X	0	0	0	0	0
...	X	X	X	0	0	0	0
...	X	X	X	X	0	0	0
...	X	X	X	X	X	0	0

ARMA orders

- Recruitment series



Conclusion?

ARMA orders

- EACF

```
> TSA::eacf(rec)
```

AR/MA

0 1 2 3 4 5 6 7 8 9 10 11 12 13

0 x x x x x x x o o o o o x

1 x x x o o o o o o o o o o o

2 o o x x o o o o o o o o o o o

3 x o o x o o o o o o o o o o o

4 x x o o o o o o o o o o o o o

5 x x x o o o o o o o o o o o o

6 x x x o o o o o o o o o o o o

7 x x o o o o o o o o o o o o x o

ARMA orders

- AR(2) and ARMA(1,3)

- Conclusions?

```
> arima(rec, order=c(2,0,0))

Call:
arima(x = rec, order = c(2, 0, 0))

Coefficients:
          ar1      ar2  intercept
        1.3512 -0.4612    61.8585
  s.e.  0.0416  0.0417     4.0039

sigma^2 estimated as 89.33:  log likelihood = -1661.51,  aic = 3331.02
> arima(rec, order=c(1,0,3))

Call:
arima(x = rec, order = c(1, 0, 3))

Coefficients:
          ar1      ma1      ma2      ma3  intercept
        0.7826  0.5484  0.3239  0.2119    61.8609
  s.e.  0.0390  0.0554  0.0621  0.0530     4.1953

sigma^2 estimated as 88.43:  log likelihood = -1659.24,  aic = 3330.48
> |
```

Model selection

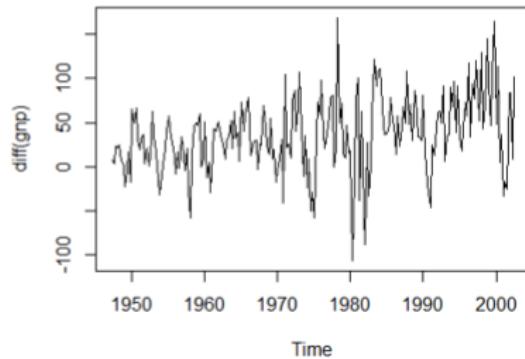
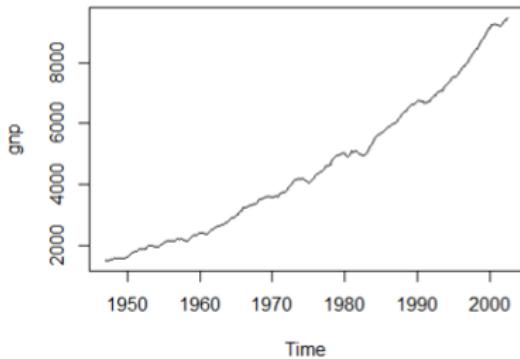
- Which model is suitable?
 - ▶ What is p, d, q is ARIMA(p,d,q)?
 - ▶ d is defined before!
 - ▶
- Step 1: Check ACF, PACF and EACF to define a few tentative models

Model selection

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

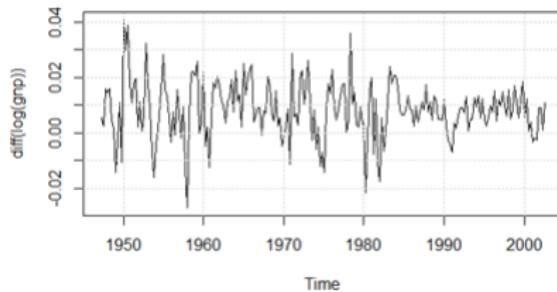
Model selection

- **Example:** GNP data
 - ▶ Trying differencing → non-constant variance and maybe trend? → transformation



Model selection

- Example: GNP data
 - ▶ Taking log and then differencing → still not perfect, but ... keep it as is.



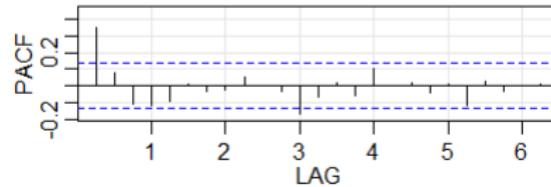
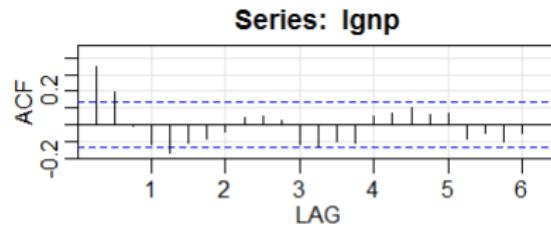
```
> adf.test(lgnp)

Augmented Dickey-Fuller Test

data: lgnp
Dickey-Fuller = -6.1756, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary
```

Model selection

- Example: GNP data
 - ▶ Testing ACF and PACF



Conclusion?

Model selection

- Example: GMP data
 - ▶ Checking EACF

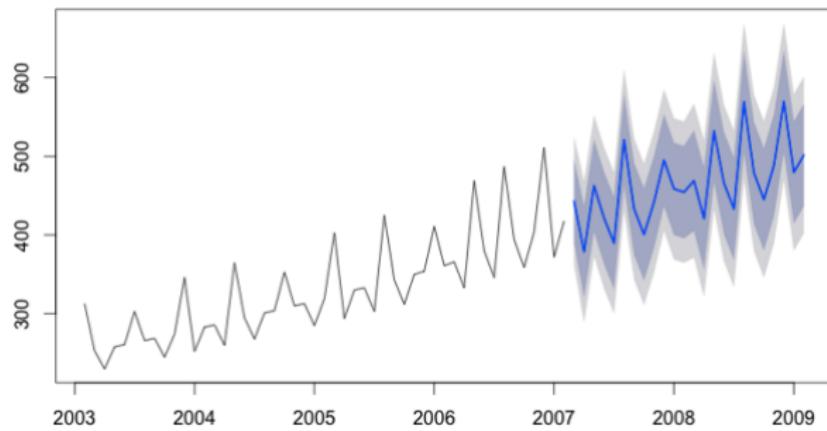
```
> TSA::eacf(lgnp)
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x o o x o o o o o o o o o o
1 x x o o o o o o o o o o o o o
2 x x o o o o o o o o o o o o o
3 x o x o o o o o o o o o o o o
4 x o x o o o o o o o o o o o o
5 o x x o x o o o o o o o o o o
6 x x x x x o o o o o o o o o o
7 x x o x o o x o o o o o o o o
```

Conclusion?

Forecasting

- We have our series $x_1 \dots x_n$
- Use series to predict m steps ahead: x_{n+m}^n should be based on our observed data $x_{n+m}^n = g(x_1, \dots, x_n)$

Forecasts from ARIMA(0,0,1)(1,1,0)[12] with drift



Forecasting

- Assume $g(x_1, \dots, x_n) = \alpha_0 + \sum_{k=1}^n \alpha_k x_k$
 - ▶ Best linear predictors
- How to find α 's?

$$\min E[(x_{n+m} - g(x_1, \dots, x_n))^2]$$

- Prediction equations
 - ▶ Find α 's by solving ($x_0 = 1$)
$$E[(x_{n+m} - x_{n+m}^n)x_k] = 0, k = 0, \dots, n$$
- **Note:** $n+1$ equations, $n+1$ unknowns

One-step-ahead

- Denote $x_{n+1}^n = \phi_{n1}x_n + \dots + \phi_{nn}x_1$
- Prediction equations give

$$\Gamma_n \phi_n = \gamma_n$$

$$\Gamma_n = \begin{pmatrix} \gamma(1-1) & \gamma(2-1) & \dots & \gamma(n-1) \\ \gamma(2-1) & \gamma(2-2) & \dots & \gamma(n-2) \\ \dots & \dots & \dots & \dots \\ \gamma(n-1) & \gamma(n-2) & \dots & \gamma(n-n) \end{pmatrix}$$

$$\phi_n = \begin{pmatrix} \phi_{n1} \\ \dots \\ \phi_{nn} \end{pmatrix} \quad \gamma_n = \begin{pmatrix} \gamma_1 \\ \dots \\ \gamma_n \end{pmatrix}$$

- **Note:** for ARMA models Γ_n is positive def \rightarrow unique solution

One-step-ahead

- Causal AR(p): for $n \geq p$ best linear prediction is

$$x_{n+1}^n = \phi_1 x_n + \dots + \phi_p x_{n-p+1}$$

- In general, solve system of equations $\rightarrow O(n^3)$ operations
- Much faster algorithms exist
 - ▶ Durbin-Levinson algorithm
 - ▶ Innovations algorithm
- **Property:** PACF of a stationary process can be obtained as ϕ_{nn} by solving $\Gamma_n \phi_n = \gamma_n$

One-step-ahead

- Mean square prediction error (MSPE)

$$P_{n+1}^n = E[(x_{n+1} - x_{n+1}^n)^2] = \gamma(0) - \gamma_n' \Gamma_n^{-1} \gamma_n$$

- Confidence intervals for x_{n+1}

$$x_{n+1}^n \pm \alpha \sqrt{P_{n+1}^n}$$

- m-step ahead in general? Prediction equations
 - ▶ Difficult in general

Read home

- Ch 3.2-3.4
- Paper "Consistent Estimates of Autoregressive Parameters and Extended Sample Autocorrelation" by Tsay and Tiao
- R code: eacf in TSA package

m-step-ahead for ARMA

- Assume causal and invertible ARMA(p,q)
- Finite past prediction

$$x_{n+1}^n = E(x_{n+1}|x_n, \dots, x_1)$$

- Infinite past prediction

$$\tilde{x}_{n+m}^n = E(x_{n+m}|x_n, \dots, x_1, x_0, x_{-1}, \dots)$$

- m-step-ahead forecast for infinite past

- ▶ Compute recursively

$$\tilde{x}_{n+m} = - \sum_{j=1}^{m-1} \pi_j \hat{x}_{n+m-j} - \sum_{j=m}^{\infty} \pi_j \tilde{x}_{n+m-j}, \quad m = 1, 2, \dots$$

- m-step ahead prediction error: $P_{n+m}^n = \sigma_w^2 \sum_{j=0}^{m-1} \psi_j^2$

Long-range forecasts

- What if $m \rightarrow \infty$?

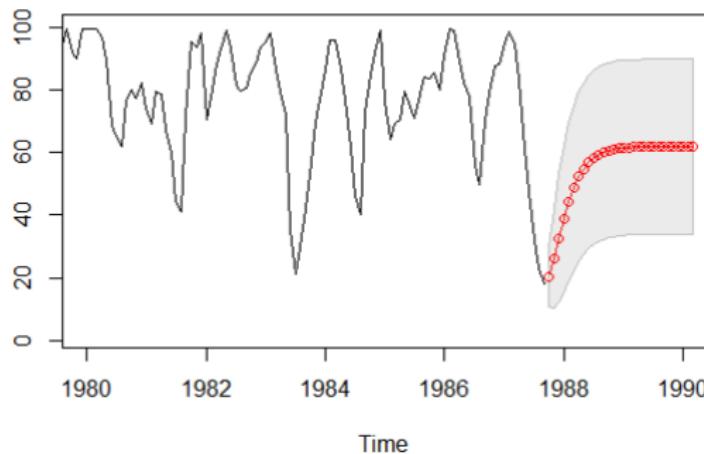
$$\tilde{x}_{n+m} \rightarrow 0(\text{or } \mu)$$

$$P_{n+m}^n \rightarrow \sigma_x^2$$

m-step-ahead

- Recruitment, AR(2)

$$x_{n+m}^n \pm 2\sqrt{P_{n+m}^n}$$



Truncated prediction

- Ignore non-positive j in x_j

$$\tilde{x}_{n+m} = - \sum_{j=1}^{m-1} \pi_j \tilde{x}_{n+m-j} - \sum_{j=m}^{\infty} \pi_j x_{n+m-j}, m = 1, 2, \dots$$

- For ARMA, truncated prediction formula:
 - Recursive computation, explicit

$$\tilde{x}_{n+m}^n = \phi_1 \tilde{x}_{n+m-1}^n + \dots + \phi_p \tilde{x}_{n+m-p}^n + \theta_1 \tilde{w}_{n+m-1}^n + \dots + \theta_q \tilde{w}_{n+m-q}^n$$

$$\tilde{w}_t^n = \tilde{x}_t^n - \phi_1 \tilde{x}_{t-1}^n - \dots - \phi_p \tilde{x}_{t-p}^n - \theta_1 \tilde{w}_{t-1}^n - \dots - \theta_q \tilde{w}_{t-q}^n$$

- Boundary conditions: $\tilde{x}_t^n = x_n, 1 \leq t \leq n, \tilde{x}_t^n = 0, t \leq 0$

$$\tilde{w}_t^n = 0, t \leq 0 \quad \text{or } t > n$$

Time Series Analysis

Lecture 6: ARIMA models summary

State space models

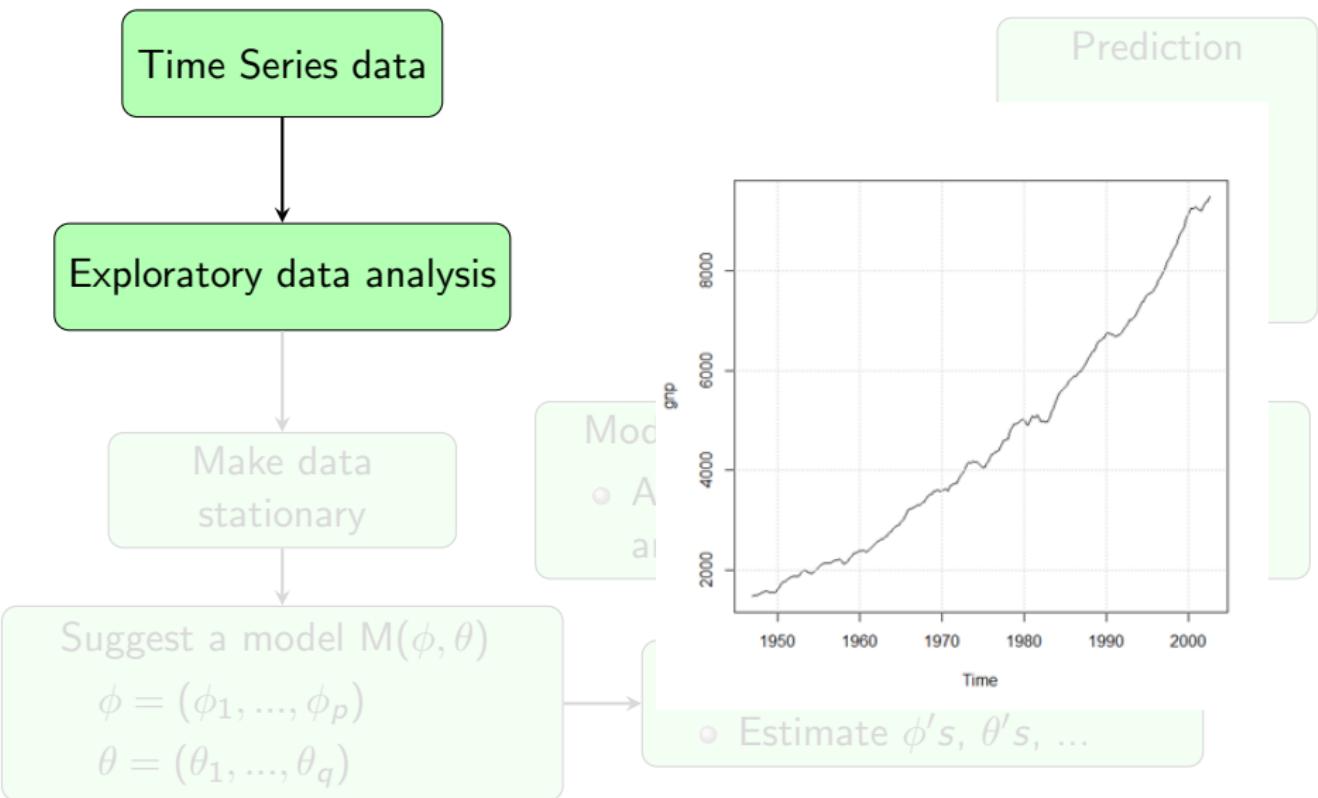
Tohid Ardesthiri

Linköping University
Division of Statistics and Machine Learning

September 27, 2019



Time domain: The Big Picture



Time domain: The Big Picture

Time Series data

$$Y_t = \nabla(\log(X_t))$$

Prediction

Exploratory data analysis

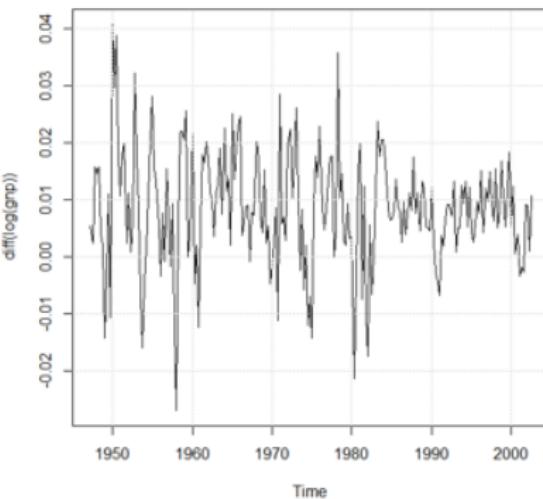
Make data stationary

Suggest a model $M(\phi, \theta)$

$$\phi = (\phi_1, \dots, \phi_p)$$

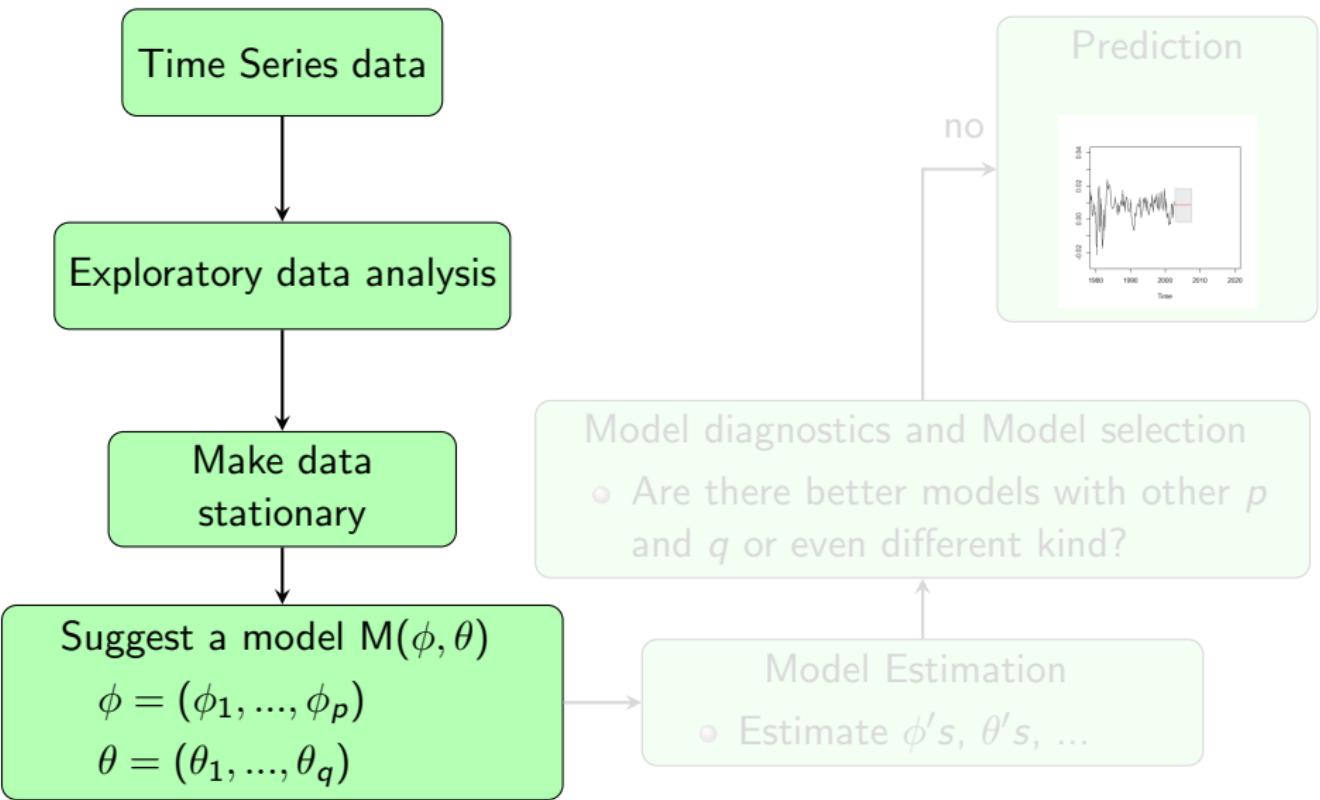
$$\theta = (\theta_1, \dots, \theta_q)$$

Model
A
ai

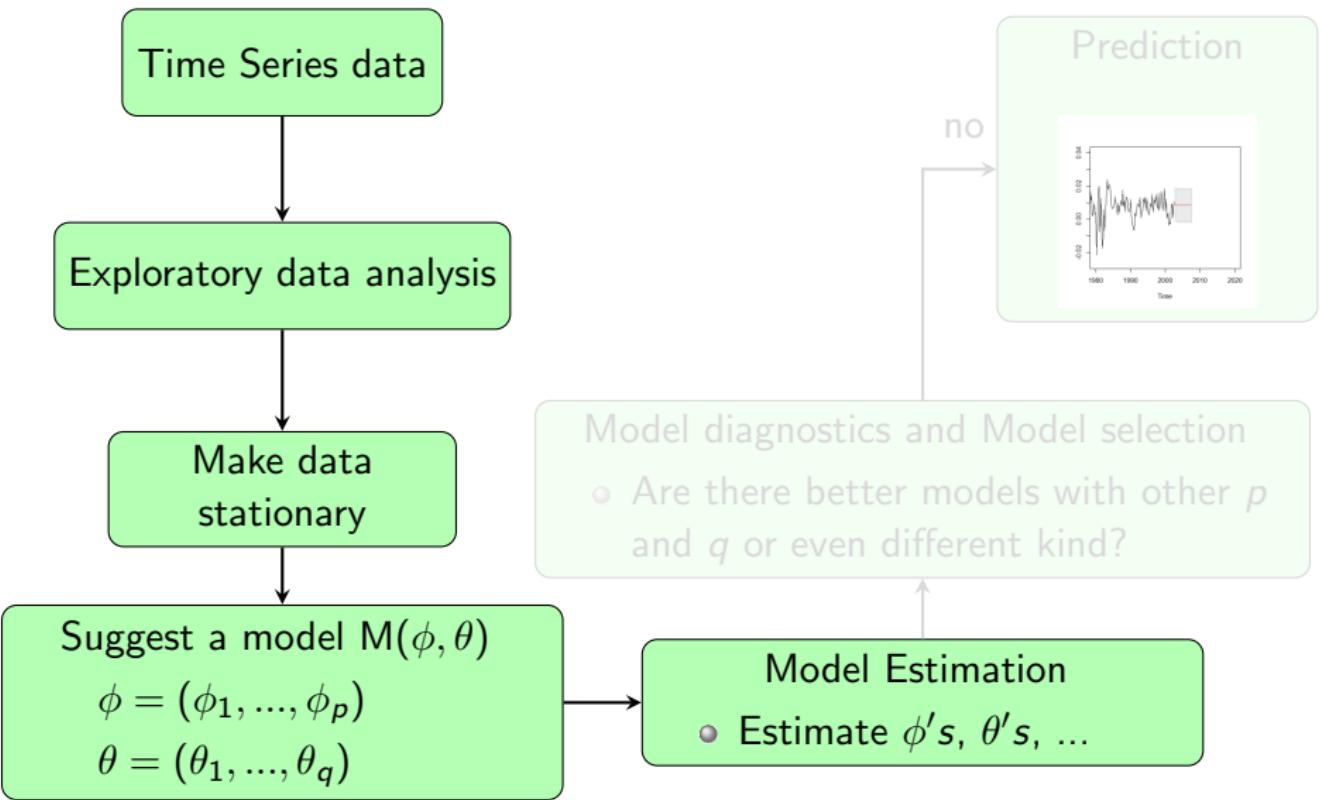


Estimate ϕ 's, θ 's, ...

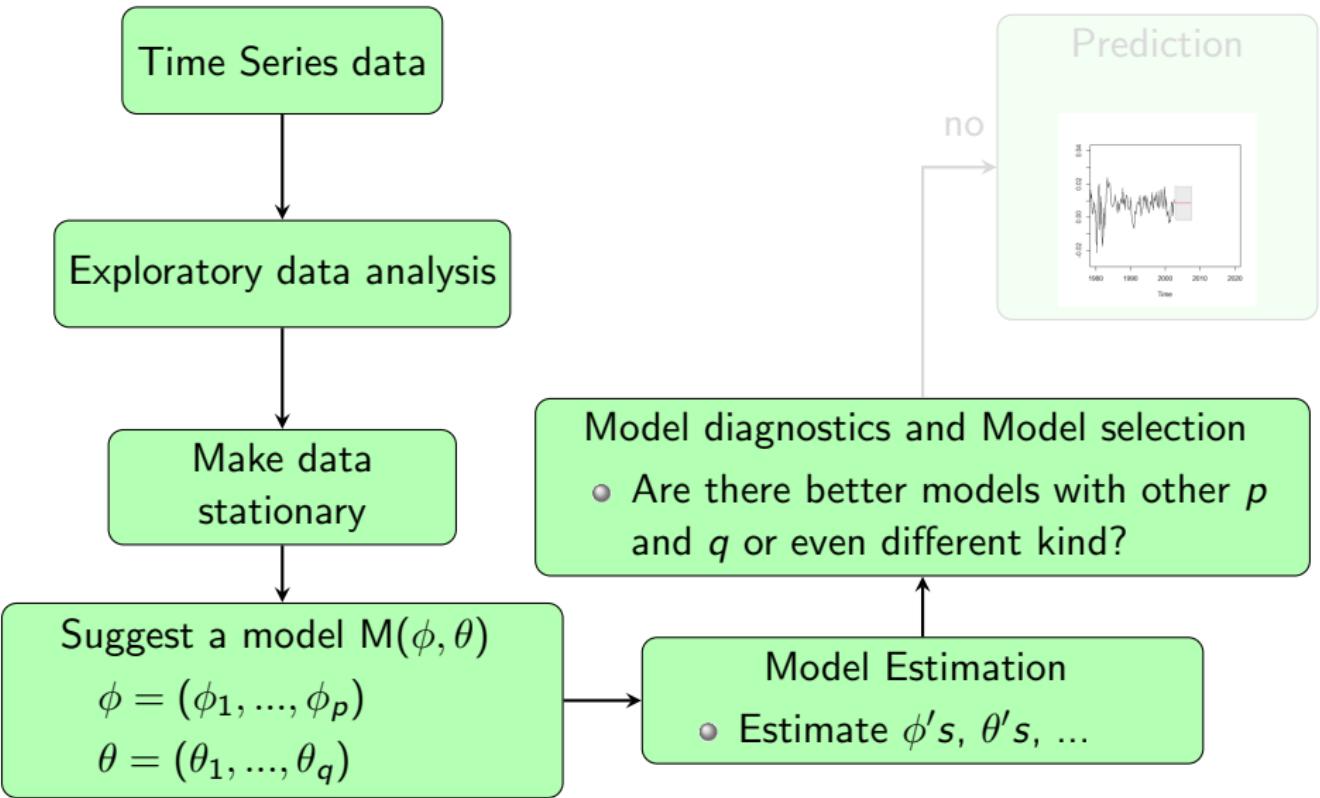
Time domain: The Big Picture



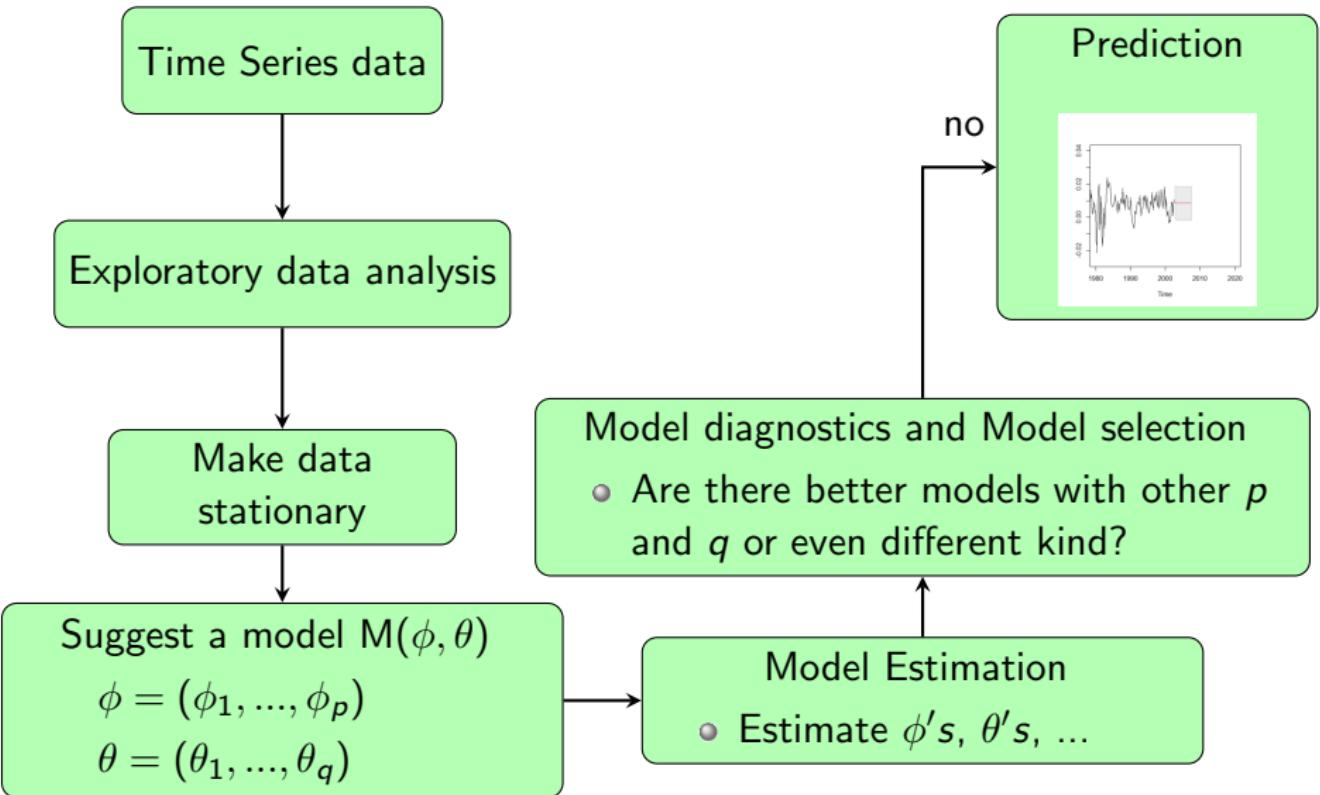
Time domain: The Big Picture



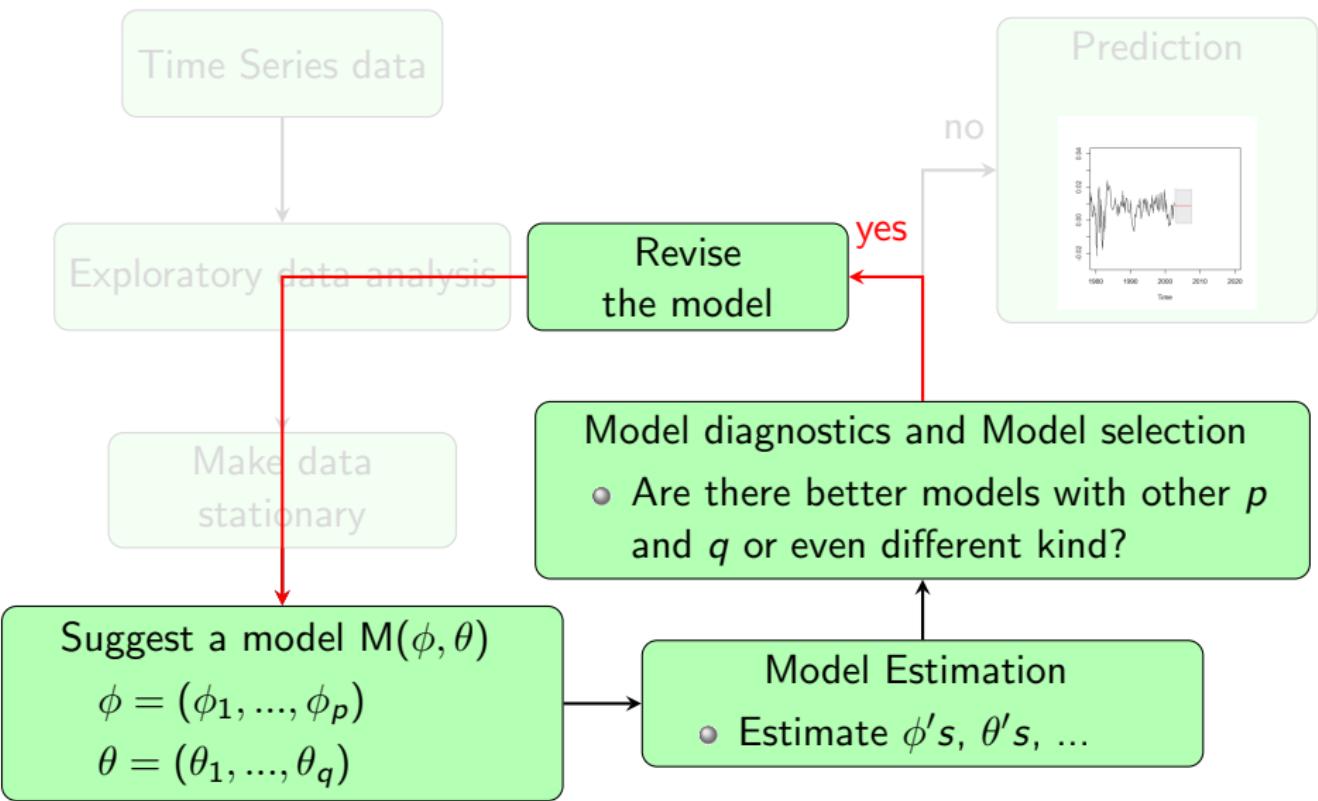
Time domain: The Big Picture



Time domain: The Big Picture



Time domain: The Big Picture



Model selection

Fit the tentative models, compare them

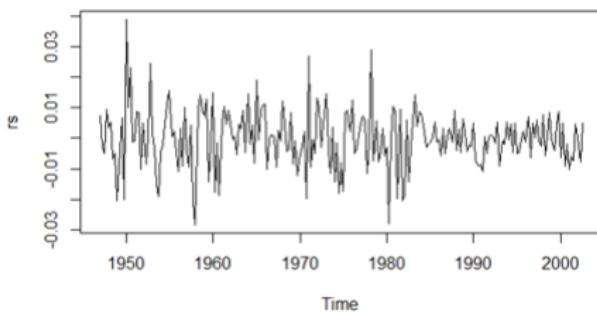
- Analytical measures: AIC, BIC
 - ▶ Penalize models with many parameters → simpler models
- Residual analysis

Residual analysis

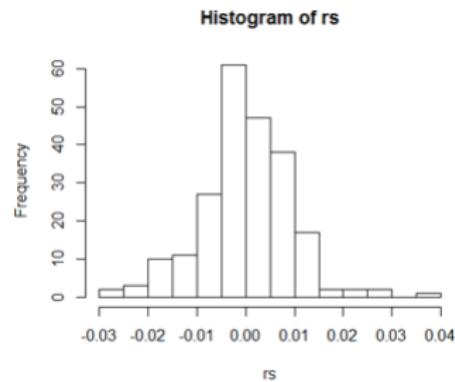
- Residuals $r_t = x_t - \hat{x}_t^{t-1}$? they are innovations
 - ▶ Note: computed from one-step-ahead predictions!
 - ▶ Measures predictive quality of the model (compare OLS)
- Residual analysis
 - ▶ Visual inspection: stationary? Patterns?
 - ▶ Histograms, Q-Q plots
 - ▶ ACF, PACF
 - ▶ Runs test
 - ▶ Box-Ljung test

Residual analysis - Visual inspection

Histogram and visual inspection

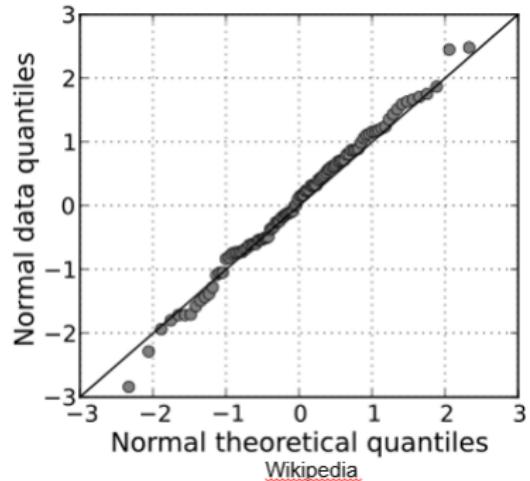
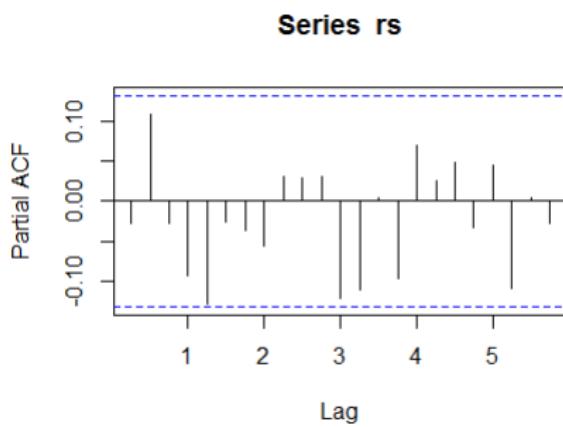


If looks white is good



If looks Normal is good

Residual analysis - ACF /PACF Q-Q plots



If between the blue lines good

If along the diagonal line GOOD

Statistical tests

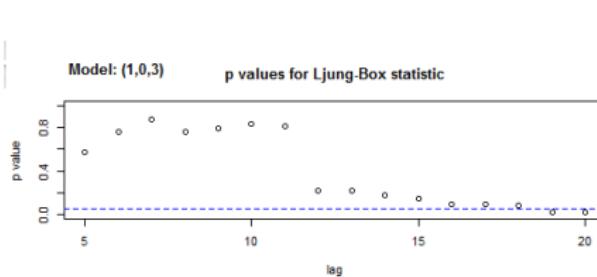
Tests are used to test independence

Runs test

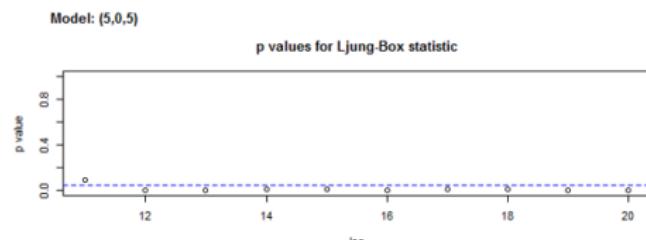
- H_0 : x_t values are i.i.d. **p-value NOT small**
- H_a : x_t values are not i.i.d. **p-value small**

Box-Ljung test

- H_0 : data are independent **p-value NOT small**
- H_a : data are not independent **p-value small**



GOOD



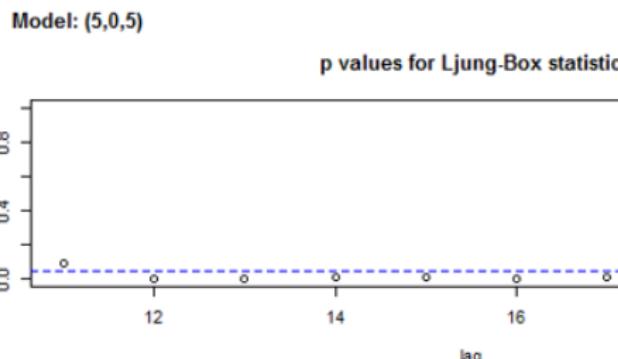
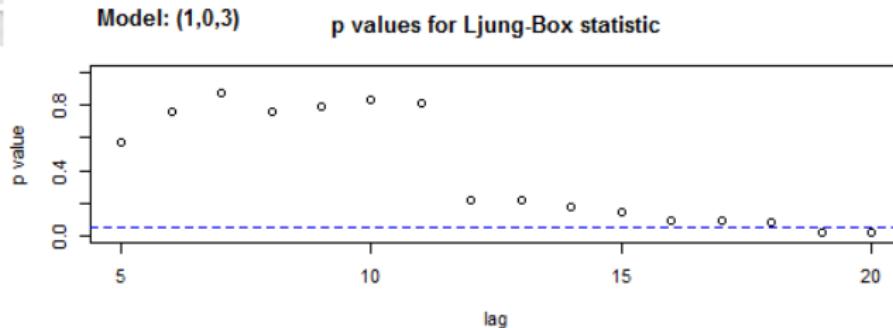
BAD

Overfitting

- Occams razor: among equally good models, choose the simplest one
- Overfitting: taking too complex models leads to bad predictions
- If ARIMA(p, d, q) has almost the same predictive quality as ARIMA(p', d', q') , take the one with less parameters

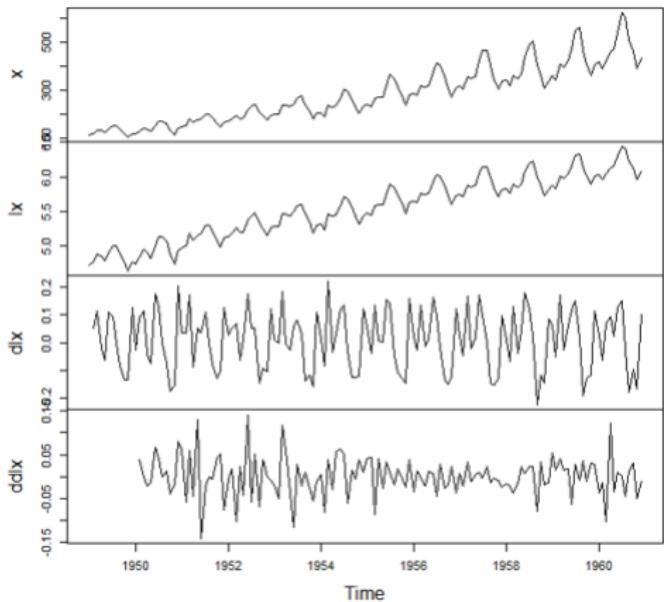
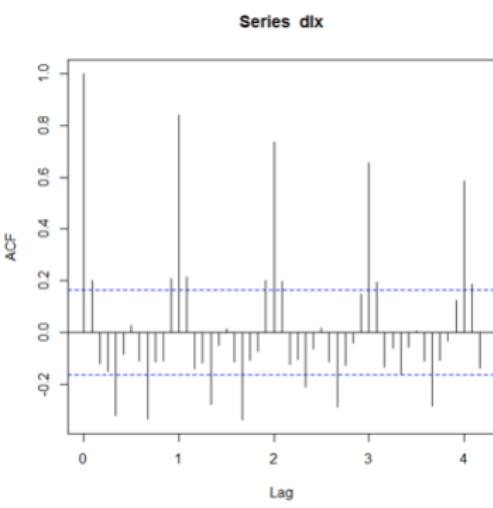
Overfitting

- Example: Recruitment series
 - Fit ARIMA(1,0,3) and ARIMA(5,0,5)



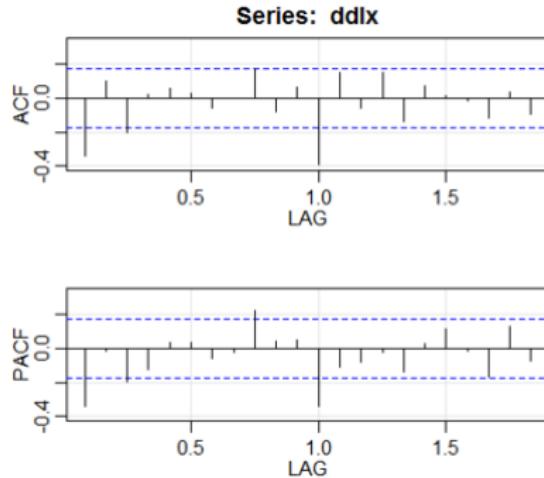
SARIMA - Air passangers

- Example: Air passangers



SARIMA - Air passangers

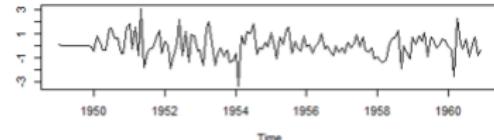
- Example: Air passangers



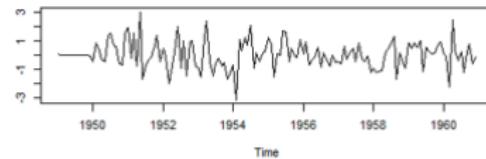
ARIMA(0, 1, 1)₁₂ or
ARIMA(1, 1, 0)₁₂

SARIMA - Air passengers

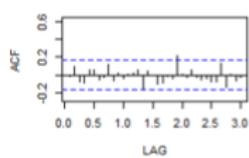
Model: (1,1,1) (0,1,1) Standardized Residuals



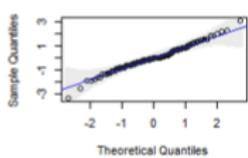
Model: (1,1,1) (1,1,0) Standardized Residuals



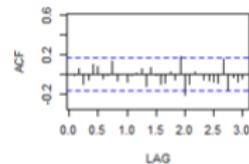
ACF of Residuals



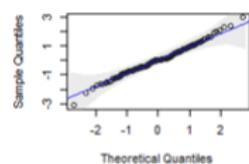
Normal Q-Q Plot of Std Residuals



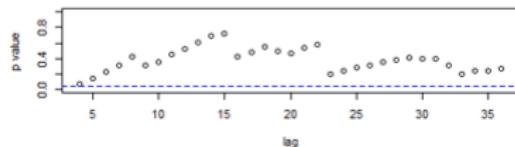
ACF of Residuals



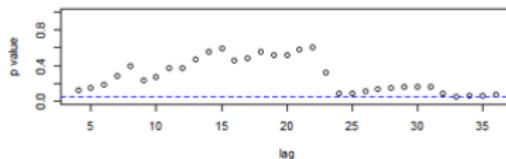
Normal Q-Q Plot of Std Residuals



p values for Ljung-Box statistic

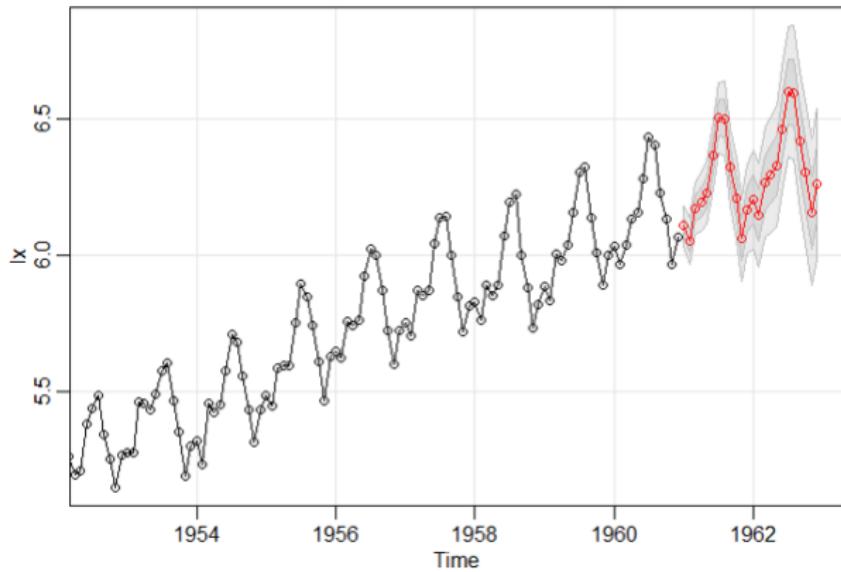


p values for Ljung-Box statistic



SARIMA

- Forecasting



Read home

- Shumway and Stoffer, Chapter 1, 2 and 3

ARIMA models

Time series models so far

$$\phi^P(B)x_t = \theta^q(B)w_t$$

Model	Concise form
AR(p)	$\phi^P(B)x_t = w_t$
MA(q)	$x_t = \theta^q(B)w_t$
ARMA(p, q)	$\phi^P(B)x_t = \theta^q(B)w_t$
ARIMA(p, d, q)	$\phi^P(B)(1 - B)^d x_t = \theta^q(B)w_t$
ARMA($P, Q)_s$	$\Phi^P(B^s)x_t = \Theta^Q(s)w_t$
ARIMA($P, D, Q)_s$	$\Phi^P(B^s)(1 - B^s)^D x_t = \Theta^Q(B^s)w_t$
ARMA($p, q) \times (P, Q)_s$	$\Phi^P(B^s)\phi^P(B)x_t = \Theta^Q(B^s)\theta^q(B)w_t$
ARIMA($p, d, q) \times (P, D, Q)_s$	$\Phi^P(B^s)\phi^P(B)(1 - B^s)^D(1 - B)^d x_t = \Theta^Q(B^s)\theta^q(B)w_t$

* The notation used in this slide deviates from the notation used in the course literature so far.

Consider an AR(2) model

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + w_t$$

Let $\mathbf{z}_t = \begin{bmatrix} x_t \\ x_{t-1} \end{bmatrix}$ and $e_t = \begin{bmatrix} w_t \\ 0 \end{bmatrix}$.

Show that we rewrite the AR(2) model in the state space form:

$$\begin{aligned}\mathbf{z}_t &= \begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{bmatrix} \mathbf{z}_{t-1} + e_t \\ x_t &= [1 \ 0] \mathbf{z}_t,\end{aligned}$$

$$\phi^P(B)x_t = \theta^q(B)w_t$$

Can we rewrite any model of this form as a state space model?

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t,$$

$$\mathbf{x}_t = C\mathbf{z}_t + \nu_t,$$

$$\phi^p(B)x_t = \theta^q(B)w_t$$

Outline of the solution:

Let $r = \max(p, q + 1)$,

$$\phi^r(B) = 1 - \phi_1 B - \cdots - \phi_r B^r,$$

$$\theta^r(B) = 1 + \theta_1 B + \cdots + \theta_{r-1} B^{r-1},$$

$\phi^r(B)(\theta^r(B))^{-1}x_t = w_t$. Hence, for $z_t = (\theta^r(B))^{-1}x_t$ we can have

$$\phi^r(B)z_t = w_t$$

$$z_t = \begin{bmatrix} z_t \\ z_{t-1} \\ z_{t-2} \\ \vdots \\ z_{t-r+1} \end{bmatrix} \text{ and } z_t = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_r \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} z_{t-1} + \begin{bmatrix} w_t \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

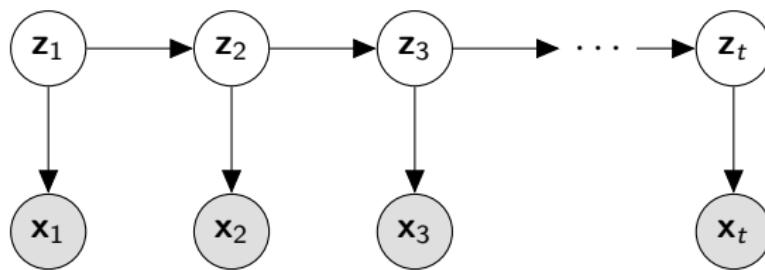
$$x_t = [1 \ \theta_1 \ \theta_2 \ \cdots \ \theta_r] z_t$$

State Space models - graphical models

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t, \quad e_t \sim f_e(\cdot)$$

$$\mathbf{x}_t = C\mathbf{z}_t + \nu_t, \quad \nu_t \sim f_\nu(\cdot)$$

A probabilistic graphical model for stochastic dynamical system with latent state \mathbf{z}_k and observations \mathbf{x}_k

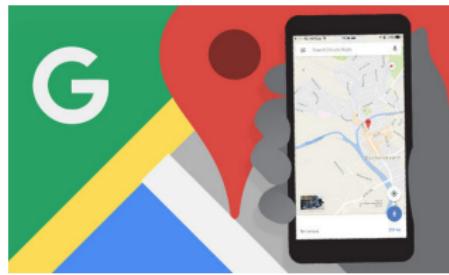
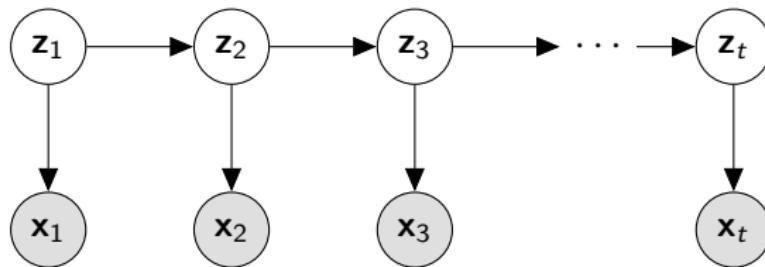


The main tool here is the probability Calculus; Bayes rule and marginalization.

Dynamical systems - more general case

$$\mathbf{z}_t = \mathcal{F}(\mathbf{z}_{t-1}) + e_t, \quad e_t \sim f_e(\cdot)$$

$$\mathbf{x}_t = \mathcal{C}(\mathbf{z}_t) + \nu_t, \quad \nu_t \sim f_\nu(\cdot)$$

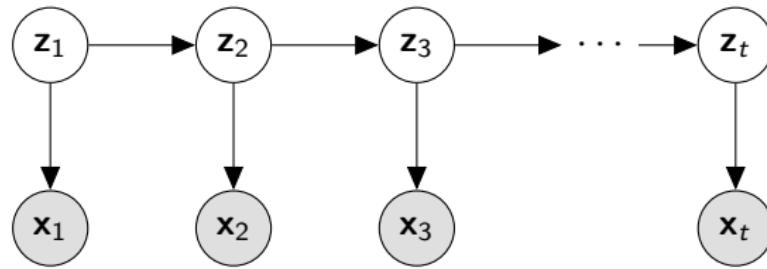


State Space models - Linear and Gaussian

Our main focus will be on linear and Gaussian models:

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t, \quad e_t \sim N(0, Q)$$

$$\mathbf{x}_t = C\mathbf{z}_t + \nu_t, \quad \nu_t \sim N(0, R)$$



Bayesian Inference

Bayesian inference is a means of combining prior beliefs with the data (evidence) to obtain posterior beliefs.

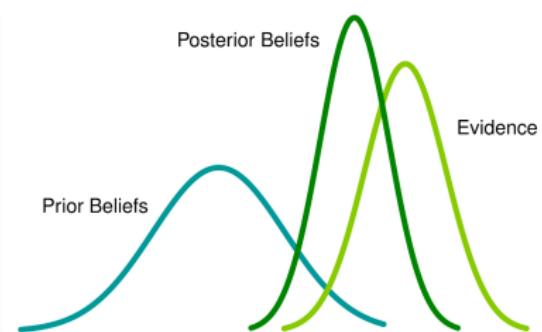
Example: likelihood update

$$f(z|x) \propto f(x|z)f(z)$$

Probability Calculus

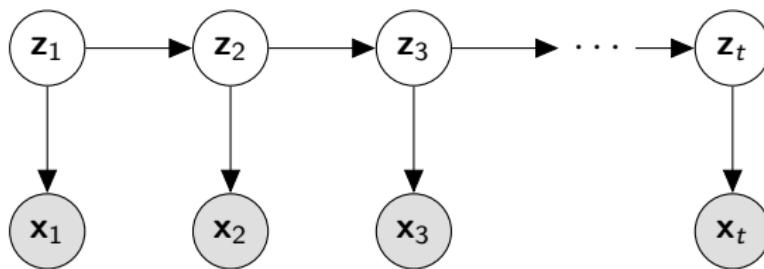
$$f(z, x) = f(z|x)f(x)$$

$$f(z, x) = f(x|z)f(z)$$



Online recursive algorithms

Consider a stochastic dynamical system represented by the following recursion



$$z_1 \sim f(z_1), \quad (1a)$$

$$x_k \sim f(x_k | z_k), \quad (1b)$$

$$z_{k+1} \sim f(z_{k+1} | z_k). \quad (1c)$$

The Bayesian filtering recursion corresponds to computing the posterior distributions $f(z_k | x_{1:k})$;

$$f(z_k | x_{1:k}) = \frac{f(z_k | x_{1:k-1}) f(x_k | z_k)}{\int f(z_k | x_{1:k-1}) f(x_k | z_k) dz_k}. \quad (2)$$

The density $f(z_k | x_{1:k-1})$ in the numerator of (2) which is called the predicted density of z_k and is obtained by integration as in

$$f(z_k | x_{1:k-1}) = \int f(z_k | z_{k-1}) f(z_{k-1} | x_{1:k-1}) dz_{k-1}. \quad (3)$$

Properties of the Normal density function

Property 1: $f(\mathbf{z})f(\mathbf{x}|\mathbf{z}) = f(\mathbf{z}, \mathbf{x})$

$$N(\mathbf{z}; \mu, \Sigma)N(\mathbf{x}; C\mathbf{z}, R) = N\left(\begin{bmatrix}\mathbf{z} \\ \mathbf{x}\end{bmatrix}; \begin{bmatrix}\mu \\ C\mu\end{bmatrix}, \begin{bmatrix}\Sigma & \Sigma C^T \\ C\Sigma & C\Sigma C^T + R\end{bmatrix}\right)$$

Property 2: marginalization and conditioning

If x, y were jointly normal:

$$f(x, y) = N\left(\begin{bmatrix}x \\ y\end{bmatrix}; \begin{bmatrix}\mu_1 \\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right)$$

then

$$f(x) = N(x; \mu_1, \Sigma_{11})$$

$$f(y) = N(y; \mu_2, \Sigma_{22})$$

$$f(x|y) = N(x; \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})$$

$$f(y|x) = N(y; \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})$$

The Kalman Filter's Foundation

Let \mathbf{z} have a normal prior distribution with mean μ and covariance Σ , i.e., $\mathbf{z} \sim N(\mathbf{z}; \mu, \Sigma)$.

An observation \mathbf{x} with the likelihood function $f(\mathbf{x}|\mathbf{z}) = N(\mathbf{x}; C\mathbf{z}, R)$ is in hand where C is a matrix with proper dimensions and R is a covariance matrix. The posterior distribution of \mathbf{z} can be obtained using the Bayes' rule

$$f(\mathbf{z}|\mathbf{x}) = \frac{f(\mathbf{z})f(\mathbf{x}|\mathbf{z})}{\int f(\mathbf{z})f(\mathbf{x}|\mathbf{z}) d\mathbf{z}} \quad (4)$$

$$= \frac{N(\mathbf{z}; \mu, \Sigma)N(\mathbf{x}; C\mathbf{z}, R)}{\int N(\mathbf{z}; \mu, \Sigma)N(\mathbf{x}; C\mathbf{z}, R) d\mathbf{z}}. \quad (5)$$

The posterior distribution $f(\mathbf{z}|\mathbf{x})$ has an analytical solution and turns out to be the normal distribution $N(\mathbf{z}; \mu', \Sigma')$ where

$$\mu' = \mu + K(\mathbf{x} - C\mu), \quad (6a)$$

$$\Sigma' = \Sigma - KC\Sigma, \quad (6b)$$

where

$$K = \Sigma C^T (C\Sigma C^T + R)^{-1}. \quad (7)$$

Time Series Analysis

Lecture 7: State Space Model - Estimation

Tohid Ardestiri

Linköping University
Division of Statistics and Machine Learning

October 2, 2019



Kalman filter

Kalman filter is an algorithm that uses time series data, **containing statistical noise and unknown innovations**, and produces estimates of latent (hidden) process that tend to be more accurate than those based on a single observations using a probabilistic framework.

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + \mathbf{e}_t,$$

$$\mathbf{x}_t = C\mathbf{z}_t + \nu_t,$$

Kalman filtering output is

$$f(\mathbf{z}_t | \mathbf{x}_{1:t}).$$

That is, it computes the the posterior density of \mathbf{z}_t using the observations up to time t .

Kalman filtering recursion

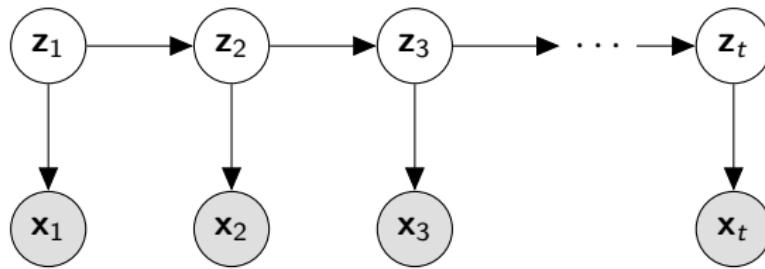
- ① initial estimate at $t = 1 \rightarrow N(\mathbf{z}_1; m_0, P_0)$
- ② observation update using \mathbf{x}_t and $\mathbf{x}_t = C\mathbf{z}_t + \nu_t \rightarrow N(\mathbf{z}_t; m_{t|t}, P_{t|t})$
- ③ prediction using $\mathbf{z}_{t+1} = A\mathbf{z}_t + e_{t+1} \rightarrow N(\mathbf{z}_t; m_{t+1|t}, P_{t+1|t})$
- ④ $t \leftarrow t + 1$
- ⑤ go to 2

State Space models - Time varying

State space models can be time-varying

$$\mathbf{z}_t = A_t \mathbf{z}_{t-1} + e_t, \quad e_t \sim N(0, Q_t)$$

$$\mathbf{x}_t = C_t \mathbf{z}_t + \nu_t, \quad \nu_t \sim N(0, R_t)$$



State space models with known deterministic input

State space model with
input \mathbf{u} .

$$\begin{aligned}\mathbf{z}_t &= A\mathbf{z}_{t-1} + B\mathbf{u}_{t-1} + \mathbf{e}_t, \\ \mathbf{x}_t &= C\mathbf{z}_t + \nu_t,\end{aligned}$$

Initialization:

$$f(\mathbf{z}_1) = N(\mathbf{z}_1; m_{1|0}, P_{1|0})$$

```
1: Inputs:  $A, B, C, Q, R, \mathbf{u}_{1:T},$ 
    $\mathbf{x}_{1:T}, m_{1|0}, P_{1|0}$ 
2: for  $t = 1$  to  $T$  do
   Kalman filter observation update step
3:    $K_t \leftarrow P_{t|t-1} C^T (C P_{t|t-1} C^T + R)^{-1}$ 
4:    $m_{t|t} \leftarrow m_{t|t-1} + K_t (\mathbf{x}_t - C m_{t|t-1})$ 
5:    $P_{t|t} \leftarrow P_{t|t-1} - K_t C P_{t|t-1}$ 
   Kalman filter prediction step
6:    $m_{t+1|t} \leftarrow A m_{t|t} + B \mathbf{u}_t$ 
7:    $P_{t+1|t} \leftarrow A P_{t|t} A^T + Q$ 
8: end for
9: Outputs:  $m_{t|t}$  and  $P_{t|t}$  for  $t = 1 : T$ 
```

Kalman Smoothing

The purpose of Kalman smoothing is to compute the marginal posterior distribution of \mathbf{z}_t at time t after receiving observations up to time T where $T > t$:

$$f(\mathbf{z}_t | \mathbf{x}_{1:T}) = N(\mathbf{z}_t; m_{t|T}, P_{t|T})$$

The RTS smoother uses a Kalman filter in its forward path. In its backwards path it updates the densities using the relation

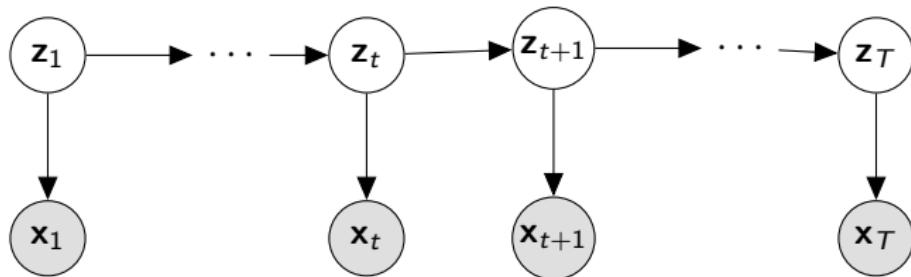
$$\mathbf{z}_t = A_{t-1} \mathbf{z}_{t-1} + \mathbf{e}_t$$

RTS Smoother's derivation

Assume $f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T})$ is available as in

$$f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T}) = N(\mathbf{z}_{t+1}; \mathbf{m}_{t+1|T}, \mathbf{P}_{t+1|T})$$

For example $f(\mathbf{z}_T | \mathbf{x}_{1:T})$ which is the filtering density of \mathbf{z}_T is available after filtering.



The objective is to compute $f(\mathbf{z}_t, \mathbf{z}_{t+1} | \mathbf{x}_{1:T})$.

RTS Smoother's derivation

The joint posterior $f(\mathbf{z}_t, \mathbf{z}_{t+1} | \mathbf{x}_{1:t})$ can be written as

$$\begin{aligned} f(\mathbf{z}_t, \mathbf{z}_{t+1} | \mathbf{x}_{1:t}) &= N(\mathbf{z}_t; m_{t|t}, P_{t|t}) N(\mathbf{z}_{t+1}; A\mathbf{z}_t, Q) \\ &= N\left(\begin{bmatrix} \mathbf{z}_t \\ \mathbf{z}_{t+1} \end{bmatrix}, \begin{bmatrix} m_{t|t} \\ Am_{t|t} \end{bmatrix}, \begin{bmatrix} P_{t|t} & P_{t|t}A^T \\ AP_{t|t} & AP_{t|t}A^T + Q \end{bmatrix}\right) \end{aligned}$$

Using the conditioning property of the multivariate normal distribution $f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t})$ can be computed as a normal density as given in the following:

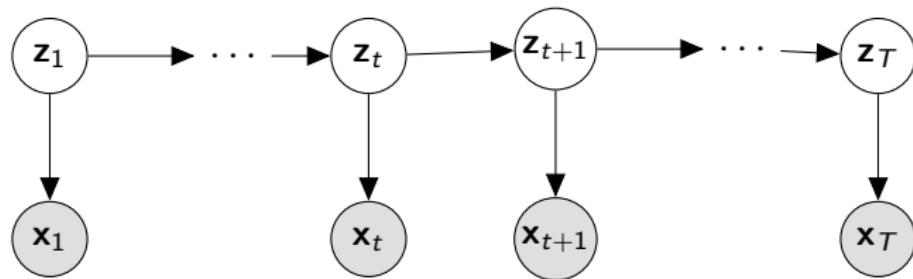
$$f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t}) = N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t)$$

where \tilde{m}_t is a function of \mathbf{z}_{t+1} .

RTS Smoother's derivation

Note the Markov property

$$f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:T}) = f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t})$$



Assume $f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T})$ is available as in

$$f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T}) = N(\mathbf{z}_{t+1}; m_{t+1|T}, P_{t+1|T})$$

Recall that

$$\begin{aligned} f(\mathbf{z}_{t+1}, \mathbf{z}_t | \mathbf{x}_{1:T}) &= f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T}) f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:T}) \\ &= f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T}) f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t}) \\ &= N(\mathbf{z}_{t+1}; m_{t+1|T}, P_{t+1|T}) N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t) \end{aligned}$$

RTS Smoother's derivation **Whiteboard**

where

$$G_t = P_{t|t} A_t^T (A P_{t|t} A^T + Q)^{-1} = P_{t|t} A_t^T P_{t+1|t}^{-1}$$

$$\tilde{m}_t = m_{t|t} + G_t (\mathbf{z}_{t+1} - A m_{t|t})$$

$$\tilde{P}_t = P_{t|t} - G_t (A P_{t|t} A^T + Q) G_t^T = P_{t|t} - G_t P_{t+1|t} G_t^T$$

Hence,

$$f(\mathbf{z}_{t+1}, \mathbf{z}_t | \mathbf{x}_{1:T}) = N(\mathbf{z}_{t+1}; m_{t+1|T}, P_{t+1|T}) N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t)$$

$$= N \left(\begin{bmatrix} \mathbf{z}_t \\ \mathbf{z}_{t+1} \end{bmatrix}, \begin{bmatrix} m_{t|t} + G_t (m_{t+1|T} - A m_{t|t}) \\ m_{t+1|T} \end{bmatrix}, \begin{bmatrix} G_t P_{t+1|T} G_t^T + \tilde{P}_t & G_t P_{t+1|T} \\ P_{t+1|T} G_t^T & P_{t+1|T} \end{bmatrix} \right)$$

RTS Smoother's derivation **Whiteboard**

The smoothing density's parameters is given by

$$G_t = P_{t|t} A_t^T (A P_{t|t} A^T + Q)^{-1} = P_{t|t} A_t^T P_{t+1|t}^{-1}$$

$$m_{t|T} = m_{t|t} + G_t(m_{t+1|T} - A m_{t|t})$$

$$\begin{aligned} P_{t|T} &= \tilde{P}_t + G_t P_{t+1|T} G_t^T = P_{t|t} - G_t P_{t+1|t} G_t^T + G_t P_{t+1|T} G_t^T \\ &= P_{t|t} + G_t(P_{t+1|T} - P_{t+1|t}) G_t^T \end{aligned}$$

RTS smoother's backwards recursion

Prove the backwards recursion of the RTS smoother for following state space model with initial prior on the state $f(\mathbf{z}_1) = N(\mathbf{z}_1; \mathbf{m}_0, \mathbf{P}_0)$

$$\mathbf{z}_t = A_{t-1}\mathbf{z}_{t-1} + e_t, \quad e_t \sim N(0, Q_t)$$

$$\mathbf{x}_t = C_t\mathbf{z}_t + \nu_t, \quad \nu_t \sim N(0, R_t)$$

1: **Inputs:** $A_t, Q_t, m_{t|t}, P_{t|t}, m_{t+1|t}, P_{t+1|t}$ for $1 \leq t \leq T$
initialization

2: **for** $t = T-1$ down to 1 **do**

3: $G_t \leftarrow P_{t|t}A_t^T P_{t+1|t}^{-1}$

4: $m_{t|T} \leftarrow m_{t|t} + G_t(m_{t+1|T} - A_t m_{t|t})$

5: $P_{t|T} \leftarrow P_{t|t} + G_t(P_{t+1|T} - P_{t+1|t})G_t^T$

6: **end for**

7: **Outputs:** $m_{t|T}, P_{t|T}$

State Space models - Estimation

We consider three approaches.

① (Variational Bayes)

T. Ardestiri, E. Özkan, U. Orguner and F. Gustafsson, "Approximate Bayesian Smoothing with Unknown Process and Measurement Noise Covariances," in IEEE Signal Processing Letters, vol. 22, no. 12, pp. 2450-2454, Dec. 2015.

② Direct maximum likelihood estimate

③ Expectation maximization (EM)

Variational Bayes smoothing with unknown time varying R_t and Q_t

Consider a Linear and Gaussian state space model with parameters

$$A_k = \text{Diag}(a, a),$$

$$a = \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix},$$

$$R_k^{\text{True}} = \left(2 - \cos\left(\frac{4\pi k}{K}\right) \right) R_0,$$

$$Q_k^{\text{True}} = \left(\frac{2}{3} + \frac{1}{3} \cos\left(\frac{4\pi k}{K}\right) \right) Q_0,$$

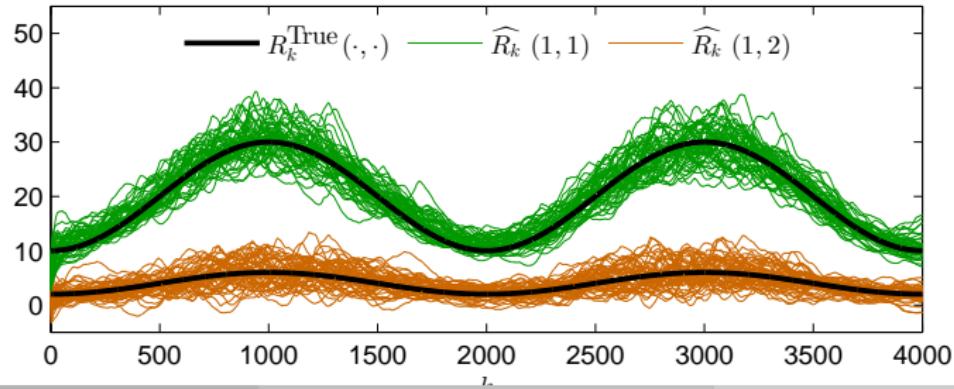
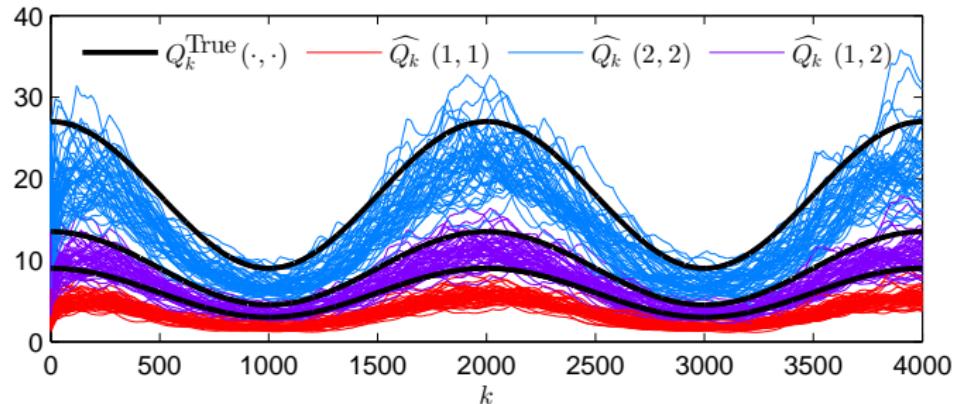
$$Q_0 = \text{Diag}(q, q),$$

$$q = \sigma_{\nu}^2 \begin{bmatrix} \tau^3/3 & \tau^2/2 \\ \tau^2/2 & \tau \end{bmatrix},$$

$$R_0 = \sigma_e^2 \begin{bmatrix} 5 & 1 \\ 1 & 5 \end{bmatrix},$$

$$C_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Variational Bayes smoothing with unknown time varying R_t and Q_t



Maximum likelihood methods

Whiteboard

Let $\theta = \{A, C, R, Q, m_0, P_0\}$ denote the unknown state space parameters

$$f(\mathbf{x}_{1:T}|\theta) = f(\mathbf{x}_1|\theta)f(\mathbf{x}_2|\mathbf{x}_1, \theta)f(\mathbf{x}_3|\mathbf{x}_{1:2}, \theta) \cdots f(\mathbf{x}_T|\mathbf{x}_{1:T-1}, \theta)$$

where

$$f(\mathbf{x}_{t+1}|\mathbf{x}_{1:t}, \theta) = \int f(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}, \mathbf{x}_{1:t}, \theta)f(\mathbf{z}_{t+1}|\mathbf{x}_{1:t}, \theta) d\mathbf{z}_{t+1}$$

This can be computed using the Kalman filter

$$\begin{aligned} f(\mathbf{x}_{t+1}|\mathbf{x}_{1:t}, \theta) &= \int f(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}, \mathbf{x}_{1:t}, \theta)f(\mathbf{z}_{t+1}|\mathbf{x}_{1:t}, \theta) d\mathbf{z}_{t+1} \\ &= \int N(\mathbf{x}_{t+1}; C\mathbf{z}_{t+1}, R)N(\mathbf{z}_{t+1}; m_{t+1|t}, P_{t+1|t}) d\mathbf{z}_{t+1} \\ &= N(\mathbf{z}_{t+1}; Cm_{t+1|t}, CP_{t+1|t}C^T + R) \end{aligned}$$

The negative logarithm of the likelihood becomes

$$\begin{aligned} I(\theta) &= - \sum_{t=1}^T \log f(\mathbf{x}_t | \mathbf{x}_{1:t-1}, \theta) \\ &= - \sum_{t=1}^T \log N(\mathbf{z}_{t+1}; Cm_{t+1|t}, CP_{t+1|t}C^T + R) \\ &= \frac{1}{2} \sum_{t=1}^T \log |CP_{t+1|t}C^T + R| \\ &\quad + \frac{1}{2} \sum_{t=1}^T (\mathbf{x}_t - Cm_{t+1|t})(CP_{t+1|t}C^T + R)^{-1}(\mathbf{x}_t - Cm_{t+1|t})^T \end{aligned}$$

which can be solved using for example Newton-Raphson method.

Maximum likelihood methods

The first two derivatives of the negative log-likelihood is computed with respect to the θ .

Then in the iterations of the Newton-Raphson method

- ① An initial value for for θ is selected, say $\theta^{(0)}$.
- ② A Kalman filter is run to compute the quantities for the first two derivatives.
- ③ A new set of parameters are obtained from a Newton-Raphson procedure.
- ④ Iterations are repeated until convergence.

Expectation Maximization

Whiteboard

- Expectation-maximization (EM) method can be used to compute the maximum likelihood (ML) estimate of the state space parameters.
- In the E (Expectation) step of the EM algorithm the conditional expectation of the joint log-likelihood is computed using the last estimates of the unknown parameters as in

$$\mathcal{Q} = E \left[\log f(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}) \mid \mathbf{x}_{1:T}, \theta^{(i)} \right] \quad (1)$$

where

$$\begin{aligned} \log f(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}) &= \log N(\mathbf{z}_1; m_0, P_0) - \frac{T+1}{2} \log |R| \\ &\quad - \frac{1}{2} \sum_{t=1}^T \text{Tr} (R^{-1}(\mathbf{x}_t - C\mathbf{z}_t)(\mathbf{x}_t - C\mathbf{z}_t)^T) - \frac{T}{2} \log |Q| \\ &\quad - \frac{1}{2} \sum_{t=1}^{T-1} \text{Tr} (Q^{-1}(\mathbf{z}_{t+1} - A\mathbf{z}_t)(\mathbf{z}_{t+1} - A\mathbf{z}_t)^T) + c. \end{aligned} \quad (2)$$

Therefore,

$$\begin{aligned} \mathcal{Q} = & -\frac{1}{2} E[(\mathbf{z}_0 - m_0) P_0^{-1} (\mathbf{z}_0 - m_0)^T + \log |P_0|] \\ & - \frac{T+1}{2} \log |R| - \frac{1}{2} \text{Tr} \left(R^{-1} \sum_{t=0}^T E[(\mathbf{x}_t - C\mathbf{z}_t)(\mathbf{x}_t - C\mathbf{z}_t)^T | \mathbf{x}_{1:T}] \right) \\ & - \frac{T}{2} \log |Q| - \frac{1}{2} \text{Tr} \left(Q^{-1} \sum_{t=0}^{T-1} E[(\mathbf{z}_{t+1} - A\mathbf{z}_t)(\mathbf{z}_{t+1} - A\mathbf{z}_t)^T | \mathbf{x}_{1:T}] \right) + c, \end{aligned} \tag{3}$$

In order to compute the expectations the RTS smoother's posterior $f(\mathbf{z}_t | \mathbf{z}_{1:T})$ is used.

Then in the iterations of the EM method

- ① An initial value for θ is selected, say $\theta^{(0)}$.
- ② A Kalman smoother is run using $\theta^{(i)}$
- ③ In the expectation step Q function as a function of θ is derived.
- ④ A new set of parameters $\theta^{(i+1)}$ are obtained from maximization of the Q function.
- ⑤ Iterations are repeated until convergence.

Read home

- Shumway and Stoffer, Chapter 6.3

Time Series Analysis

Lecture 8: State Space Model

Stochastic Volatility

Tohid Ardestiri

Linköping University
Division of Statistics and Machine Learning

October 4, 2019



Remaining Course topics

- ARIMA models
- State space models (2 lectures, 1 teaching session with hand-in, 1 computer lab with short report)
 - ▶ Linear and Gaussian state space models (Chapter 6.1)
 - ▶ Kalman filtering, Kalman smoothing and Forecasting (Chapter 6.2)
 - ▶ Maximum likelihood estimate of the state space models (Chapter 6.3)
 - ▶ Stochastic volatility (Chapter 6.11)
- Recurrent Neural Networks (RNNs) (1 lecture and 1 Computer lab No examination)
- Summary lecture

Why Stochastic volatility

$$\begin{aligned}\mathbf{z}_t &= A\mathbf{z}_{t-1} + \mathbf{e}_t, & \mathbf{e}_t &\sim N(0, Q) \\ \mathbf{x}_t &= C\mathbf{z}_t + \nu_t, & \nu_t &\sim N(0, R)\end{aligned}$$

- **Filtering:** Kalman filtering, $f(\mathbf{z}_t | \mathbf{x}_{1:t})$
- **Smoothing:** Kalman smoothing, $f(\mathbf{z}_t | \mathbf{x}_{1:T})$
- **Modelling:** Maximum likelihood and EM, $\hat{\theta} = \arg \max_{\theta} f(\mathbf{x}_{1:T} | \theta)$
- Case study on **Stochastic volatility** via a generalization of the above tools

Stochastic Volatility

In finance, **return** is a profit on an investment. It comprises any change in value of the investment, and/or cash flows which the investor receives from the investment, such as interest payments or dividends.

Stochastic volatility models are those in which the variance of a stochastic process is itself randomly distributed.

In the following:

- r_t denote the **return** of some financial asset. A common model for the return is

$$r_t = \beta \sigma_t \epsilon_t$$

- σ_t is the **volatility process** and
- ϵ_t is an **iid sequence** and $\epsilon_t \sim iid(0, 1)$ and ϵ_t is independent of past σ_s ($s \leq t$)

Stochastic Volatility

In the following:

- r_t denote the **return** of some financial asset. A common model for the return is

$$r_t = \beta \sigma_t \epsilon_t$$

- σ_t is the **volatility process** and
- ϵ_t is an **iid sequence** and $\epsilon_t \sim iid(0, 1)$ and ϵ_t is independent of past σ_s ($s \leq t$)
- Let $\mathbf{z}_t = \log \sigma_t^2$ and consider the hidden autoregressive model

$$\mathbf{z}_t = \phi \mathbf{z}_{t-1} + w_t$$

$$r_t = \beta \exp(\mathbf{z}_t/2) \epsilon_t$$

In this model $w_t \sim iidN(0, \sigma_w^2)$ and ϵ_t is iid noise with finite moments.

$$\mathbf{z}_t = \phi \mathbf{z}_{t-1} + w_t$$

$$r_t = \beta \exp(\mathbf{z}_t/2) \epsilon_t$$

Furthermore, let $\mathbf{x}_t = \log r_t^2$ and $\nu_t = \log \epsilon_t^2$. We obtain

$$\mathbf{x}_t = \alpha + \mathbf{z}_t + \nu_t$$

We can move the α to the state equation and rewrite it as

$$\mathbf{z}_t = \phi_0 + \phi_1 \mathbf{z}_{t-1} + w_t$$

$$\mathbf{x}_t = \mathbf{z}_t + \nu_t$$

where the ϕ_0 is called the leverage effect.

Stochastic Volatility

The distribution of ν_t is not Gaussian because

$$\begin{aligned}\nu_t &= \log \epsilon_t^2 \text{ and} \\ \epsilon_t &\sim iidN(0, 1)\end{aligned}$$

Hence, ν is distributed as a log of a chi-squared distribution with degree of freedom 1 with density

$$f(\nu) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(e^\nu - \nu)\right\} \quad -\infty < \nu < \infty$$

Stochastic Volatility - Gaussian mixture approximation

Instead let us approximate $f(\nu)$ by a Gaussian mixture

$$f(\eta) = \pi_0 N(\eta; 0, \sigma_0^2) + \pi_1 N(\eta; \mu_1, \sigma_1^2)$$

That is,

$$\eta_t = I_t n_{t0} + (1 - I_t) n_{t1}$$

where I_t is an iid Bernoulli process where $Pr\{I = 0\} = \pi_0$ and $Pr\{I = 1\} = \pi_1$, $\pi_0 + \pi_1 = 1$. Also,

$$n_{t0} \sim N(0, \sigma_0^2)$$

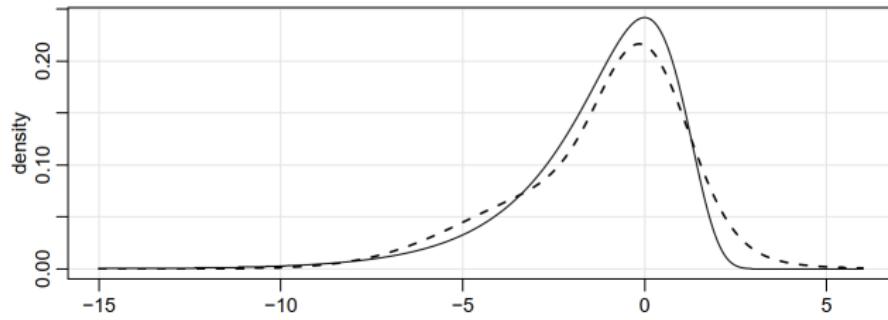
$$n_{t1} \sim N(\mu_1, \sigma_1^2)$$

Stochastic Volatility - Gaussian sum approximation

$$f(\eta) = \pi_0 N(\eta; 0, \sigma_0^2) + \pi_1 N(\eta; \mu_1, \sigma_1^2) \quad -\infty < \eta < \infty$$

$$f(\nu) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(e^\nu - \nu)\right\} \quad -\infty < \nu < \infty$$

$f(\nu)$ and $f(\eta)$ are plotted for comparison. The dashed line is the Gaussian sum approximation, $f(\eta)$.



Stochastic Volatility - Gaussian sum formulation

The problem is finding the filtering distribution of $\mathbf{z}_t | \mathbf{x}_{1:t}$ when

$$\mathbf{z}_t = \phi_0 + \phi_1 \mathbf{z}_{t-1} + w_t$$

$$\mathbf{x}_t = \mathbf{z}_t + \eta_t$$

and

$$w_t \sim iidN(0, \sigma_w^2)$$

$$\eta_t \sim \pi_0 N(0, \sigma_0^2) + \pi_1 N(\mu_1, \sigma_1^2)$$

where $\pi_0 + \pi_1 = 1$

The problem is finding the filtering distribution of $\mathbf{z}_t | \mathbf{x}_{1:t}$ when

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + w_t$$

$$\mathbf{x}_t = C\mathbf{z}_t + \eta_t$$

and

$$w_t \sim iidN(0, Q)$$

$$\eta_t \sim \pi_0 N(\mu_0, R_1) + \pi_1 N(\mu_1, R_2)$$

where $\pi_0 + \pi_1 = 1$

Read home

- Shumway and Stoffer, Chapter 6.11

Time Series Analysis – Lecture 9

Recurrent and Temporal Convolutional Networks

Fredrik Lindsten, Linköping University

2019-10-14

Aim and outline

Aim:

- Introduce two popular deep-learning-based methods for time series analysis
- Highlight some formal connections with classical models (state space and auto-regressive) that you have seen in the course.

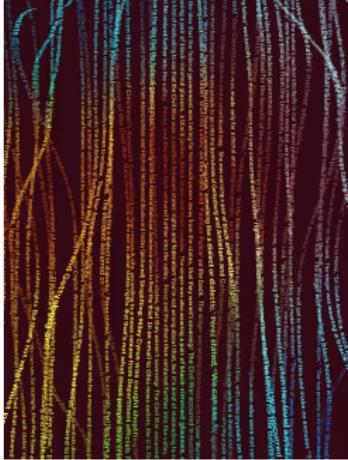
Outline:

1. Basics of neural network models — the multi-layer perceptron
2. Linear Gaussian state space models on innovation form
3. A nonlinear generalization — Recurrent Neural Networks
4. Nonlinear auto-regressive models
5. Temporal Convolutional Networks

ex) Generating text

Input (human-written) In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Model completion (machine-written)



The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. ‘‘By the time we reached the top of one peak, the water looked blue, with some crystals on top,’’ said Pérez.

<https://openai.com/blog/better-language-models/>

State Space Models \Rightarrow
Recurrent Neural Networks

Linear state space models

Linear state space model:

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t,$$

$$x_t = C\mathbf{z}_t + \nu_t.$$

Limitation:

The next state \mathbf{z}_{t+1} as well as the observation x_t depend **linearly** on the current state \mathbf{z}_t .

The model flexibility is limited.

Going nonlinear

Aim: Increase the flexibility of the model by replacing the linear functions by **generic** and **flexible** nonlinear functions.

Linear function: $y = \mathbf{A}\mathbf{z}$, where the matrix \mathbf{A} is the **parameter**.

Nonlinear function: $y = f_{\theta}(\mathbf{z})$. Here, θ is a vector of **parameters** determining the shape of the function $f_{\theta}(\cdot)$.

ex) Let $\theta = (\theta_1, \theta_2, \theta_3)$, and

$$f_{\theta}(z) = \frac{\theta_1}{\theta_2 + z^{\theta_3}}.$$

Neural networks

Recall: We want to use **generic** and **flexible** nonlinear functions.

This is precisely what **neural networks** provide!

Fully connected, 1-layer network:

We **construct** a function $f_{\theta} : \mathbb{R}^p \mapsto \mathbb{R}$ by

$$\begin{aligned}\mathbf{h} &= \sigma(W^{(1)}\mathbf{z} + b^{(1)}) \\ y &= W^{(2)}\mathbf{h} + b^{(2)}.\end{aligned}$$

That is,

$$f_{\theta}(\mathbf{z}) = W^{(2)}\sigma(W^{(1)}\mathbf{z} + b^{(1)}) + b^{(2)}$$

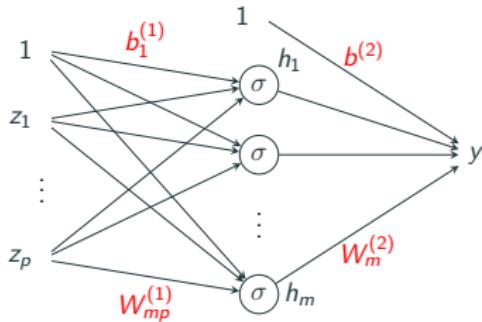
Neural network – graphical illustration

Input variables Hidden units Output

The equations

$$\mathbf{h} = \sigma(W^{(1)}\mathbf{z} + b^{(1)})$$
$$y = W^{(2)}\mathbf{h} + b^{(2)}.$$

can be illustrated graphically.

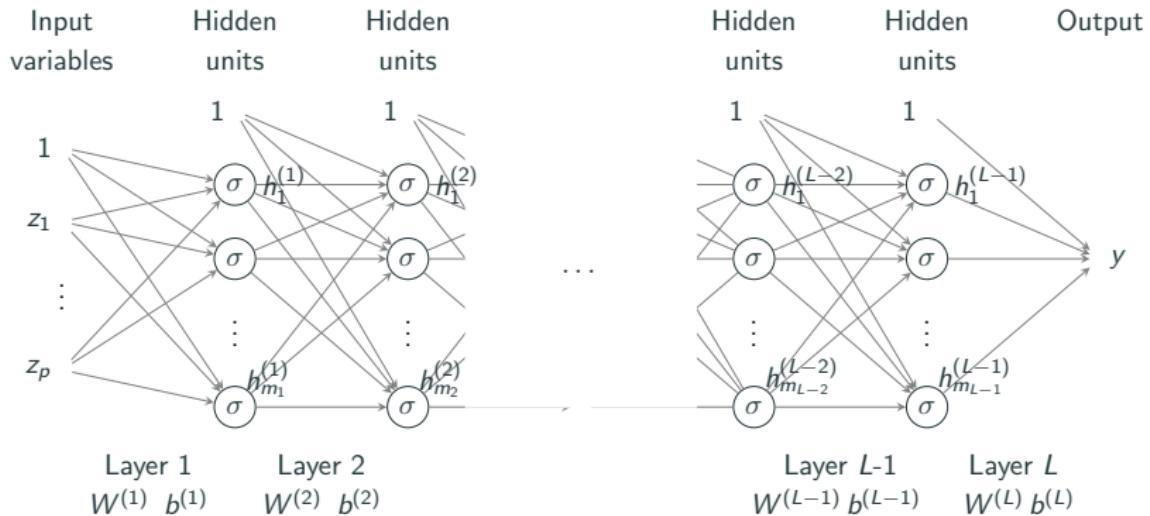


- The variables $\mathbf{h} = (h_1, \dots, h_m)$ are referred to as a **hidden layer**.
- The function $\sigma(\cdot)$ is an element-wise nonlinearity, referred to as an **activation function**. Typical choices are

$$\sigma(x) = \tanh(x) \quad \text{or} \quad \sigma(x) = \text{ReLU}(x) = x \mathbb{1}(x \geq 0)$$

- The model **parameters** are the weight matrices and bias vectors $\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}$.

Multi-layer perceptron



Innovation form

Linear state space model:

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t,$$

$$x_t = C\mathbf{z}_t + \nu_t.$$

Innovation form. There exists an **equivalent** representation given by

$$\mathbf{h}_t = W\mathbf{h}_{t-1} + Ux_{t-1},$$

$$x_t = C\mathbf{h}_t + \nu'_t.$$

(Assuming stationarity for simplicity.)

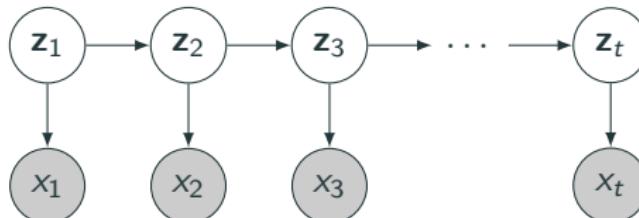
Proof. Let $\mathbf{h}_t = m_{t|t-1}$, the Kalman predictive mean.

Innovation form

Original form:

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t,$$

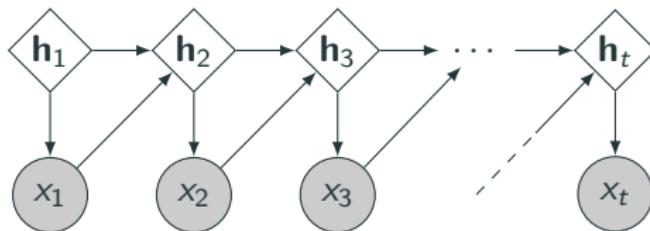
$$x_t = C\mathbf{z}_t + \nu_t.$$



Innovation form:

$$\mathbf{h}_t = W\mathbf{h}_{t-1} + Ux_{t-1},$$

$$x_t = C\mathbf{h}_t + \nu'_t.$$



The hidden state variables can be **deterministically and recursively computed** from the data.

Going nonlinear

Doesn't this look suspiciously similar to an MLP...?

$$\mathbf{h}_t = W\mathbf{h}_{t-1} + Ux_{t-1},$$

$$x_t = C\mathbf{h}_t + \nu'_t,$$

for some **nonlinear activation function** $\sigma(\cdot)$.

This is a basic **Recurrent Neural Network (RNN)**.

Learning the parameters

The model parameters are the weight matrices and bias vectors:

$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}x_{t-1} + \mathbf{b}),$$

$$x_t = \mathbf{C}\mathbf{h}_t + \mathbf{c} + \nu'_t,$$

with $\theta = \{\mathbf{W}, \mathbf{U}, \mathbf{b}, \mathbf{C}, \mathbf{c}\}$.

Note:

- The parameters are the same for all time steps (“weight sharing”).
- The fact that there is no state noise means that we can compute

$$p_\theta(x_t | x_{1:t-1}) = N(x_t | \mathbf{C}\mathbf{h}_t + \mathbf{c}, \sigma_{\nu'}^2).$$

Learning the parameters

We can thus learn the parameters θ directly by optimizing the negative log-likelihood,

$$L(\theta; x_{1:T}) = - \sum_{t=1}^T \log p_\theta(x_t | x_{1:t-1}),$$

using gradient-based optimization.

The gradient $\nabla_\theta L(\theta; x_{1:T})$ is computed using the chain rule of differentiation, propagating information from $t = 1$ to $t = T$ and then back again.

⇒ Back-propagation through time.

RNN extensions

- GRU/LSTM
- Non-Gaussian likelihood (e.g., for discrete data)
- Conditioning on context (input)
- Stochastic hidden layers
- Bidirectional connections
- ...

Autoregressive Models \Rightarrow
Temporal Convolutional Nets

Autoregressive models

State space models and RNNs use a latent state vector to model temporal dependencies.

An alternative is to model the dependency of the current data point x_t on the past data points $x_{1:t-1}$ by a **direct functional relationship**.

Auto-regressive model, AR(p):

$$x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + w_t, \quad w_t \sim N(0, \sigma_w^2).$$

The AR model is linear in the parameters:

- ▲ Learning of parameters easy \Leftrightarrow linear regression
- ▼ Flexibility/ability to model complex temporal dependencies is limited
- ▼ Memory/receptive field is just p time steps

Going nonlinear

Nonlinear auto-regressive model, NAR(p):

$$x_t = \sigma(\phi_1 x_{t-1} + \cdots + \phi_p x_{t-p}) + w_t, \quad w_t \sim N(0, \sigma_w^2),$$

for some nonlinear activation function σ .

- ▲ Flexibility increased...
- ▼ ...but only slightly!
- ▼ Memory/receptive field is *still* just p steps

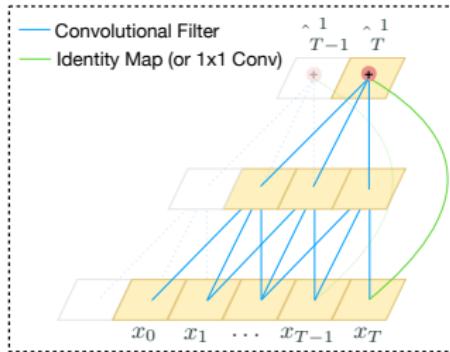
We can address these issues with a **multi-layer network architecture!**

Temporal Convolutional Network

2-layer TCN:

$$h_{t-1} = \sigma(\phi_1^{(1)}x_{t-1} + \cdots + \phi_p^{(1)}x_{t-p}),$$
$$x_t = \sigma(\phi_1^{(2)}h_{t-1} + \cdots + \phi_p^{(2)}h_{t-p}) + w_t,$$

with $w_t \sim N(0, \sigma_w^2)$.

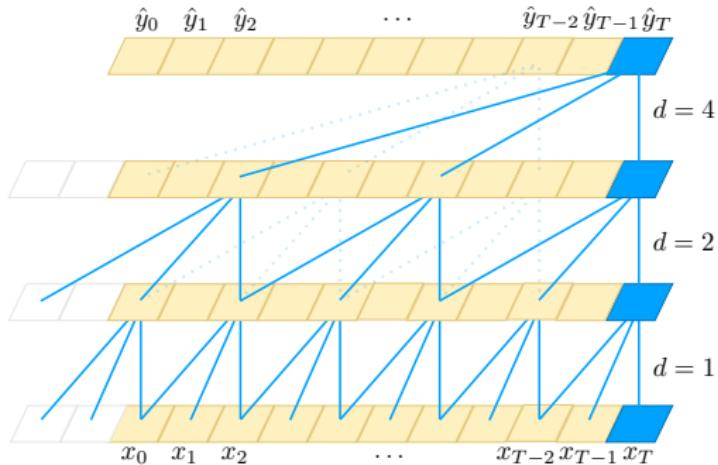


Can extend to multiple hidden layers, $h_t^{(1)}, h_t^{(2)}, \dots$

- ▲ Multiple layers \Rightarrow very flexible models
- ▲ Receptive field increases with depth...
- ▼ ... but only linearly

TCN with dilated convolutions

By using **dilated convolutions** we can increase receptive field **exponentially** with depth.



RNN vs TCN

RNNs are still the *de facto* standard deep learning approach to time series modeling, *but...*

...TCNs have outperformed them on many benchmark problems

 Shaojie Bai, J. Zico Kolter, Vladlen Koltun. **An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.** arXiv.org: 1803.01271, 2018.

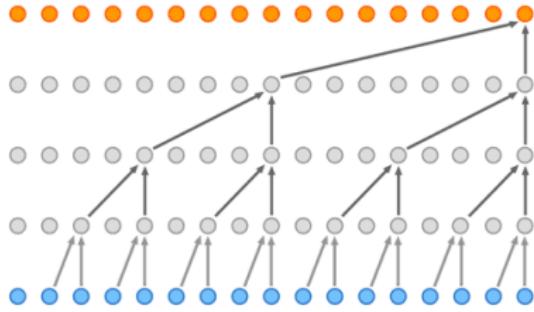
...and have other advantages too: ... but also some disadvantages:

- Easier parallelization
- Better control over receptive field
- Lower memory requirements during training
- ...
- More difficult to reuse model for multiple tasks
- Larger memory requirements after deployment
- ...

ex) WaveNet



WaveNet by DeepMind powers
Google's text-to-speech technology.



Deep Learning for TSA

So is this what we should always do for modeling sequential data?!

No!

- Only makes sense to use something as complex as RNN or TCN when classical methods fail — **Try Simple Things First!**
- Methods based on deep learning work best if we have multiple sequences, or one long sequence that can be split into segments
- For a single univariate time series, classical methods (ARIMA, state space, ...) often work better.

1 Lectures 1-3

- Probability density function for x : $f(x)$
- Marginal density $f_i(x_i) = \int f(x) dx_1 \dots dx_{i-1} dx_{i+1} \dots dx_p$
- Expected (mean) value $Ex = \int xf(x)dx$
- Covariance $\text{cov}(x, y) = E\{(x - Ex)(y - Ey)\}$
- Correlation $\rho_{x,y} = \text{corr}(x, y) = \frac{\text{cov}(x,y)}{\sigma_x \sigma_y}$
- Variance $\text{var}(x) = E\{(x - Ex)^2\} = \text{cov}(x, x)$
- Relationships (a is a constant)
 - $E(x + a) = Ex + a$, $E(ax) = aEx$
 - $E(x + y) = Ex + Ey$
 - $\text{cov}(x + a, y) = \text{cov}(x, y)$
 - $\text{cov}(x + z, y) = \text{cov}(x, y) + \text{cov}(z, y)$
 - $\text{var}(ax) = a^2 \text{var}(x)$

uncorrelated $\iff E(XY) = EX.EY$
 independent $\iff f_{X,Y}(x, y) = f_X(x).f_Y(y)$

- Autocovariance function

$$\gamma(s, t) = \text{cov}(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)]$$

Note: $\text{var}(x_t) = \gamma(t, t)$

- Autocorrelation function (ACF)

$$\rho(s, t) = \frac{\gamma(s, t)}{\sqrt{\gamma(s, s)\gamma(t, t)}}$$

Useful fact: If $U = \sum_{j=1}^m a_j x_j$ and

$$V = \sum_{k=1}^r b_k y_k$$

$$\text{cov}(U, V) = \sum_{j=1}^m \sum_{k=1}^r a_j b_k \text{cov}(x_j, y_k)$$

1.1 stationarity

- Time series x_t is weakly stationary (stationary) if
 - $Ex_t = \text{const}$
 - $\gamma(s, t) = \gamma(|s - t|)$
 - $\text{var}(x_t) < \infty$
- $\gamma(t, t + h) = \gamma(|t + h - t|) = \gamma(h)$
 - Autocovariance depends on lag only!
- Autocovariance for stationary process
 $\gamma(h) = \text{cov}(x_t, x_{t+h})$
- ACF for stationary process $\rho(h) = \frac{\gamma(h)}{\gamma(0)}$

Properties of stationary process:

$$\gamma(h) = \gamma(-h) \quad \rho(h) = \rho(-h)$$

$$|\gamma(h)| \leq \gamma(0) \quad \rho(h) \leq 1, \rho(0) = 1$$

If x_t is stationary,

- Sample mean

$$Ex \approx \bar{x} = \frac{1}{n} \sum_{t=1}^n x_t$$

- Sample autocovariance function

$$\hat{\gamma}(h) = \frac{1}{n} \sum_{t=1}^{n-h} (x_{t+h} - \bar{x})(x_t - \bar{x})$$

Theorem: Under weak conditions,
 if x_t is white noise and $n \rightarrow \infty$
 then $\hat{\rho}(h)$ is approximately $N(0, \frac{1}{n})$

Consequence: If some $|\hat{\rho}(h)| > \frac{2}{\sqrt{n}}$ then the time series is not a white noise (with approximately 95 % confidence).

1.2 Backshift operator

- Backshift operator $Bx_t = x_{t-1}$,
Powers $B^k x_t = x_{t-k}$
- Forward-shift operator $B^{-1}x_t = x_{t+1}$
- Note $BB^{-1}x_t = x_t$ (i.e. $BB^{-1} = 1$)
- Differencing $\nabla x_t = (1 - B)x_t$
- Differences of order d : $\nabla^d = (1 - B)^d$
- Property: Operators can be manipulated as polynomials
- Example Check that $\nabla^2 x_t = x_t - 2x_{t-1} + x_{t-2}$
- Property: Differencing of order p can remove polynomial trend of order p

• Autoregressive operator

$$\phi(B) = 1 - \phi_1 B - \dots - \phi_p B^p$$

- AR(p) model

$$\boxed{\phi(B)x_t = w_t}$$

• ARMA(p,q)

$$\begin{aligned} x_t = & \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} \\ & + w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q} \\ - & \phi_p \neq 0, \theta_q \neq 0 \\ - & \text{Is stationary} \\ - & E x_t = 0 \end{aligned}$$

1.3 MA, AR, ARMA

- Moving average model of order q, MA(q)

$$\begin{aligned} x_t = & w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q} \\ = & \sum_{j=0}^q \theta_j w_{t-j} \end{aligned}$$

- $w_t \sim \text{wn}(0, \sigma_w^2)$
- $\theta_1, \dots, \theta_q$ constants, $\theta_q \neq 0$ and $\theta_0 = 1$

- Moving average operator

$$\theta(B) = \sum_{j=0}^q \theta_j B^j$$

- MA(q):

$$\boxed{x_t = \theta(B)w_t}$$

- Autoregressive model of order p, AR(p)

$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t$$

- x_t is stationary if x_0 is sampled from the stationary distribution
- $w_t \sim \text{wn}(0, \sigma_w^2)$
- ϕ_1, \dots, ϕ_p constants, $\phi_p \neq 0$
- $E x_t = 0$

1.4 Causality / invertibility

A stationary process is **causal** if it is only dependent on the past values of the process

Def: A linear process is **nonexplosive** and **causal** if it can be written as a one-sided sum:

$$x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j} = \psi(B)w_t$$

where $\psi(B) = \sum_{j=0}^{\infty} \psi_j B^j$ and $\sum_{j=0}^{\infty} |\psi_j| < \infty$.

Def: An MA process is **invertible** if it has a causal AR representation,

$$w_t = \sum_{j=0}^{\infty} \pi_j x_{t-j}$$

Def: Linear process is **causal** and **nonexplosive** if

- $x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j}$ (depends on the past only)
- $\sum_{j=0}^{\infty} |\psi_j| < \infty$
- We set $\psi_0 = 1$ by convention.

Property: ARMA(p,q) is **causal** iff roots $\phi(z') = 0$ are outside unit circle, i.e. $|z'| > 1$

$$\boxed{\phi(B)x_t = \theta(B)w_t}$$

Def: ARMA(p,q) is **invertible** if

- $w_t = \sum_{j=0}^{\infty} \pi_j x_{t-j}$ (depends on the past only)
- $\sum_{j=0}^{\infty} |\pi_j| < \infty$

Property: ARMA(p,q) is **invertible** iff roots $\theta(z') = 0$ are outside unit circle, i.e. $|z'| > 1$

$$\boxed{\phi(B)x_t = \theta(B)w_t}$$

Time Series Analysis

Teaching session III : State Space Models, Kalman filtering
Kalman Smoothing

Tohid Ardesthiri

Linköping University
Division of Statistics and Machine Learning

September 30, 2019



Remaining Course topics

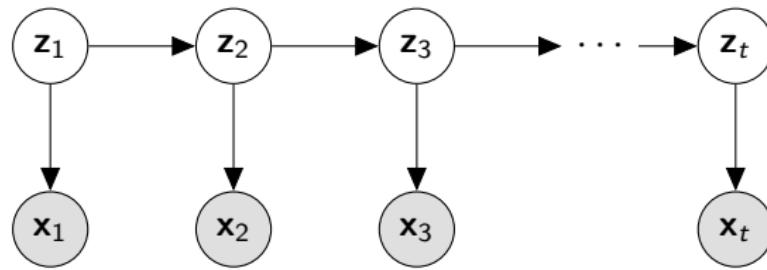
- ARIMA models
- State space models (2 lectures, 1 teaching session with hand-in, 1 computer lab with short report)
 - ▶ Linear and Gaussian state space models (Chapter 6.1)
 - ▶ Kalman filtering, Kalman smoothing and Forecasting (Chapter 6.2)
 - ▶ Maximum likelihood estimate of the state space models (Chapter 6.3)
 - ▶ Stochastic volatility (Chapter 6.11)
- Recurrent Neural Networks (RNNs) (1 lecture and 1 Computer lab No examination)
- Summary lecture

State Space models - Linear and Gaussian

Our main focus will be on linear and Gaussian models:

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t, \quad e_t \sim N(0, Q)$$

$$\mathbf{x}_t = C\mathbf{z}_t + \nu_t, \quad \nu_t \sim N(0, R)$$



Bayesian Inference

Bayesian inference is a means of combining prior beliefs with the data (evidence) to obtain posterior beliefs.

Example: likelihood update

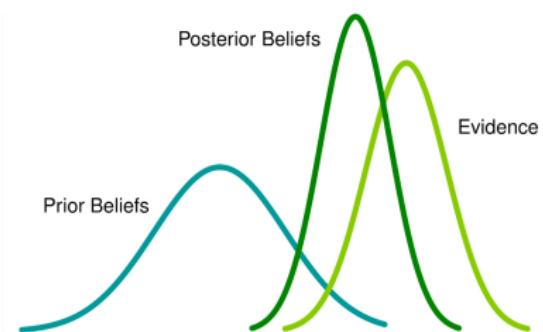
$$f(z|x) \propto f(x|z)f(z)$$

Probability Calculus

$$f(z, x) = f(z|x)f(x)$$

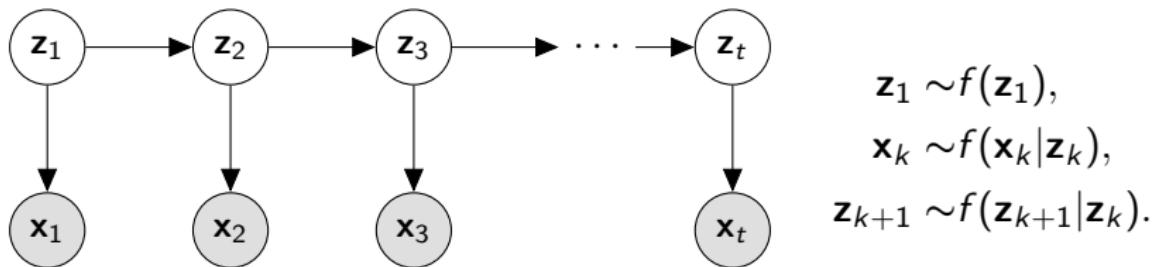
$$f(z, x) = f(x|z)f(z)$$

$$f(z) = \int f(z, x) dx$$



Online recursive algorithms

Consider a stochastic dynamical system represented by the following recursion



The Bayesian filtering recursion corresponds to computing the posterior distributions $f(z_k|x_{1:k})$;

$$f(z_k|x_{1:k}) = \frac{f(z_k|x_{1:k-1})f(x_k|z_k)}{\int f(z_k|x_{1:k-1})f(x_k|z_k) dz_k}$$

The density $f(z_k|x_{1:k-1})$ in the numerator which is called the predicted density of z_k and is obtained by integration as in

$$f(z_k|x_{1:k-1}) = \int f(z_k|z_{k-1})f(z_{k-1}|x_{1:k-1}) dz_{k-1}.$$

Kalman filter

Kalman filter is an algorithm that uses time series data, **containing statistical noise and unknown innovations**, and produces estimates of latent (hidden) process that tend to be more accurate than those based on a single observations using a probabilistic framework.

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + \mathbf{e}_t,$$

$$\mathbf{x}_t = C\mathbf{z}_t + \mathbf{\nu}_t,$$



The Kalman Filter's Foundation

Let \mathbf{z} have a normal prior distribution with mean μ and covariance Σ , i.e., $\mathbf{z} \sim N(\mathbf{z}; \mu, \Sigma)$.

An observation \mathbf{x} with the likelihood function $f(\mathbf{x}|\mathbf{z}) = N(\mathbf{x}; C\mathbf{z}, R)$ is in hand where C is a matrix with proper dimensions and R is a covariance matrix. The posterior distribution of \mathbf{z} can be obtained using the Bayes' rule

$$\begin{aligned} f(\mathbf{z}|\mathbf{x}) &= \frac{f(\mathbf{z})f(\mathbf{x}|\mathbf{z})}{\int f(\mathbf{z})f(\mathbf{x}|\mathbf{z}) d\mathbf{z}} \\ &= \frac{N(\mathbf{z}; \mu, \Sigma)N(\mathbf{x}; C\mathbf{z}, R)}{\int N(\mathbf{z}; \mu, \Sigma)N(\mathbf{x}; C\mathbf{z}, R) d\mathbf{z}}. \end{aligned}$$

The posterior distribution $f(\mathbf{z}|\mathbf{x})$ has an analytical solution and turns out to be the normal distribution $N(\mathbf{z}; \mu', \Sigma')$ where

$$\begin{aligned} \mu' &= \mu + K(\mathbf{x} - C\mu), \\ \Sigma' &= \Sigma - KC\Sigma, \end{aligned}$$

where

$$K = \Sigma C^T (C\Sigma C^T + R)^{-1}.$$

Properties of the normal density function

Property 1: $f(\mathbf{y}_1)f(\mathbf{y}_2|\mathbf{y}_1) = f(\mathbf{y}_1, \mathbf{y}_2)$

$$N(\mathbf{y}_1; \mu, \Sigma)N(\mathbf{y}_2; B\mathbf{y}_1, R) = N\left(\begin{bmatrix}\mathbf{y}_1 \\ \mathbf{y}_2\end{bmatrix}; \begin{bmatrix}\mu \\ B\mu\end{bmatrix}, \begin{bmatrix}\Sigma & \Sigma B^T \\ B\Sigma & B\Sigma B^T + R\end{bmatrix}\right)$$

Property 2: marginalization and conditioning

If $\mathbf{y}_1, \mathbf{y}_2$ were jointly normal:

$$f(\mathbf{y}_1, \mathbf{y}_2) = N\left(\begin{bmatrix}\mathbf{y}_1 \\ \mathbf{y}_2\end{bmatrix}; \begin{bmatrix}\mu_1 \\ \mu_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right)$$

then

$$f(\mathbf{y}_1) = N(\mathbf{y}_1; \mu_1, \Sigma_{11})$$

$$f(\mathbf{y}_2) = N(\mathbf{y}_2; \mu_2, \Sigma_{22})$$

$$f(\mathbf{y}_1|\mathbf{y}_2) = N(\mathbf{y}_1; \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{y}_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})$$

$$f(\mathbf{y}_2|\mathbf{y}_1) = N(\mathbf{y}_2; \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{y}_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})$$

Kalman filter's derivation

Consider State space model

$$\begin{aligned}\mathbf{z}_t &= A\mathbf{z}_{t-1} + \mathbf{e}_t, \\ \mathbf{x}_t &= C\mathbf{z}_t + \nu_t.\end{aligned}$$

And initial prior on the state \mathbf{z}_1

$$f(\mathbf{z}_1) = N(\mathbf{z}_1; m_0, P_0)$$

We want to derive a recursive algorithm to compute the posterior filtering density

$$f(\mathbf{z}_t | \mathbf{x}_{1:t}).$$

That is, computing the the posterior density of \mathbf{z}_t using the observations up to time t .

Kalman filter's derivation

Assume that we have

$$f(\mathbf{z}_t | \mathbf{x}_{1:t}) = N(\mathbf{z}_t; m_{t|t}, P_{t|t}).$$

The state transition density $f(\mathbf{z}_{t+1} | \mathbf{z}_t)$ and the likelihood function $f(\mathbf{x}_{t+1} | \mathbf{z}_{t+1})$ can be written as

$$\begin{aligned} f(\mathbf{z}_{t+1} | \mathbf{z}_t) &= N(\mathbf{z}_{t+1}; A\mathbf{z}_t, Q), \\ f(\mathbf{x}_{t+1} | \mathbf{z}_{t+1}) &= N(\mathbf{x}_{t+1}; C\mathbf{z}_{t+1}, R). \end{aligned}$$

Therefore, the joint posterior $f(\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{x}_{t+1} | \mathbf{x}_{1:t})$ can be written as

$$\begin{aligned} f(\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{x}_{t+1} | \mathbf{x}_{1:t}) &= N(\mathbf{z}_t; m_{t|t}, P_{t|t}) \\ &\quad \times N(\mathbf{z}_{t+1}; A\mathbf{z}_t, Q)N(\mathbf{x}_{t+1}; C\mathbf{z}_{t+1}, R), \end{aligned}$$

Kalman filter's derivation

The $f(\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{x}_{t+1} | \mathbf{x}_{1:t})$ can be rewritten in matrix form as

$$f(\mathbf{z}_t, \mathbf{z}_{t+1}, \mathbf{x}_{t+1} | \mathbf{x}_{1:t}) = N([\mathbf{z}_t^T, \mathbf{z}_{t+1}^T, \mathbf{x}_{t+1}^T]^T; \mu_t, \Sigma_t),$$

where

$$\mu_t = \begin{bmatrix} \mu_1 \\ \hline \mu_2 \end{bmatrix} = \begin{bmatrix} m_{t|t} \\ \hline Am_{t|t} \\ \hline CAM_{t|t} \end{bmatrix}$$

and

$$\Sigma_t \triangleq \left[\begin{array}{c|c} \Sigma_{11} & \Sigma_{12} \\ \hline \Sigma_{21} & \Sigma_{22} \end{array} \right] = \left[\begin{array}{cc|c} P_{t|t} & P_{t|t}A^T & (P_{t|t}A^T)C^T \\ AP_{t|t} & AP_{t|t}A^T + Q & (AP_{t|t}A^T + Q)^TC^T \\ \hline C(AP_{t|t}) & C(AP_{t|t}A^T + Q) & C(AP_{t|t}A^T + Q)C^T + R \end{array} \right].$$

Kalman filtering algorithm

Prove the Kalman filtering recursion for the following state space model with initial prior on the state $f(\mathbf{z}_1) = N(\mathbf{z}_1; \mathbf{m}_0, \mathbf{P}_0)$

$$\mathbf{z}_t = \mathbf{A}_{t-1} \mathbf{z}_{t-1} + \mathbf{e}_t, \quad \mathbf{e}_t \sim N(0, \mathbf{Q}_t)$$

$$\mathbf{x}_t = \mathbf{C}_t \mathbf{z}_t + \nu_t, \quad \nu_t \sim N(0, \mathbf{R}_t)$$

1: **Inputs:** \mathbf{A}_t , \mathbf{C}_t , \mathbf{Q}_t , \mathbf{R}_t , \mathbf{m}_0 , \mathbf{P}_0 and $\mathbf{x}_{1:T}$.

initialization

2: $\mathbf{m}_{1|0} \leftarrow \mathbf{m}_0$, $\mathbf{P}_{1|0} \leftarrow \mathbf{P}_0$

3: **for** $t = 1$ to T **do**

observation update step

4: $\mathbf{K}_t \leftarrow \mathbf{P}_{t|t-1} \mathbf{C}_t^T (\mathbf{C}_t \mathbf{P}_{t|t-1} \mathbf{C}_t^T + \mathbf{R}_t)^{-1}$

5: $\mathbf{m}_{t|t} \leftarrow \mathbf{m}_{t|t-1} + \mathbf{K}_t (\mathbf{x}_t - \mathbf{C}_t \mathbf{m}_{t|t-1})$

6: $\mathbf{P}_{t|t} \leftarrow (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t) \mathbf{P}_{t|t-1}$

prediction step

7: $\mathbf{m}_{t+1|t} \leftarrow \mathbf{A}_t \mathbf{m}_{t|t}$

8: $\mathbf{P}_{t+1|t} \leftarrow \mathbf{A}_t \mathbf{P}_{t|t} \mathbf{A}_t^T + \mathbf{Q}_{t+1}$

9: **end for**

10: **Outputs:** $\mathbf{m}_{t|t}$, $\mathbf{P}_{t|t}$ for $t = 1 : T$

Bayesian Smoothing

The purpose of Bayesian smoothing is to compute the marginal posterior distribution of \mathbf{z}_t at time t after receiving observations up to time T where $T > t$:

$$f(\mathbf{z}_t | \mathbf{x}_{1:T})$$

The Rauch-Tung-Striebel smoother (RTS smoother) which is also called the Kalman smoother is used to compute

$$f(\mathbf{z}_t | \mathbf{x}_{1:T}) = N(\mathbf{z}_t; m_{t|T}, P_{t|T})$$

The RTS smoother uses a Kalman filter in its forward path. In its backwards path it updates the densities using the relation

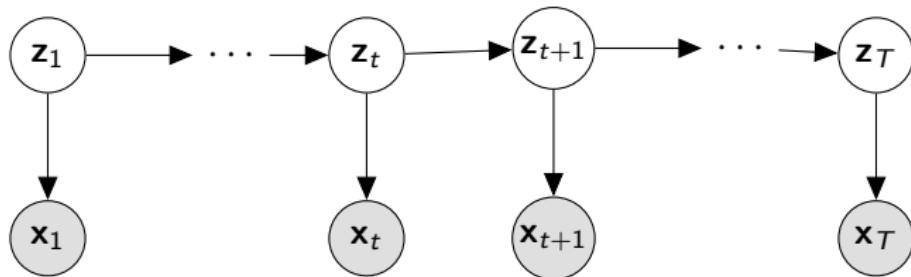
$$\mathbf{z}_t = A_{t-1} \mathbf{z}_{t-1} + e_t$$

RTS Smoother's derivation

Assume $f(\mathbf{z}_{t+1}|\mathbf{x}_{1:T})$ is available as in

$$f(\mathbf{z}_{t+1}|\mathbf{x}_{1:T}) = N(\mathbf{z}_{t+1}; \mathbf{m}_{t+1|T}, \mathbf{P}_{t+1|T})$$

For example $f(\mathbf{z}_T|\mathbf{x}_{1:T})$ which is the filtering density of \mathbf{z}_T is available after filtering.



The objective is to compute $f(\mathbf{z}_t, \mathbf{z}_{t+1}|\mathbf{x}_{1:T})$.

RTS Smoother's derivation

The joint posterior $f(\mathbf{z}_t, \mathbf{z}_{t+1} | \mathbf{x}_{1:t})$ can be written as

$$\begin{aligned} f(\mathbf{z}_t, \mathbf{z}_{t+1} | \mathbf{x}_{1:t}) &= N(\mathbf{z}_t; m_{t|t}, P_{t|t}) N(\mathbf{z}_{t+1}; A\mathbf{z}_t, Q) \\ &= N\left(\begin{bmatrix} \mathbf{z}_t \\ \mathbf{z}_{t+1} \end{bmatrix}, \begin{bmatrix} m_{t|t} \\ Am_{t|t} \end{bmatrix}, \begin{bmatrix} P_{t|t} & P_{t|t}A^T \\ AP_{t|t} & AP_{t|t}A^T + Q \end{bmatrix}\right) \end{aligned}$$

Using the conditioning property of the multivariate normal distribution $f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t})$ can be computed as a normal density as given in the following:

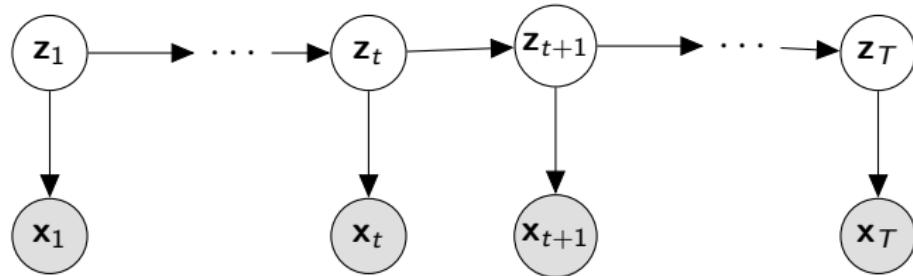
$$f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t}) = N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t)$$

where \tilde{m}_t is a function of \mathbf{z}_{t+1} .

RTS Smoother's derivation

Note the Markov property

$$f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:T}) = f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t})$$



Assume $f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T})$ is available as in

$$f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T}) = N(\mathbf{z}_{t+1}; m_{t+1|T}, P_{t+1|T})$$

Recall that

$$\begin{aligned} f(\mathbf{z}_{t+1}, \mathbf{z}_t | \mathbf{x}_{1:T}) &= f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T}) f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:T}) \\ &= f(\mathbf{z}_{t+1} | \mathbf{x}_{1:T}) f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t}) \\ &= N(\mathbf{z}_{t+1}; m_{t+1|T}, P_{t+1|T}) N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t) \end{aligned}$$

RTS Smoother's derivation

where

$$G_t = P_{t|t} A_t^T (A P_{t|t} A^T + Q)^{-1} = P_{t|t} A_t^T P_{t+1|t}^{-1}$$

$$\tilde{m}_t = m_{t|t} + G_t (\mathbf{z}_{t+1} - A m_{t|t})$$

$$\tilde{P}_t = P_{t|t} - G_t (A P_{t|t} A^T + Q) G_t^T = P_{t|t} - G_t P_{t+1|t} G_t^T$$

Hence,

$$\begin{aligned} f(\mathbf{z}_{t+1}, \mathbf{z}_t | \mathbf{x}_{1:T}) &= N(\mathbf{z}_{t+1}; m_{t+1|T}, P_{t+1|T}) N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t) \\ &= N\left(\begin{bmatrix} \mathbf{z}_t \\ \mathbf{z}_{t+1} \end{bmatrix}, \begin{bmatrix} \cdot & \cdot \\ m_{t+1|T} & \cdot \\ \cdot & P_{t+1|T} \end{bmatrix}\right) \end{aligned}$$

RTS smoother's backwards recursion

Prove the backwards recursion of the RTS smoother for following state space model with initial prior on the state $f(\mathbf{z}_1) = N(\mathbf{z}_1; \mathbf{m}_0, \mathbf{P}_0)$

$$\mathbf{z}_t = A_{t-1}\mathbf{z}_{t-1} + e_t, \quad e_t \sim N(0, Q_t)$$

$$\mathbf{x}_t = C_t\mathbf{z}_t + \nu_t, \quad \nu_t \sim N(0, R_t)$$

1: **Inputs:** $A_t, Q_t, m_{t|t}, P_{t|t}, m_{t+1|t}, P_{t+1|t}$ for $1 \leq t \leq T$
initialization

2: **for** $t = T-1$ down to 1 **do**

3: $G_t \leftarrow P_{t|t}A_t^T P_{t+1|t}^{-1}$

4: $m_{t|T} \leftarrow m_{t|t} + G_t(m_{t+1|T} - A_t m_{t|t})$

5: $P_{t|T} \leftarrow P_{t|t} + G_t(P_{t+1|T} - P_{t+1|t})G_t^T$

6: **end for**

7: **Outputs:** $m_{t|T}, P_{t|T}$

Read home

- Shumway and Stoffer, Chapters 6.1 and 6.2

Time Series Analysis

Teaching Session II: ARIMA models-3

Seasonal models

Tohid Ardesthiri

Linköping University
Division of Statistics and Machine Learning

September 18, 2019



Seasonal ARMA

- Seasonal patterns
 - ▶ Yearly (ocean temperature)
 - ▶ Daily, weekly (Server workload)
- Strong correlation of x_t and x_{t+s}
 - ▶ $s = 12, 24, \dots$
- Applications
 - ▶ Physics, biology, economics, computer science

Seasonal ARMA

- Pure seasonal $ARMA(P, Q)_s$

$$\Phi_P(B^s)x_t = \theta_Q(B^s)w_t$$

- Seasonal autoregressive operator

$$\Phi_P(B^s) = 1 - \Phi_1(B^{(1\cdot s)}) - \dots - \Phi_P B^{P\cdot s}$$

- Seasonal moving average operator

$$\Theta_Q(B^s) = 1 + \Theta_1(B^{(1\cdot s)}) + \dots + \Theta_Q B^{Q\cdot s}$$

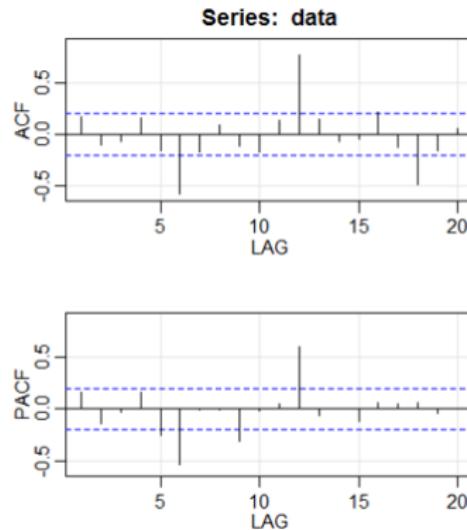
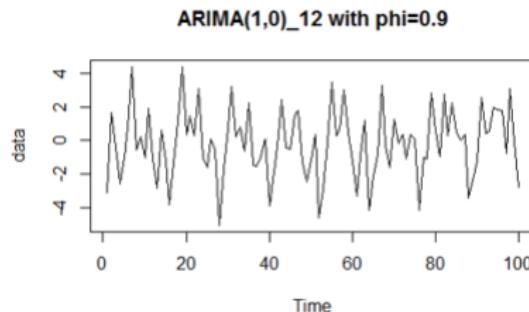
- Same principles for causality and invertibility

- Example: $ARMA(1, 0)_{12}$ and $ARMA(0, 1)_{12}$

- ▶ Autocovariance

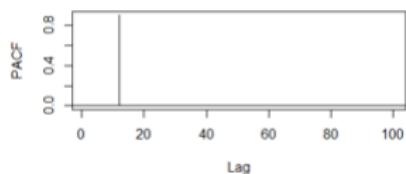
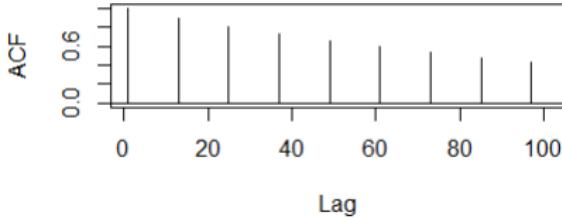
Seasonal ARMA

- Example: Simulated $ARMA(1, 0)_{12}$, $\Phi = 0.9$



Seasonal ARMA

- **Example:** Simulated $ARMA(1, 0)_{12}$, $\Phi = 0.9$
 - ▶ Theoretical ones



Seasonal ARMA

	$\text{AR}(P)_s$	$\text{MA}(Q)_s$	$\text{ARMA}(P, Q)_s$
ACF*	Tails off at lags ks , $k = 1, 2, \dots,$	Cuts off after lag Qs	Tails off at lags ks
PACF*	Cuts off after lag Ps	Tails off at lags ks $k = 1, 2, \dots,$	Tails off at lags ks

*The values at nonseasonal lags $h \neq ks$, for $k = 1, 2, \dots$, are zero.

Multiplicative seasonal ARMA

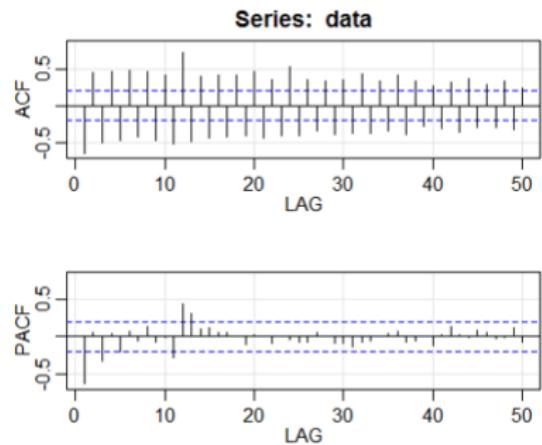
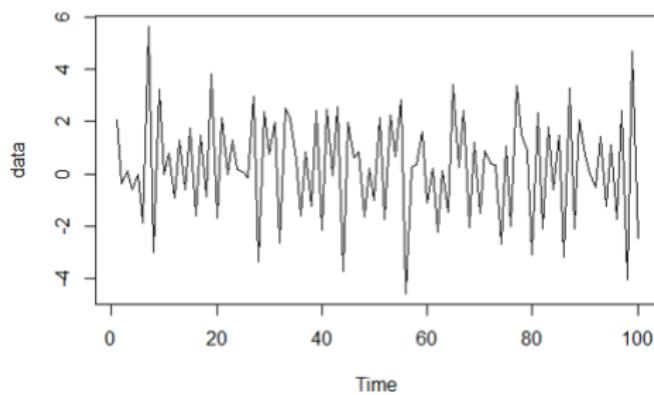
- **Problem:** in real data, it is hard to assume x_t is dependent on x_{t-kh} only...
 - ▶ Combinal seasonal and nonseasonal!
- Multiplicative Seasonal ARMA(p, q) \times (P, Q)_s

$$\Phi_P(B^s)\phi(B)x_t = \Theta_Q(B^s)\theta(B)w_t$$

- **Example** Expression for ARMA(1, 1) \times (1, 0)_s

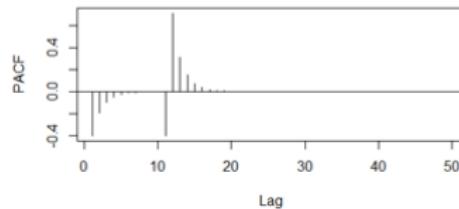
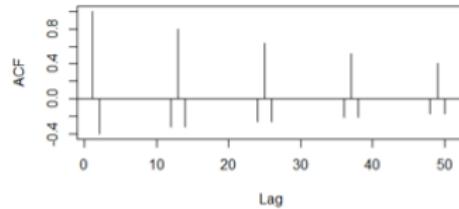
Multiplicative seasonal ARMA

- Example $x_t = 0.8x_{t-12} + w_t - 0.5w_{t-1}$



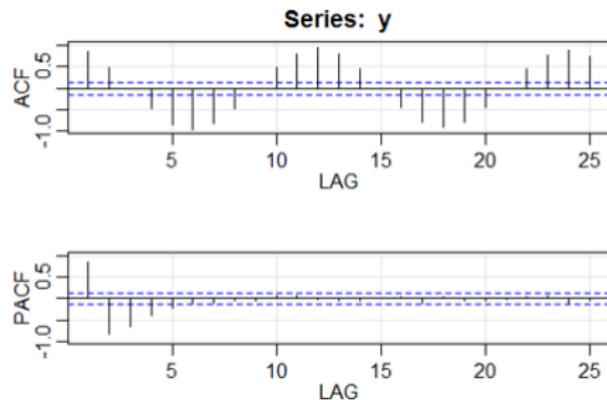
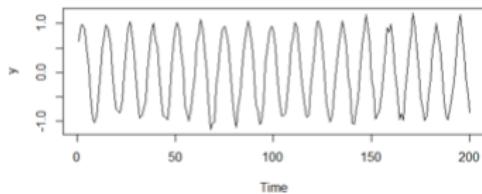
Multiplicative seasonal ARMA

- Example $x_t = 0.8x_{t-12} + w_t - 0.5w_{t-1}$
 - ▶ Theoretical



SARIMA

- What if there is a seasonal pattern which differs a little between the series



Note: ACF almost decays very slowly at peaks 12h

SARIMA

- Multiplicative seasonal autoregressive integrated moving average model $ARIMA(p, d, q) \times (P, D, Q)_s$

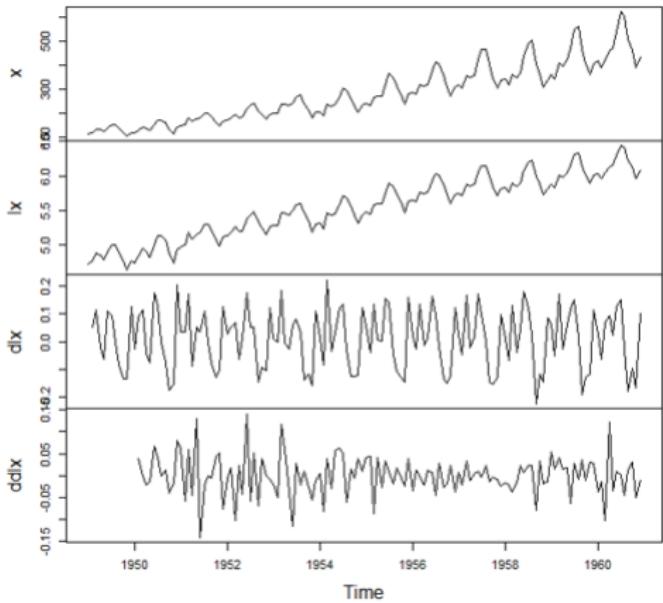
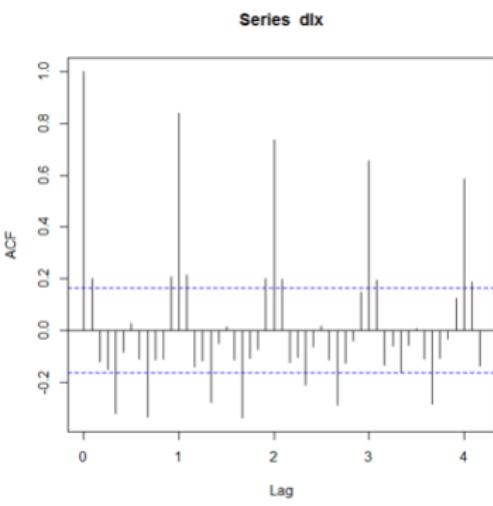
$$\Phi_p(B^s)\phi(B)\nabla_s^D\nabla^d x_t = \delta + \Theta_Q(B^s)\theta(B)w_t$$

$$\nabla_s^D = (1 - B^s)^D$$

- How to identify SARIMA?
 - ① Perform differencing first (trend)
 - ② Investigate ACF → slowly decays at peaks?
 - ① Yes → Additional differencing by ∇_s^D
 - ③ Model non-seasonal part
 - ④ Model seasonal part (check peaks), check ACF and PACF of residuals

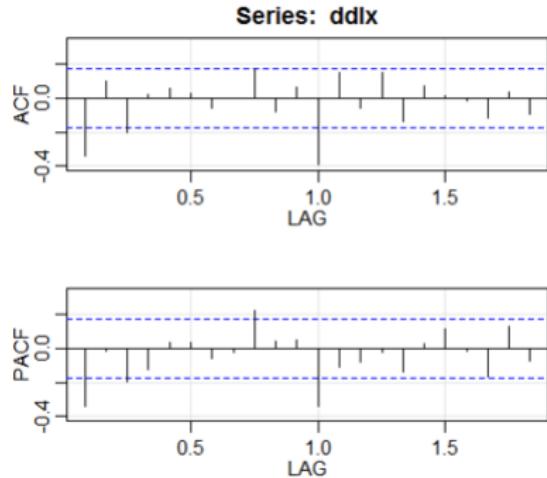
SARIMA

- Example: Air passengers



SARIMA

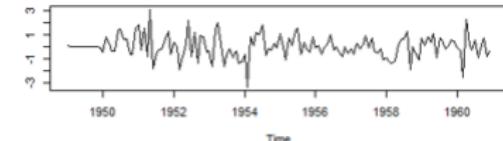
- Example: Air passengers



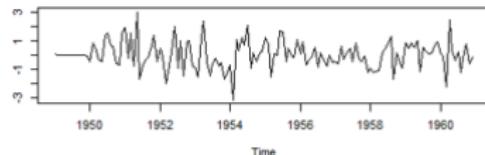
$(0, 1, 1)_{12}$ or $(1, 1, 0)_{12}$

SARIMA

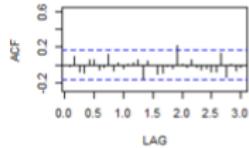
Model: (1,1,1) (0,1,1) Standardized Residuals



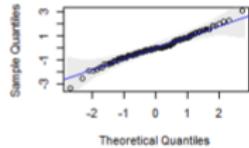
Model: (1,1,1) (1,1,0) Standardized Residuals



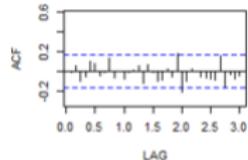
ACF of Residuals



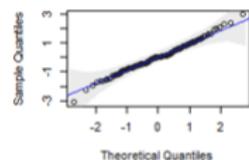
Normal Q-Q Plot of Std Residuals



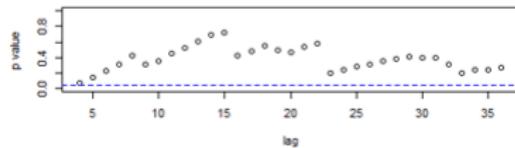
ACF of Residuals



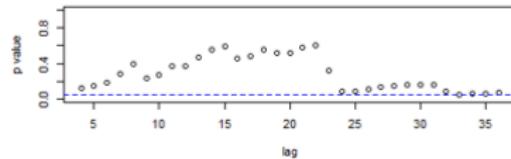
Normal Q-Q Plot of Std Residuals



p values for Ljung-Box statistic



p values for Ljung-Box statistic



SARIMA

- Remove AR term!

Is one model much better than the other one?

```
> m1$fit
Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
REPORT = 1, reltol = tol))

Coefficients:
            ar1      ma1      sar1
0.0547   -0.4886  -0.4731
s.e.  0.2161    0.1933   0.0800

sigma2 estimated as 0.001425:  log likelihood = 241.73,  aic = -475.47
> m2$fit
Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
REPORT = 1, reltol = tol))

Coefficients:
            ar1      ma1      sma1
0.1960   -0.5784  -0.5643
s.e.  0.2475    0.2132   0.0747

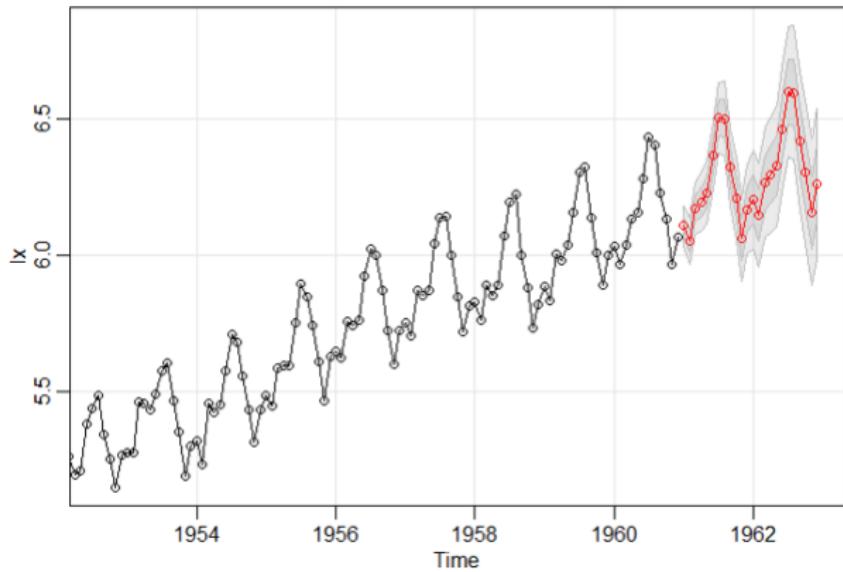
sigma^2 estimated as 0.001341:  log likelihood = 244.95,  aic = -481.9
```

$(1, 1, 1) \times (1, 1, 0)_{12}$

$(1, 1, 1) \times (0, 1, 1)_{12}$

SARIMA

- Forecasting



Read home

- Shumway and Stoffer, section 3.9
- R code: sarima, sarima.for, runs

Springer Texts in Statistics

Robert H. Shumway
David S. Stoffer

Time Series Analysis and Its Applications

With R Examples

Fourth Edition



Robert H. Shumway
David S. Stoffer

**Time Series Analysis and
Its Applications**
With R Examples

Fourth Edition



[live free or bark](#)

Preface to the Fourth Edition

The fourth edition follows the general layout of the third edition but includes some modernization of topics as well as the coverage of additional topics. The preface to the third edition—which follows—still applies, so we concentrate on the differences between the two editions here. As in the third edition, R code for each example is given in the text, even if the code is excruciatingly long. Most of the examples with seemingly endless coding are in the latter chapters. The R package for the text, `astsa`, is still supported and details may be found in [Appendix R](#). A number of data sets have been updated. For example, the global temperature deviation series have been updated to 2015 and are included in the newest version of the package; the corresponding examples and problems have been updated accordingly.

[Chapter 1](#) of this edition is similar to the previous edition, but we have included the definition of trend stationarity and the concept of prewhitening when using cross-correlation. The New York Stock Exchange data set, which focused on an old financial crisis, was replaced with a more current series of the Dow Jones Industrial Average, which focuses on a newer financial crisis. In [Chapter 2](#), we rewrote some of the regression review, changed the smoothing examples from the mortality data example to the Southern Oscillation Index and finding El Niño. We also expanded the discussion of lagged regression to [Chapter 3](#) to include the possibility of autocorrelated errors.

In [Chapter 3](#), we removed normality from definition of ARMA models; while the assumption is not necessary for the definition, it is essential for inference and prediction. We added a section on regression with ARMA errors and the corresponding problems; this section was previously in [Chapter 5](#). Some of the examples have been modified and we added some examples in the seasonal ARMA section.

In [Chapter 4](#), we improved and added some examples. The idea of modulated series is discussed using the classic star magnitude data set. We moved some of the filtering section forward for easier access to information when needed. We removed the reliance on `spec.pgram` (from the `stats` package) to `mvspec` (from the `astsa` package) so we can avoid having to spend pages explaining the quirks of `spec.pgram`, which tended to take over the narrative. The section on wavelets was removed because

there are so many accessible texts available. The spectral representation theorems are discussed in a little more detail using examples based on simple harmonic processes.

The general layout of [Chapter 5](#) and of [Chapter 7](#) is the same, although we have revised some of the examples. As previously mentioned, we moved regression with ARMA errors to [Chapter 3](#).

[Chapter 6](#) sees the biggest change in this edition. We have added a section on smoothing splines, and a section on hidden Markov models and switching autoregressions. The Bayesian section is completely rewritten and is on linear Gaussian state space models only. The nonlinear material in the previous edition is removed because it was old, and the newer material is in Douc, Moulines, and Stoffer (2014). Many of the examples have been rewritten to make the chapter more accessible. Our goal was to be able to have a course on state space models based primarily on the material in [Chapter 6](#).

The Appendices are similar, with some minor changes to [Appendix A](#) and [Appendix B](#). We added material to [Appendix C](#), including a discussion of Riemann–Stieltjes and stochastic integration, a proof of the fact that the spectra of autoregressive processes are dense in the space of spectral densities, and a proof of the fact that spectra are approximately the eigenvalues of the covariance matrix of a stationary process.

We tweaked, rewrote, improved, or revised some of the exercises, but the overall ordering and coverage is roughly the same. And, of course, we moved regression with ARMA errors problems to [Chapter 3](#) and removed the [Chapter 4](#) wavelet problems. The exercises for [Chapter 6](#) have been updated accordingly to reflect the new and improved version of the chapter.

Davis, CA
Pittsburgh, PA
September 2016

*Robert H. Shumway
David S. Stoffer*

Preface to the Third Edition

The goals of this book are to develop an appreciation for the richness and versatility of modern time series analysis as a tool for analyzing data, and still maintain a commitment to theoretical integrity, as exemplified by the seminal works of Brillinger (1975) and Hannan (1970) and the texts by Brockwell and Davis (1991) and Fuller (1995). The advent of inexpensive powerful computing has provided both real data and new software that can take one considerably beyond the fitting of simple time domain models, such as have been elegantly described in the landmark work of Box and Jenkins (1970). This book is designed to be useful as a text for courses in time series on several different levels and as a reference work for practitioners facing the analysis of time-correlated data in the physical, biological, and social sciences.

We have used earlier versions of the text at both the undergraduate and graduate levels over the past decade. Our experience is that an undergraduate course can be accessible to students with a background in regression analysis and may include Section 1.1–Section 1.5, Section 2.1–Section 2.3, the results and numerical parts of Section 3.1–Section 3.9, and briefly the results and numerical parts of Section 4.1–Section 4.4. At the advanced undergraduate or master’s level, where the students have some mathematical statistics background, more detailed coverage of the same sections, with the inclusion of extra topics from Chapter 5 or Chapter 6 can be used as a one-semester course. Often, the extra topics are chosen by the students according to their interests. Finally, a two-semester upper-level graduate course for mathematics, statistics, and engineering graduate students can be crafted by adding selected theoretical appendices. For the upper-level graduate course, we should mention that we are striving for a broader but less rigorous level of coverage than that which is attained by Brockwell and Davis (1991), the classic entry at this level.

The major difference between this third edition of the text and the second edition is that we provide R code for almost all of the numerical examples. An R package called `astsa` is provided for use with the text; see Section R.2 for details. R code is provided simply to enhance the exposition by making the numerical examples reproducible.

We have tried, where possible, to keep the problem sets in order so that an instructor may have an easy time moving from the second edition to the third edition.

However, some of the old problems have been revised and there are some new problems. Also, some of the data sets have been updated. We added one section in [Chapter 5](#) on unit roots and enhanced some of the presentations throughout the text. The exposition on state-space modeling, ARMAX models, and (multivariate) regression with autocorrelated errors in [Chapter 6](#) have been expanded. In this edition, we use standard R functions as much as possible, but we use our own scripts (included in [astsa](#)) when we feel it is necessary to avoid problems with a particular R function; these problems are discussed in detail on the website for the text under R Issues.

We thank John Kimmel, Executive Editor, Springer Statistics, for his guidance in the preparation and production of this edition of the text. We are grateful to Don Percival, University of Washington, for numerous suggestions that led to substantial improvement to the presentation in the second edition, and consequently in this edition. We thank Doug Wiens, University of Alberta, for help with some of the R code in [Chapter 4](#) and [Chapter 7](#), and for his many suggestions for improvement of the exposition. We are grateful for the continued help and advice of Pierre Duchesne, University of Montreal, and Alexander Aue, University of California, Davis. We also thank the many students and other readers who took the time to mention typographical errors and other corrections to the first and second editions. Finally, work on the this edition was supported by the National Science Foundation while one of us (D.S.S.) was working at the Foundation under the Intergovernmental Personnel Act.

Davis, CA
Pittsburgh, PA
September 2010

*Robert H. Shumway
David S. Stoffer*

Contents

Preface to the Fourth Edition	v
Preface to the Third Edition	vii
1 Characteristics of Time Series	1
1.1 The Nature of Time Series Data	2
1.2 Time Series Statistical Models	8
1.3 Measures of Dependence	14
1.4 Stationary Time Series	19
1.5 Estimation of Correlation	26
1.6 Vector-Valued and Multidimensional Series	33
Problems	38
2 Time Series Regression and Exploratory Data Analysis	47
2.1 Classical Regression in the Time Series Context	47
2.2 Exploratory Data Analysis	56
2.3 Smoothing in the Time Series Context	67
Problems	72
3 ARIMA Models	77
3.1 Autoregressive Moving Average Models	77
3.2 Difference Equations	90
3.3 Autocorrelation and Partial Autocorrelation	96
3.4 Forecasting	102
3.5 Estimation	115
3.6 Integrated Models for Nonstationary Data	133
3.7 Building ARIMA Models	137
3.8 Regression with Autocorrelated Errors	145
3.9 Multiplicative Seasonal ARIMA Models	148
Problems	156

4	Spectral Analysis and Filtering	167
4.1	Cyclical Behavior and Periodicity	168
4.2	The Spectral Density	174
4.3	Periodogram and Discrete Fourier Transform	181
4.4	Nonparametric Spectral Estimation	191
4.5	Parametric Spectral Estimation	205
4.6	Multiple Series and Cross-Spectra	208
4.7	Linear Filters	213
4.8	Lagged Regression Models	218
4.9	Signal Extraction and Optimum Filtering	223
4.10	Spectral Analysis of Multidimensional Series	227
	Problems	230
5	Additional Time Domain Topics	241
5.1	Long Memory ARMA and Fractional Differencing	241
5.2	Unit Root Testing	250
5.3	GARCH Models	253
5.4	Threshold Models	261
5.5	Lagged Regression and Transfer Function Modeling	265
5.6	Multivariate ARMAX Models	271
	Problems	284
6	State Space Models	287
6.1	Linear Gaussian Model	288
6.2	Filtering, Smoothing, and Forecasting	292
6.3	Maximum Likelihood Estimation	302
6.4	Missing Data Modifications	310
6.5	Structural Models: Signal Extraction and Forecasting	315
6.6	State-Space Models with Correlated Errors	319
6.6.1	ARMAX Models	320
6.6.2	Multivariate Regression with Autocorrelated Errors	322
6.7	Bootstrapping State Space Models	325
6.8	Smoothing Splines and the Kalman Smoother	331
6.9	Hidden Markov Models and Switching Autoregression	334
6.10	Dynamic Linear Models with Switching	345
6.11	Stochastic Volatility	357
6.12	Bayesian Analysis of State Space Models	365
	Problems	375
7	Statistical Methods in the Frequency Domain	383
7.1	Introduction	383
7.2	Spectral Matrices and Likelihood Functions	386
7.3	Regression for Jointly Stationary Series	388
7.4	Regression with Deterministic Inputs	397
7.5	Random Coefficient Regression	405

7.6 Analysis of Designed Experiments	407
7.7 Discriminant and Cluster Analysis	421
7.8 Principal Components and Factor Analysis	437
7.9 The Spectral Envelope	453
Problems	464
Appendix A Large Sample Theory	471
A.1 Convergence Modes	471
A.2 Central Limit Theorems	478
A.3 The Mean and Autocorrelation Functions	482
Appendix B Time Domain Theory	491
B.1 Hilbert Spaces and the Projection Theorem	491
B.2 Causal Conditions for ARMA Models	495
B.3 Large Sample Distribution of the AR Conditional Least Squares Estimators	497
B.4 The Wold Decomposition	500
Appendix C Spectral Domain Theory	503
C.1 Spectral Representation Theorems	503
C.2 Large Sample Distribution of the Smoothed Periodogram	507
C.3 The Complex Multivariate Normal Distribution	517
C.4 Integration	522
C.4.1 Riemann–Stieltjes Integration	522
C.4.2 Stochastic Integration	524
C.5 Spectral Analysis as Principal Component Analysis	525
C.6 Parametric Spectral Estimation	529
Appendix R R Supplement	531
R.1 First Things First	531
R.2 astsa	531
R.3 Getting Started	532
R.4 Time Series Primer	536
R.4.1 Graphics	539
References	541
Index	553

Chapter 1

Characteristics of Time Series

The analysis of experimental data that have been observed at different points in time leads to new and unique problems in statistical modeling and inference. The obvious correlation introduced by the sampling of adjacent points in time can severely restrict the applicability of the many conventional statistical methods traditionally dependent on the assumption that these adjacent observations are independent and identically distributed. The systematic approach by which one goes about answering the mathematical and statistical questions posed by these time correlations is commonly referred to as time series analysis.

The impact of time series analysis on scientific applications can be partially documented by producing an abbreviated listing of the diverse fields in which important time series problems may arise. For example, many familiar time series occur in the field of economics, where we are continually exposed to daily stock market quotations or monthly unemployment figures. Social scientists follow population series, such as birthrates or school enrollments. An epidemiologist might be interested in the number of influenza cases observed over some time period. In medicine, blood pressure measurements traced over time could be useful for evaluating drugs used in treating hypertension. Functional magnetic resonance imaging of brain-wave time series patterns might be used to study how the brain reacts to certain stimuli under various experimental conditions.

In our view, the first step in any time series investigation always involves careful examination of the recorded data plotted over time. This scrutiny often suggests the method of analysis as well as statistics that will be of use in summarizing the information in the data. Before looking more closely at the particular statistical methods, it is appropriate to mention that two separate, but not necessarily mutually exclusive, approaches to time series analysis exist, commonly identified as the *time domain approach* and the *frequency domain approach*. The time domain approach views the investigation of lagged relationships as most important (e.g., how does what happened today affect what will happen tomorrow), whereas the frequency domain approach views the investigation of cycles as most important (e.g., what is the economic cycle through periods of expansion and recession). We will explore both types of approaches in the following sections.

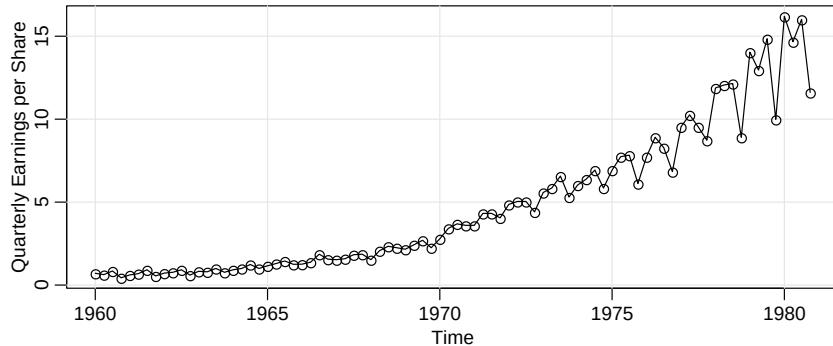


Fig. 1.1. Johnson & Johnson quarterly earnings per share, 84 quarters, 1960-I to 1980-IV.

1.1 The Nature of Time Series Data

Some of the problems and questions of interest to the prospective time series analyst can best be exposed by considering real experimental data taken from different subject areas. The following cases illustrate some of the common kinds of experimental time series data as well as some of the statistical questions that might be asked about such data.

Example 1.1 Johnson & Johnson Quarterly Earnings

Figure 1.1 shows quarterly earnings per share for the U.S. company Johnson & Johnson, furnished by Professor Paul Griffin (personal communication) of the Graduate School of Management, University of California, Davis. There are 84 quarters (21 years) measured from the first quarter of 1960 to the last quarter of 1980. Modeling such series begins by observing the primary patterns in the time history. In this case, note the gradually increasing underlying trend and the rather regular variation superimposed on the trend that seems to repeat over quarters. Methods for analyzing data such as these are explored in [Chapter 2](#) and [Chapter 6](#). To plot the data using the R statistical package, type the following:^{1.1}

```
library(astsa) # SEE THE FOOTNOTE
plot(jj, type="o", ylab="Quarterly Earnings per Share")
```

Example 1.2 Global Warming

Consider the global temperature series record shown in [Figure 1.2](#). The data are the global mean land–ocean temperature index from 1880 to 2015, with the base period 1951–1980. In particular, the data are deviations, measured in degrees centigrade, from the 1951–1980 average, and are an update of Hansen et al. (2006). We note an apparent upward trend in the series during the latter part of the twentieth century that has been used as an argument for the global warming hypothesis. Note also the leveling off at about 1935 and then another rather sharp upward trend at about

^{1.1} Throughout the text, we assume that the R package for the book, `astsa`, has been installed and loaded. See [Section R.2](#) for further details.

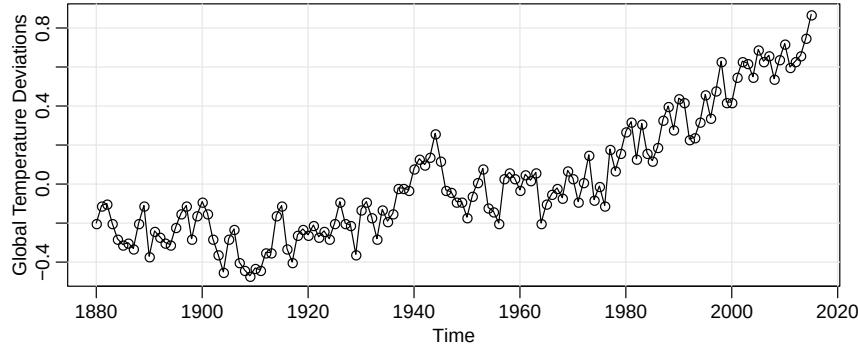


Fig. 1.2. Yearly average global temperature deviations (1880–2015) in degrees centigrade.

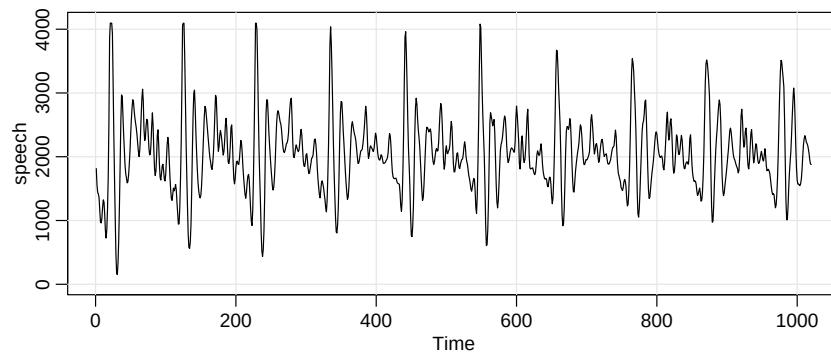


Fig. 1.3. Speech recording of the syllable *aaa*⋯*hhh* sampled at 10,000 points per second with $n = 1020$ points.

1970. The question of interest for global warming proponents and opponents is whether the overall trend is natural or whether it is caused by some human-induced interface. [Problem 2.8](#) examines 634 years of glacial sediment data that might be taken as a long-term temperature proxy. Such percentage changes in temperature do not seem to be unusual over a time period of 100 years. Again, the question of trend is of more interest than particular periodicities. The R code for this example is similar to the code in [Example 1.1](#):

```
plot(globtemp, type="o", ylab="Global Temperature Deviations")
```

Example 1.3 Speech Data

[Figure 1.3](#) shows a small .1 second (1000 point) sample of recorded speech for the phrase *aaa*⋯*hhh*, and we note the repetitive nature of the signal and the rather regular periodicities. One current problem of great interest is computer recognition of speech, which would require converting this particular signal into the recorded phrase *aaa*⋯*hhh*. Spectral analysis can be used in this context to produce a signature of this phrase that can be compared with signatures of various

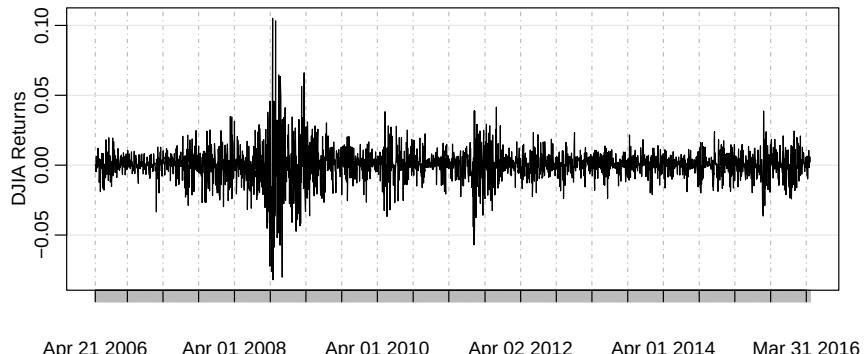


Fig. 1.4. The daily returns of the Dow Jones Industrial Average (DJIA) from April 20, 2006 to April 20, 2016.

library syllables to look for a match. One can immediately notice the rather regular repetition of small wavelets. The separation between the packets is known as the *pitch period* and represents the response of the vocal tract filter to a periodic sequence of pulses stimulated by the opening and closing of the glottis. In R, you can reproduce Figure 1.3 using `plot(speech)`.

Example 1.4 Dow Jones Industrial Average

As an example of financial time series data, Figure 1.4 shows the daily *returns* (or percent change) of the Dow Jones Industrial Average (DJIA) from April 20, 2006 to April 20, 2016. It is easy to spot the financial crisis of 2008 in the figure. The data shown in Figure 1.4 are typical of return data. The mean of the series appears to be stable with an average return of approximately zero, however, highly volatile (variable) periods tend to be clustered together. A problem in the analysis of these type of financial data is to forecast the volatility of future returns. Models such as *ARCH* and *GARCH* models (Engle, 1982; Bollerslev, 1986) and *stochastic volatility* models (Harvey, Ruiz and Shephard, 1994) have been developed to handle these problems. We will discuss these models and the analysis of financial data in Chapter 5 and Chapter 6. The data were obtained using the Technical Trading Rules (TTR) package to download the data from YahooTM and then plot it. We then used the fact that if x_t is the actual value of the DJIA and $r_t = (x_t - x_{t-1})/x_{t-1}$ is the return, then $1 + r_t = x_t/x_{t-1}$ and $\log(1 + r_t) = \log(x_t/x_{t-1}) = \log(x_t) - \log(x_{t-1}) \approx r_t$.^{1.2} The data set is also available in `astsa`, but `xts` must be loaded.

```
# library(TTR)
# dji = getYahooData("^DJI", start=20060420, end=20160420, freq="daily")
library(xts)
djiar = diff(log(dji$Close))[-1] # approximate returns
plot(djiar, main="DJIA Returns", type="n")
lines(djiar)
```

^{1.2} $\log(1 + p) = p - \frac{p^2}{2} + \frac{p^3}{3} - \dots$ for $-1 < p \leq 1$. If p is near zero, the higher-order terms in the expansion are negligible.

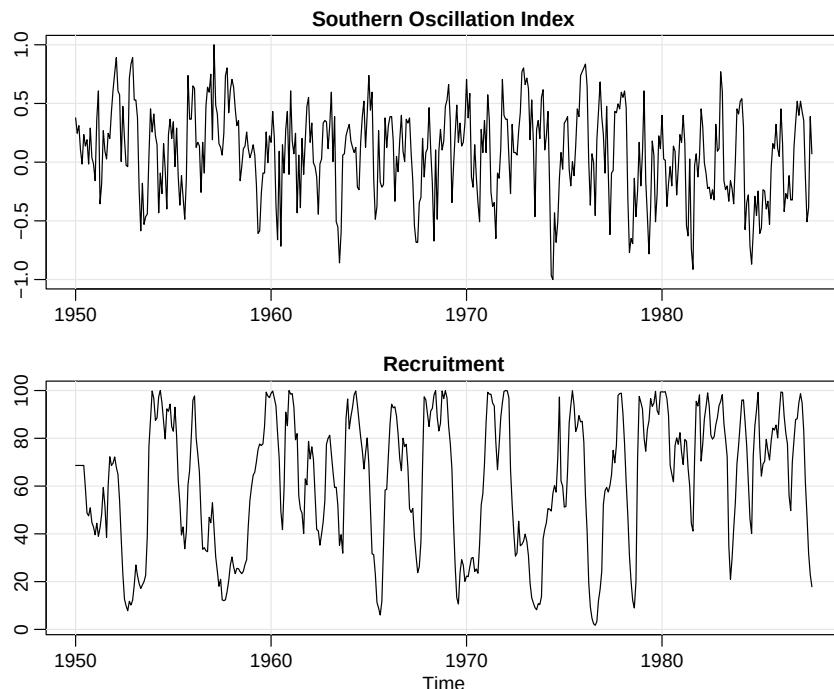


Fig. 1.5. Monthly SOI and Recruitment (estimated new fish), 1950-1987.

Example 1.5 El Niño and Fish Population

We may also be interested in analyzing several time series at once. Figure 1.5 shows monthly values of an environmental series called the *Southern Oscillation Index* (SOI) and associated Recruitment (number of new fish) furnished by Dr. Roy Mendelsohn of the Pacific Environmental Fisheries Group (personal communication). Both series are for a period of 453 months ranging over the years 1950–1987. The SOI measures changes in air pressure, related to sea surface temperatures in the central Pacific Ocean. The central Pacific warms every three to seven years due to the El Niño effect, which has been blamed for various global extreme weather events. Both series in Figure 1.5 exhibit repetitive behavior, with regularly repeating cycles that are easily visible. This periodic behavior is of interest because underlying processes of interest may be regular and the rate or *frequency* of oscillation characterizing the behavior of the underlying series would help to identify them. The series show two basic oscillations types, an obvious annual cycle (hot in the summer, cold in the winter), and a slower frequency that seems to repeat about every 4 years. The study of the kinds of cycles and their strengths is the subject of Chapter 4. The two series are also related; it is easy to imagine the fish population is dependent on the ocean temperature. This possibility suggests trying some version of regression analysis as a procedure for relating the two series. *Transfer function*

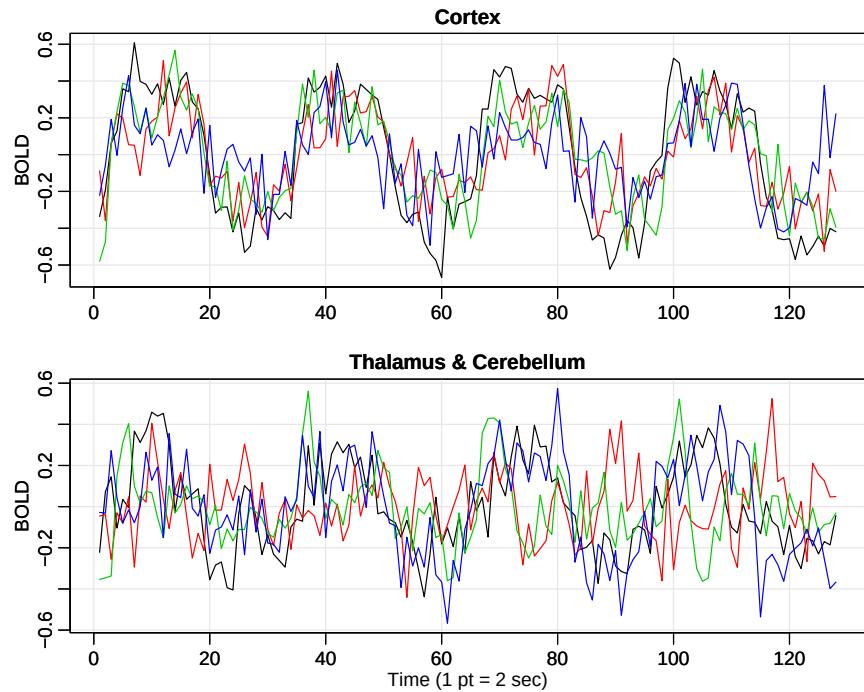


Fig. 1.6. fMRI data from various locations in the cortex, thalamus, and cerebellum; $n = 128$ points, one observation taken every 2 seconds.

modeling, as considered in [Chapter 5](#), can also be applied in this case. The following R code will reproduce [Figure 1.5](#):

```
par(mfrow = c(2,1)) # set up the graphics
plot(soi, ylab="", xlab="", main="Southern Oscillation Index")
plot(rec, ylab="", xlab="", main="Recruitment")
```

Example 1.6 fMRI Imaging

A fundamental problem in classical statistics occurs when we are given a collection of independent series or vectors of series, generated under varying experimental conditions or treatment configurations. Such a set of series is shown in [Figure 1.6](#), where we observe data collected from various locations in the brain via functional magnetic resonance imaging (fMRI). In this example, five subjects were given periodic brushing on the hand. The stimulus was applied for 32 seconds and then stopped for 32 seconds; thus, the signal period is 64 seconds. The sampling rate was one observation every 2 seconds for 256 seconds ($n = 128$). For this example, we averaged the results over subjects (these were evoked responses, and all subjects were in phase). The series shown in [Figure 1.6](#) are consecutive measures of blood oxygenation-level dependent (BOLD) signal intensity, which measures areas of activation in the brain. Notice that the periodicities appear strongly in the motor cortex

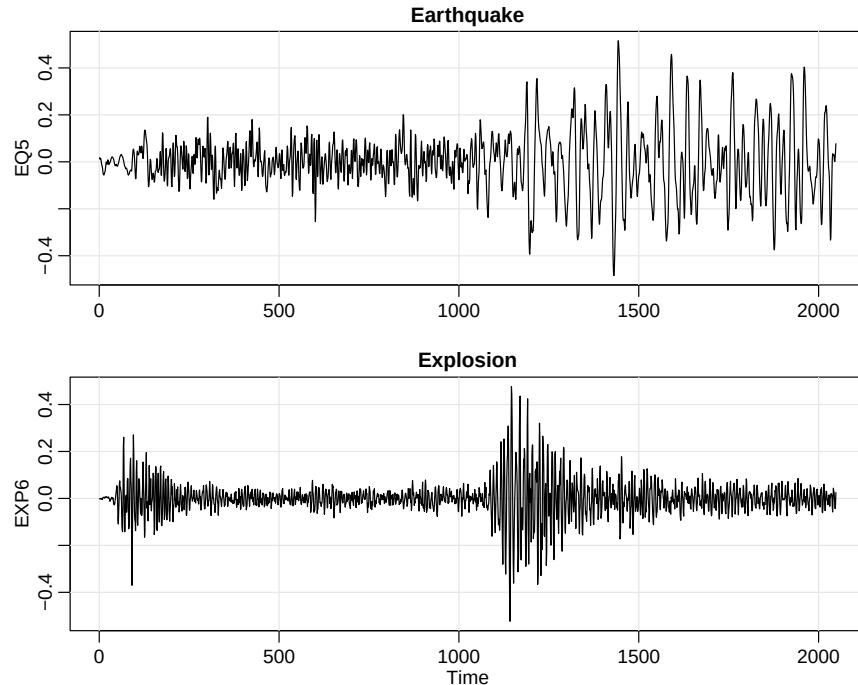


Fig. 1.7. Arrival phases from an earthquake (top) and explosion (bottom) at 40 points per second.

series and less strongly in the thalamus and cerebellum. The fact that one has series from different areas of the brain suggests testing whether the areas are responding differently to the brush stimulus. Analysis of variance techniques accomplish this in classical statistics, and we show in Chapter 7 how these classical techniques extend to the time series case, leading to a spectral analysis of variance. The following R commands can be used to plot the data:

```
par(mfrow=c(2,1))
ts.plot(fmri1[,2:5], col=1:4, ylab="BOLD", main="Cortex")
ts.plot(fmri1[,6:9], col=1:4, ylab="BOLD", main="Thalamus & Cerebellum")
```

Example 1.7 Earthquakes and Explosions

As a final example, the series in Figure 1.7 represent two phases or arrivals along the surface, denoted by P ($t = 1, \dots, 1024$) and S ($t = 1025, \dots, 2048$), at a seismic recording station. The recording instruments in Scandinavia are observing earthquakes and mining explosions with one of each shown in Figure 1.7. The general problem of interest is in distinguishing or discriminating between waveforms generated by earthquakes and those generated by explosions. Features that may be important are the rough amplitude ratios of the first phase P to the second phase S, which tend to be smaller for earthquakes than for explosions. In the case of the

two events in [Figure 1.7](#), the ratio of maximum amplitudes appears to be somewhat less than .5 for the earthquake and about 1 for the explosion. Otherwise, note a subtle difference exists in the periodic nature of the S phase for the earthquake. We can again think about spectral analysis of variance for testing the equality of the periodic components of earthquakes and explosions. We would also like to be able to classify future P and S components from events of unknown origin, leading to the *time series discriminant analysis* developed in [Chapter 7](#).

To plot the data as in this example, use the following commands in R:

```
par(mfrow=c(2,1))
plot(EQS, main='Earthquake')
plot(EXP6, main="Explosion")
```

1.2 Time Series Statistical Models

The primary objective of time series analysis is to develop mathematical models that provide plausible descriptions for sample data, like that encountered in the previous section. In order to provide a statistical setting for describing the character of data that seemingly fluctuate in a random fashion over time, we assume a *time series* can be defined as a collection of random variables indexed according to the order they are obtained in time. For example, we may consider a time series as a sequence of random variables, x_1, x_2, x_3, \dots , where the random variable x_1 denotes the value taken by the series at the first time point, the variable x_2 denotes the value for the second time period, x_3 denotes the value for the third time period, and so on. In general, a collection of random variables, $\{x_t\}$, indexed by t is referred to as a *stochastic process*. In this text, t will typically be discrete and vary over the integers $t = 0, \pm 1, \pm 2, \dots$, or some subset of the integers. The observed values of a stochastic process are referred to as a *realization* of the stochastic process. Because it will be clear from the context of our discussions, we use the term *time series* whether we are referring generically to the process or to a particular realization and make no notational distinction between the two concepts.

It is conventional to display a sample time series graphically by plotting the values of the random variables on the vertical axis, or ordinate, with the time scale as the abscissa. It is usually convenient to connect the values at adjacent time periods to reconstruct visually some original hypothetical continuous time series that might have produced these values as a discrete sample. Many of the series discussed in the previous section, for example, could have been observed at any continuous point in time and are conceptually more properly treated as *continuous time series*. The approximation of these series by *discrete time parameter series* sampled at equally spaced points in time is simply an acknowledgment that sampled data will, for the most part, be discrete because of restrictions inherent in the method of collection. Furthermore, the analysis techniques are then feasible using computers, which are limited to digital computations. Theoretical developments also rest on the idea that a continuous parameter time series should be specified in terms of finite-dimensional *distribution functions* defined over a finite number of points in time. This is not to

say that the selection of the sampling interval or rate is not an extremely important consideration. The appearance of data can be changed completely by adopting an insufficient sampling rate. We have all seen wheels in movies appear to be turning backwards because of the insufficient number of frames sampled by the camera. This phenomenon leads to a distortion called *aliasing* (see [Section 4.1](#)).

The fundamental visual characteristic distinguishing the different series shown in [Example 1.1](#)–[Example 1.7](#) is their differing degrees of smoothness. One possible explanation for this smoothness is that it is being induced by the supposition that adjacent points in time are *correlated*, so the value of the series at time t , say, x_t , depends in some way on the past values x_{t-1}, x_{t-2}, \dots . This model expresses a fundamental way in which we might think about generating realistic-looking time series. To begin to develop an approach to using collections of random variables to model time series, consider [Example 1.8](#).

Example 1.8 White Noise (3 flavors)

A simple kind of generated series might be a collection of uncorrelated random variables, w_t , with mean 0 and finite variance σ_w^2 . The time series generated from uncorrelated variables is used as a model for noise in engineering applications, where it is called *white noise*; we shall denote this process as $w_t \sim \text{wn}(0, \sigma_w^2)$. The designation white originates from the analogy with white light and indicates that all possible periodic oscillations are present with equal strength.

We will sometimes require the noise to be independent and identically distributed (iid) random variables with mean 0 and variance σ_w^2 . We distinguish this by writing $w_t \sim \text{iid}(0, \sigma_w^2)$ or by saying *white independent noise* or *iid noise*. A particularly useful white noise series is *Gaussian white noise*, wherein the w_t are independent normal random variables, with mean 0 and variance σ_w^2 ; or more succinctly, $w_t \sim \text{iid } N(0, \sigma_w^2)$. [Figure 1.8](#) shows in the upper panel a collection of 500 such random variables, with $\sigma_w^2 = 1$, plotted in the order in which they were drawn. The resulting series bears a slight resemblance to the explosion in [Figure 1.7](#) but is not smooth enough to serve as a plausible model for any of the other experimental series. The plot tends to show visually a mixture of many different kinds of oscillations in the white noise series.

If the stochastic behavior of all time series could be explained in terms of the white noise model, classical statistical methods would suffice. Two ways of introducing serial correlation and more smoothness into time series models are given in [Example 1.9](#) and [Example 1.10](#).

Example 1.9 Moving Averages and Filtering

We might replace the white noise series w_t by a *moving average* that smooths the series. For example, consider replacing w_t in [Example 1.8](#) by an average of its current value and its immediate neighbors in the past and future. That is, let

$$v_t = \frac{1}{3}(w_{t-1} + w_t + w_{t+1}), \quad (1.1)$$

which leads to the series shown in the lower panel of [Figure 1.8](#). Inspecting the series shows a smoother version of the first series, reflecting the fact that the slower

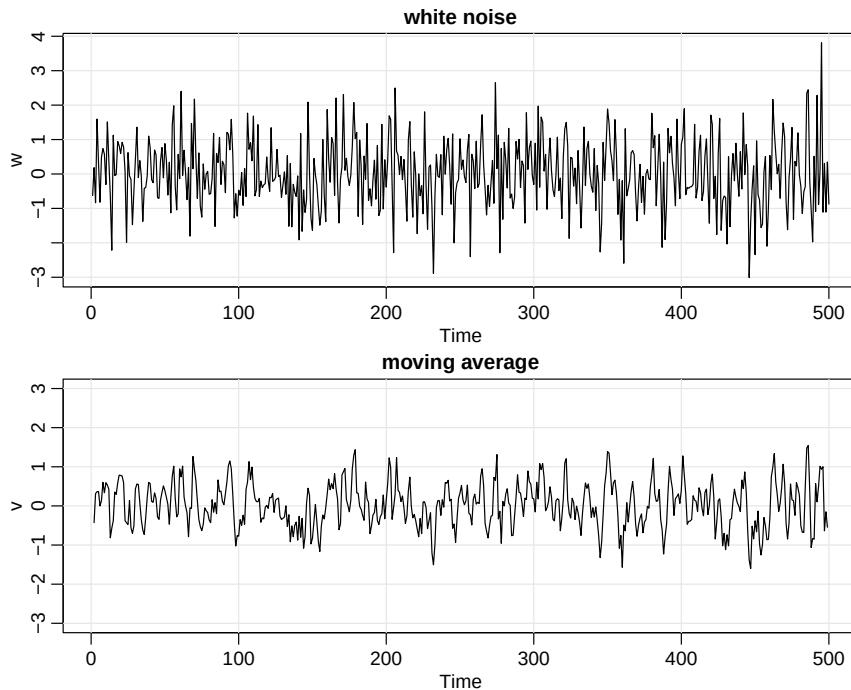


Fig. 1.8. Gaussian white noise series (top) and three-point moving average of the Gaussian white noise series (bottom).

oscillations are more apparent and some of the faster oscillations are taken out. We begin to notice a similarity to the SOI in Figure 1.5, or perhaps, to some of the fMRI series in Figure 1.6.

A linear combination of values in a time series such as in (1.1) is referred to, generically, as a filtered series; hence the command `filter` in the following code for Figure 1.8.

```
w = rnorm(500,0,1)          # 500 N(0,1) variates
v = filter(w, sides=2, filter=rep(1/3,3)) # moving average
par(mfrow=c(2,1))
plot.ts(w, main="white noise")
plot.ts(v, ylim=c(-3,3), main="moving average")
```

The speech series in Figure 1.3 and the Recruitment series in Figure 1.5, as well as some of the MRI series in Figure 1.6, differ from the moving average series because one particular kind of oscillatory behavior seems to predominate, producing a sinusoidal type of behavior. A number of methods exist for generating series with this quasi-periodic behavior; we illustrate a popular one based on the autoregressive model considered in Chapter 3.

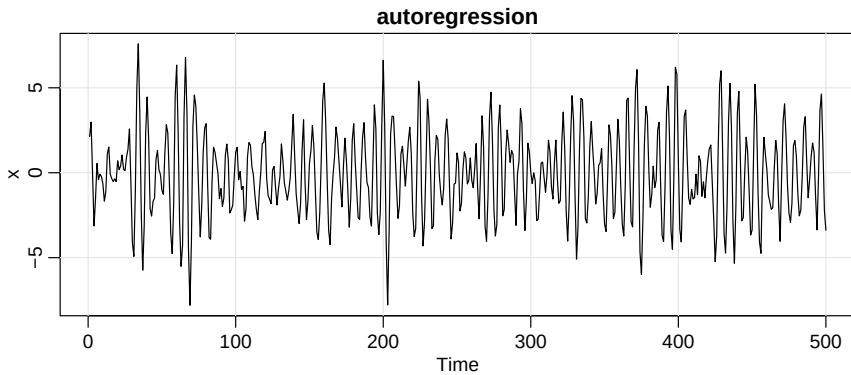


Fig. 1.9. Autoregressive series generated from model (1.2).

Example 1.10 Autoregressions

Suppose we consider the white noise series w_t of Example 1.8 as input and calculate the output using the second-order equation

$$x_t = x_{t-1} - .9x_{t-2} + w_t \quad (1.2)$$

successively for $t = 1, 2, \dots, 500$. Equation (1.2) represents a regression or prediction of the current value x_t of a time series as a function of the past two values of the series, and, hence, the term *autoregression* is suggested for this model. A problem with startup values exists here because (1.2) also depends on the initial conditions x_0 and x_{-1} , but assuming we have the values, we generate the succeeding values by substituting into (1.2). The resulting output series is shown in Figure 1.9, and we note the periodic behavior of the series, which is similar to that displayed by the speech series in Figure 1.3. The autoregressive model above and its generalizations can be used as an underlying model for many observed series and will be studied in detail in Chapter 3.

As in the previous example, the data are obtained by a filter of white noise. The function `filter` uses zeros for the initial values. In this case, $x_1 = w_1$, and $x_2 = x_1 + w_2 = w_1 + w_2$, and so on, so that the values do not satisfy (1.2). An easy fix is to run the filter for longer than needed and remove the initial values.

```
w = rnorm(550, 0, 1) # 50 extra to avoid startup problems
x = filter(w, filter=c(1, -.9), method="recursive")[-(1:50)] # remove first 50
plot.ts(x, main="autoregression")
```

Example 1.11 Random Walk with Drift

A model for analyzing trend such as seen in the global temperature data in Figure 1.2, is the *random walk with drift* model given by

$$x_t = \delta + x_{t-1} + w_t \quad (1.3)$$

for $t = 1, 2, \dots$, with initial condition $x_0 = 0$, and where w_t is white noise. The constant δ is called the *drift*, and when $\delta = 0$, (1.3) is called simply a *random walk*.

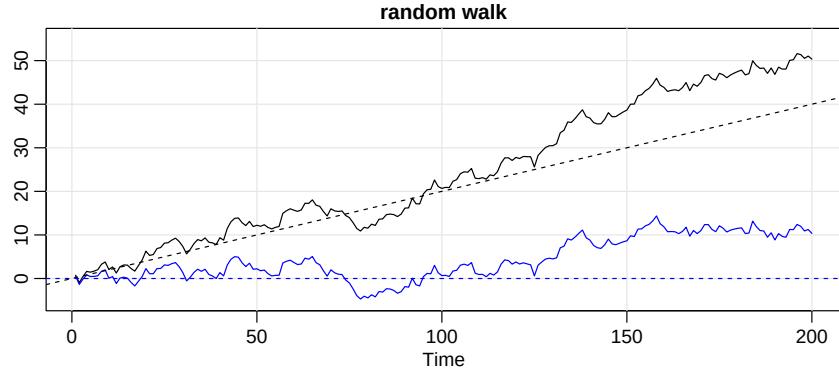


Fig. 1.10. Random walk, $\sigma_w = 1$, with drift $\delta = .2$ (upper jagged line), without drift, $\delta = 0$ (lower jagged line), and straight (dashed) lines with slope δ .

The term random walk comes from the fact that, when $\delta = 0$, the value of the time series at time t is the value of the series at time $t - 1$ plus a completely random movement determined by w_t . Note that we may rewrite (1.3) as a cumulative sum of white noise variates. That is,

$$x_t = \delta t + \sum_{j=1}^t w_j \quad (1.4)$$

for $t = 1, 2, \dots$; either use induction, or plug (1.4) into (1.3) to verify this statement. Figure 1.10 shows 200 observations generated from the model with $\delta = 0$ and $.2$, and with $\sigma_w = 1$. For comparison, we also superimposed the straight line $.2t$ on the graph. To reproduce Figure 1.10 in R use the following code (notice the use of multiple commands per line using a semicolon).

```
set.seed(154)      # so you can reproduce the results
w = rnorm(200);  x = cumsum(w)  # two commands in one line
wd = w +.2;      xd = cumsum(wd)
plot.ts(xd, ylim=c(-5,55), main="random walk", ylab='')
lines(x, col=4); abline(h=0, col=4, lty=2); abline(a=0, b=.2, lty=2)
```

Example 1.12 Signal in Noise

Many realistic models for generating time series assume an underlying signal with some consistent periodic variation, contaminated by adding a random noise. For example, it is easy to detect the regular cycle fMRI series displayed on the top of Figure 1.6. Consider the model

$$x_t = 2 \cos(2\pi \frac{t+15}{50}) + w_t \quad (1.5)$$

for $t = 1, 2, \dots, 500$, where the first term is regarded as the signal, shown in the upper panel of Figure 1.11. We note that a sinusoidal waveform can be written as

$$A \cos(2\pi\omega t + \phi), \quad (1.6)$$

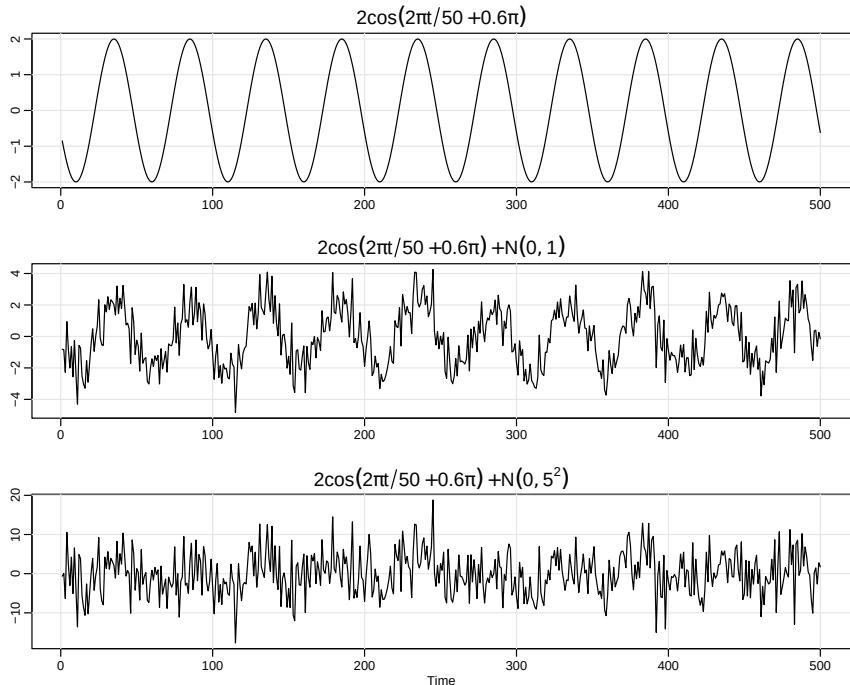


Fig. 1.11. Cosine wave with period 50 points (top panel) compared with the cosine wave contaminated with additive white Gaussian noise, $\sigma_w = 1$ (middle panel) and $\sigma_w = 5$ (bottom panel); see (1.5).

where A is the amplitude, ω is the frequency of oscillation, and ϕ is a phase shift. In (1.5), $A = 2$, $\omega = 1/50$ (one cycle every 50 time points), and $\phi = 2\pi 15/50 = .6\pi$.

An additive noise term was taken to be white noise with $\sigma_w = 1$ (middle panel) and $\sigma_w = 5$ (bottom panel), drawn from a normal distribution. Adding the two together obscures the signal, as shown in the lower panels of Figure 1.11. Of course, the degree to which the signal is obscured depends on the amplitude of the signal and the size of σ_w . The ratio of the amplitude of the signal to σ_w (or some function of the ratio) is sometimes called the *signal-to-noise ratio (SNR)*; the larger the SNR, the easier it is to detect the signal. Note that the signal is easily discernible in the middle panel of Figure 1.11, whereas the signal is obscured in the bottom panel. Typically, we will not observe the signal but the signal obscured by noise.

To reproduce Figure 1.11 in R, use the following commands:

```
cs = 2*cos(2*pi*1:500/50 + .6*pi); w = rnorm(500,0,1)
par(mfrow=c(3,1), mar=c(3,2,2,1), cex.main=1.5)
plot.ts(cs, main=expression(2*cos(2*pi*t/50+.6*pi)))
plot.ts(cs+w, main=expression(2*cos(2*pi*t/50+.6*pi) + N(0,1)))
plot.ts(cs+5*w, main=expression(2*cos(2*pi*t/50+.6*pi) + N(0,25)))
```

In [Chapter 4](#), we will study the use of *spectral analysis* as a possible technique for detecting regular or periodic signals, such as the one described in [Example 1.12](#). In general, we would emphasize the importance of simple additive models such as given above in the form

$$x_t = s_t + v_t, \quad (1.7)$$

where s_t denotes some unknown signal and v_t denotes a time series that may be white or correlated over time. The problems of detecting a signal and then in estimating or extracting the waveform of s_t are of great interest in many areas of engineering and the physical and biological sciences. In economics, the underlying signal may be a trend or it may be a seasonal component of a series. Models such as (1.7), where the signal has an autoregressive structure, form the motivation for the state-space model of [Chapter 6](#).

In the above examples, we have tried to motivate the use of various combinations of random variables emulating real time series data. Smoothness characteristics of observed time series were introduced by combining the random variables in various ways. Averaging independent random variables over adjacent time points, as in [Example 1.9](#), or looking at the output of difference equations that respond to white noise inputs, as in [Example 1.10](#), are common ways of generating correlated data. In the next section, we introduce various theoretical measures used for describing how time series behave. As is usual in statistics, the complete description involves the multivariate distribution function of the jointly sampled values x_1, x_2, \dots, x_n , whereas more economical descriptions can be had in terms of the mean and autocorrelation functions. Because correlation is an essential feature of time series analysis, the most useful descriptive measures are those expressed in terms of covariance and correlation functions.

1.3 Measures of Dependence

A complete description of a time series, observed as a collection of n random variables at arbitrary time points t_1, t_2, \dots, t_n , for any positive integer n , is provided by the joint distribution function, evaluated as the probability that the values of the series are jointly less than the n constants, c_1, c_2, \dots, c_n ; i.e.,

$$F_{t_1, t_2, \dots, t_n}(c_1, c_2, \dots, c_n) = \Pr(x_{t_1} \leq c_1, x_{t_2} \leq c_2, \dots, x_{t_n} \leq c_n). \quad (1.8)$$

Unfortunately, these multidimensional distribution functions cannot usually be written easily unless the random variables are jointly normal, in which case the joint density has the well-known form displayed in (1.33).

Although the joint distribution function describes the data completely, it is an unwieldy tool for displaying and analyzing time series data. The distribution function (1.8) must be evaluated as a function of n arguments, so any plotting of the corresponding multivariate density functions is virtually impossible. The marginal distribution functions

$$F_t(x) = P\{x_t \leq x\}$$

or the corresponding marginal density functions

$$f_t(x) = \frac{\partial F_t(x)}{\partial x},$$

when they exist, are often informative for examining the marginal behavior of a series.^{1.3} Another informative marginal descriptive measure is the mean function.

Definition 1.1 *The mean function is defined as*

$$\mu_{xt} = E(x_t) = \int_{-\infty}^{\infty} x f_t(x) dx, \quad (1.9)$$

provided it exists, where E denotes the usual expected value operator. When no confusion exists about which time series we are referring to, we will drop a subscript and write μ_{xt} as μ_t .

Example 1.13 Mean Function of a Moving Average Series

If w_t denotes a white noise series, then $\mu_{wt} = E(w_t) = 0$ for all t . The top series in [Figure 1.8](#) reflects this, as the series clearly fluctuates around a mean value of zero. Smoothing the series as in [Example 1.9](#) does not change the mean because we can write

$$\mu_{vt} = E(v_t) = \frac{1}{3}[E(w_{t-1}) + E(w_t) + E(w_{t+1})] = 0.$$

Example 1.14 Mean Function of a Random Walk with Drift

Consider the random walk with drift model given in [\(1.4\)](#),

$$x_t = \delta t + \sum_{j=1}^t w_j, \quad t = 1, 2, \dots .$$

Because $E(w_t) = 0$ for all t , and δ is a constant, we have

$$\mu_{xt} = E(x_t) = \delta t + \sum_{j=1}^t E(w_j) = \delta t$$

which is a straight line with slope δ . A realization of a random walk with drift can be compared to its mean function in [Figure 1.10](#).

^{1.3} If x_t is Gaussian with mean μ_t and variance σ_t^2 , abbreviated as $x_t \sim N(\mu_t, \sigma_t^2)$, the marginal density is given by $f_t(x) = \frac{1}{\sigma_t \sqrt{2\pi}} \exp \left\{ -\frac{1}{2\sigma_t^2} (x - \mu_t)^2 \right\}, x \in \mathbb{R}$.

Example 1.15 Mean Function of Signal Plus Noise

A great many practical applications depend on assuming the observed data have been generated by a fixed signal waveform superimposed on a zero-mean noise process, leading to an additive signal model of the form (1.5). It is clear, because the signal in (1.5) is a fixed function of time, we will have

$$\begin{aligned}\mu_{xt} &= E(x_t) = E\left[2 \cos(2\pi \frac{t+15}{50}) + w_t\right] \\ &= 2 \cos(2\pi \frac{t+15}{50}) + E(w_t) \\ &= 2 \cos(2\pi \frac{t+15}{50}),\end{aligned}$$

and the mean function is just the cosine wave.

The lack of independence between two adjacent values x_s and x_t can be assessed numerically, as in classical statistics, using the notions of covariance and correlation. Assuming the variance of x_t is finite, we have the following definition.

Definition 1.2 *The autocovariance function is defined as the second moment product*

$$\gamma_x(s, t) = \text{cov}(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)], \quad (1.10)$$

for all s and t . When no possible confusion exists about which time series we are referring to, we will drop the subscript and write $\gamma_x(s, t)$ as $\gamma(s, t)$. Note that $\gamma_x(s, t) = \gamma_x(t, s)$ for all time points s and t .

The autocovariance measures the *linear* dependence between two points on the same series observed at different times. Very smooth series exhibit autocovariance functions that stay large even when the t and s are far apart, whereas choppy series tend to have autocovariance functions that are nearly zero for large separations. Recall from classical statistics that if $\gamma_x(s, t) = 0$, x_s and x_t are not linearly related, but there still may be some dependence structure between them. If, however, x_s and x_t are bivariate normal, $\gamma_x(s, t) = 0$ ensures their independence. It is clear that, for $s = t$, the autocovariance reduces to the (assumed finite) *variance*, because

$$\gamma_x(t, t) = E[(x_t - \mu_t)^2] = \text{var}(x_t). \quad (1.11)$$

Example 1.16 Autocovariance of White Noise

The white noise series w_t has $E(w_t) = 0$ and

$$\gamma_w(s, t) = \text{cov}(w_s, w_t) = \begin{cases} \sigma_w^2 & s = t, \\ 0 & s \neq t. \end{cases} \quad (1.12)$$

A realization of white noise with $\sigma_w^2 = 1$ is shown in the top panel of Figure 1.8.

We often have to calculate the autocovariance between filtered series. A useful result is given in the following proposition.

Property 1.1 Covariance of Linear Combinations

If the random variables

$$U = \sum_{j=1}^m a_j X_j \quad \text{and} \quad V = \sum_{k=1}^r b_k Y_k$$

are linear combinations of (finite variance) random variables $\{X_j\}$ and $\{Y_k\}$, respectively, then

$$\text{cov}(U, V) = \sum_{j=1}^m \sum_{k=1}^r a_j b_k \text{cov}(X_j, Y_k). \quad (1.13)$$

Furthermore, $\text{var}(U) = \text{cov}(U, U)$.

Example 1.17 Autocovariance of a Moving Average

Consider applying a three-point moving average to the white noise series w_t of the previous example as in [Example 1.9](#). In this case,

$$\gamma_v(s, t) = \text{cov}(v_s, v_t) = \text{cov}\left\{\frac{1}{3}(w_{s-1} + w_s + w_{s+1}), \frac{1}{3}(w_{t-1} + w_t + w_{t+1})\right\}.$$

When $s = t$ we have

$$\begin{aligned} \gamma_v(t, t) &= \frac{1}{9} \text{cov}\{(w_{t-1} + w_t + w_{t+1}), (w_{t-1} + w_t + w_{t+1})\} \\ &= \frac{1}{9} [\text{cov}(w_{t-1}, w_{t-1}) + \text{cov}(w_t, w_t) + \text{cov}(w_{t+1}, w_{t+1})] \\ &= \frac{3}{9} \sigma_w^2. \end{aligned}$$

When $s = t + 1$,

$$\begin{aligned} \gamma_v(t+1, t) &= \frac{1}{9} \text{cov}\{(w_t + w_{t+1} + w_{t+2}), (w_{t-1} + w_t + w_{t+1})\} \\ &= \frac{1}{9} [\text{cov}(w_t, w_t) + \text{cov}(w_{t+1}, w_{t+1})] \\ &= \frac{2}{9} \sigma_w^2, \end{aligned}$$

using (1.12). Similar computations give $\gamma_v(t-1, t) = 2\sigma_w^2/9$, $\gamma_v(t+2, t) = \gamma_v(t-2, t) = \sigma_w^2/9$, and 0 when $|t - s| > 2$. We summarize the values for all s and t as

$$\gamma_v(s, t) = \begin{cases} \frac{3}{9} \sigma_w^2 & s = t, \\ \frac{2}{9} \sigma_w^2 & |s - t| = 1, \\ \frac{1}{9} \sigma_w^2 & |s - t| = 2, \\ 0 & |s - t| > 2. \end{cases} \quad (1.14)$$

[Example 1.17](#) shows clearly that the smoothing operation introduces a covariance function that decreases as the separation between the two time points increases and disappears completely when the time points are separated by three or more time points. This particular autocovariance is interesting because it only depends on the time separation or *lag* and not on the absolute location of the points along the series. We shall see later that this dependence suggests a mathematical model for the concept of *weak stationarity*.

Example 1.18 Autocovariance of a Random Walk

For the random walk model, $x_t = \sum_{j=1}^t w_j$, we have

$$\gamma_x(s, t) = \text{cov}(x_s, x_t) = \text{cov}\left(\sum_{j=1}^s w_j, \sum_{k=1}^t w_k\right) = \min\{s, t\} \sigma_w^2,$$

because the w_t are uncorrelated random variables. Note that, as opposed to the previous examples, the autocovariance function of a random walk depends on the particular time values s and t , and not on the time separation or lag. Also, notice that the variance of the random walk, $\text{var}(x_t) = \gamma_x(t, t) = t \sigma_w^2$, increases without bound as time t increases. The effect of this variance increase can be seen in Figure 1.10 where the processes start to move away from their mean functions δt (note that $\delta = 0$ and .2 in that example).

As in classical statistics, it is more convenient to deal with a measure of association between -1 and 1 , and this leads to the following definition.

Definition 1.3 *The autocorrelation function (ACF) is defined as*

$$\rho(s, t) = \frac{\gamma(s, t)}{\sqrt{\gamma(s, s)\gamma(t, t)}}. \quad (1.15)$$

The ACF measures the linear predictability of the series at time t , say x_t , using only the value x_s . We can show easily that $-1 \leq \rho(s, t) \leq 1$ using the Cauchy–Schwarz inequality.^{1.4} If we can predict x_t perfectly from x_s through a linear relationship, $x_t = \beta_0 + \beta_1 x_s$, then the correlation will be $+1$ when $\beta_1 > 0$, and -1 when $\beta_1 < 0$. Hence, we have a rough measure of the ability to forecast the series at time t from the value at time s .

Often, we would like to measure the predictability of another series y_t from the series x_s . Assuming both series have finite variances, we have the following definition.

Definition 1.4 *The cross-covariance function between two series, x_t and y_t , is*

$$\gamma_{xy}(s, t) = \text{cov}(x_s, y_t) = E[(x_s - \mu_{xs})(y_t - \mu_{yt})]. \quad (1.16)$$

There is also a scaled version of the cross-covariance function.

Definition 1.5 *The cross-correlation function (CCF) is given by*

$$\rho_{xy}(s, t) = \frac{\gamma_{xy}(s, t)}{\sqrt{\gamma_x(s, s)\gamma_y(t, t)}}. \quad (1.17)$$

^{1.4} The Cauchy–Schwarz inequality implies $|\gamma(s, t)|^2 \leq \gamma(s, s)\gamma(t, t)$.

We may easily extend the above ideas to the case of more than two series, say, $x_{t1}, x_{t2}, \dots, x_{tr}$; that is, *multivariate time series* with r components. For example, the extension of (1.10) in this case is

$$\gamma_{jk}(s, t) = E[(x_{sj} - \mu_{sj})(x_{tk} - \mu_{tk})] \quad j, k = 1, 2, \dots, r. \quad (1.18)$$

In the definitions above, the autocovariance and cross-covariance functions may change as one moves along the series because the values depend on both s and t , the locations of the points in time. In Example 1.17, the autocovariance function depends on the separation of x_s and x_t , say, $h = |s - t|$, and not on where the points are located in time. As long as the points are separated by h units, the location of the two points does not matter. This notion, called *weak stationarity*, when the mean is constant, is fundamental in allowing us to analyze sample time series data when only a single series is available.

1.4 Stationary Time Series

The preceding definitions of the mean and autocovariance functions are completely general. Although we have not made any special assumptions about the behavior of the time series, many of the preceding examples have hinted that a sort of regularity may exist over time in the behavior of a time series. We introduce the notion of regularity using a concept called *stationarity*.

Definition 1.6 A strictly stationary time series is one for which the probabilistic behavior of every collection of values

$$\{x_{t_1}, x_{t_2}, \dots, x_{t_k}\}$$

is identical to that of the time shifted set

$$\{x_{t_1+h}, x_{t_2+h}, \dots, x_{t_k+h}\}.$$

That is,

$$\Pr\{x_{t_1} \leq c_1, \dots, x_{t_k} \leq c_k\} = \Pr\{x_{t_1+h} \leq c_1, \dots, x_{t_k+h} \leq c_k\} \quad (1.19)$$

for all $k = 1, 2, \dots$, all time points t_1, t_2, \dots, t_k , all numbers c_1, c_2, \dots, c_k , and all time shifts $h = 0, \pm 1, \pm 2, \dots$.

If a time series is strictly stationary, then all of the multivariate distribution functions for subsets of variables must agree with their counterparts in the shifted set for all values of the shift parameter h . For example, when $k = 1$, (1.19) implies that

$$\Pr\{x_s \leq c\} = \Pr\{x_t \leq c\} \quad (1.20)$$

for any time points s and t . This statement implies, for example, that the probability the value of a time series sampled hourly is negative at 1 AM is the same as at 10 AM.

In addition, if the mean function, μ_t , of the series exists, (1.20) implies that $\mu_s = \mu_t$ for all s and t , and hence μ_t must be constant. Note, for example, that a random walk process with drift is *not* strictly stationary because its mean function changes with time; see Example 1.14.

When $k = 2$, we can write (1.19) as

$$\Pr\{x_s \leq c_1, x_t \leq c_2\} = \Pr\{x_{s+h} \leq c_1, x_{t+h} \leq c_2\} \quad (1.21)$$

for any time points s and t and shift h . Thus, if the variance function of the process exists, (1.20)–(1.21) imply that the autocovariance function of the series x_t satisfies

$$\gamma(s, t) = \gamma(s + h, t + h)$$

for all s and t and h . We may interpret this result by saying the autocovariance function of the process depends only on the time difference between s and t , and not on the actual times.

The version of stationarity in Definition 1.6 is too strong for most applications. Moreover, it is difficult to assess strict stationarity from a single data set. Rather than imposing conditions on all possible distributions of a time series, we will use a milder version that imposes conditions only on the first two moments of the series. We now have the following definition.

Definition 1.7 A weakly stationary time series, x_t , is a finite variance process such that

- (i) the mean value function, μ_t , defined in (1.9) is constant and does not depend on time t , and
- (ii) the autocovariance function, $\gamma(s, t)$, defined in (1.10) depends on s and t only through their difference $|s - t|$.

Henceforth, we will use the term **stationary** to mean weakly stationary; if a process is stationary in the strict sense, we will use the term **strictly stationary**.

Stationarity requires regularity in the mean and autocorrelation functions so that these quantities (at least) may be estimated by averaging. It should be clear from the discussion of strict stationarity following Definition 1.6 that a strictly stationary, finite variance, time series is also stationary. The converse is not true unless there are further conditions. One important case where stationarity implies strict stationarity is if the time series is Gaussian [meaning all finite distributions, (1.19), of the series are Gaussian]. We will make this concept more precise at the end of this section.

Because the mean function, $E(x_t) = \mu_t$, of a stationary time series is independent of time t , we will write

$$\mu_t = \mu. \quad (1.22)$$

Also, because the autocovariance function, $\gamma(s, t)$, of a stationary time series, x_t , depends on s and t only through their difference $|s - t|$, we may simplify the notation. Let $s = t + h$, where h represents the time shift or *lag*. Then

$$\gamma(t + h, t) = \text{cov}(x_{t+h}, x_t) = \text{cov}(x_h, x_0) = \gamma(h, 0)$$

because the time difference between times $t+h$ and t is the same as the time difference between times h and 0. Thus, the autocovariance function of a stationary time series does not depend on the time argument t . Henceforth, for convenience, we will drop the second argument of $\gamma(h, 0)$.

Definition 1.8 *The autocovariance function of a stationary time series will be written as*

$$\gamma(h) = \text{cov}(x_{t+h}, x_t) = E[(x_{t+h} - \mu)(x_t - \mu)]. \quad (1.23)$$

Definition 1.9 *The autocorrelation function (ACF) of a stationary time series will be written using (1.15) as*

$$\rho(h) = \frac{\gamma(t+h, t)}{\sqrt{\gamma(t+h, t+h)\gamma(t, t)}} = \frac{\gamma(h)}{\gamma(0)}. \quad (1.24)$$

The Cauchy–Schwarz inequality shows again that $-1 \leq \rho(h) \leq 1$ for all h , enabling one to assess the relative importance of a given autocorrelation value by comparing with the extreme values -1 and 1 .

Example 1.19 Stationarity of White Noise

The mean and autocovariance functions of the white noise series discussed in Example 1.8 and Example 1.16 are easily evaluated as $\mu_{wt} = 0$ and

$$\gamma_w(h) = \text{cov}(w_{t+h}, w_t) = \begin{cases} \sigma_w^2 & h = 0, \\ 0 & h \neq 0. \end{cases}$$

Thus, white noise satisfies the conditions of Definition 1.7 and is weakly stationary or stationary. If the white noise variates are also normally distributed or Gaussian, the series is also strictly stationary, as can be seen by evaluating (1.19) using the fact that the noise would also be iid. The autocorrelation function is given by $\rho_w(0) = 1$ and $\rho(h) = 0$ for $h \neq 0$.

Example 1.20 Stationarity of a Moving Average

The three-point moving average process of Example 1.9 is stationary because, from Example 1.13 and Example 1.17, the mean and autocovariance functions $\mu_{vt} = 0$, and

$$\gamma_v(h) = \begin{cases} \frac{3}{9}\sigma_w^2 & h = 0, \\ \frac{2}{9}\sigma_w^2 & h = \pm 1, \\ \frac{1}{9}\sigma_w^2 & h = \pm 2, \\ 0 & |h| > 2 \end{cases}$$

are independent of time t , satisfying the conditions of Definition 1.7.

The autocorrelation function is given by

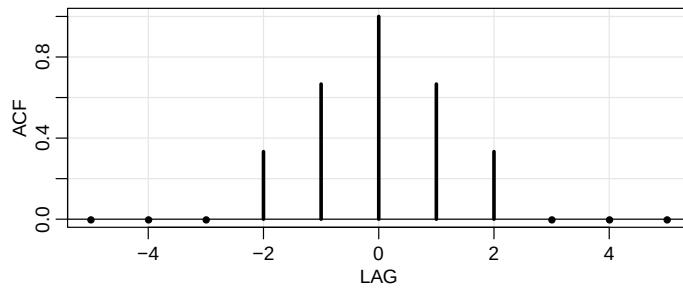


Fig. 1.12. Autocorrelation function of a three-point moving average.

$$\rho_v(h) = \begin{cases} 1 & h = 0, \\ \frac{2}{3} & h = \pm 1, \\ \frac{1}{3} & h = \pm 2, \\ 0 & |h| > 2. \end{cases}$$

Figure 1.12 shows a plot of the autocorrelations as a function of lag h . Note that the ACF is symmetric about lag zero.

Example 1.21 A Random Walk is Not Stationary

A random walk is not stationary because its autocovariance function, $\gamma(s, t) = \min\{s, t\}\sigma_w^2$, depends on time; see Example 1.18 and Problem 1.8. Also, the random walk with drift violates both conditions of Definition 1.7 because, as shown in Example 1.14, the mean function, $\mu_{xt} = \delta t$, is also a function of time t .

Example 1.22 Trend Stationarity

For example, if $x_t = \alpha + \beta t + y_t$, where y_t is stationary, then the mean function is $\mu_{x,t} = E(x_t) = \alpha + \beta t + \mu_y$, which is not independent of time. Therefore, the process is not stationary. The autocovariance function, however, is independent of time, because $\gamma_x(h) = \text{cov}(x_{t+h}, x_t) = E[(x_{t+h} - \mu_{x,t+h})(x_t - \mu_{x,t})] = E[(y_{t+h} - \mu_y)(y_t - \mu_y)] = \gamma_y(h)$. Thus, the model may be considered as having stationary behavior around a linear trend; this behavior is sometimes called *trend stationarity*. An example of such a process is the price of chicken series displayed in Figure 2.1.

The autocovariance function of a stationary process has several special properties. First, $\gamma(h)$ is non-negative definite (see Problem 1.25) ensuring that variances of linear combinations of the variates x_t will never be negative. That is, for any $n \geq 1$, and constants a_1, \dots, a_n ,

$$0 \leq \text{var}(a_1 x_1 + \dots + a_n x_n) = \sum_{j=1}^n \sum_{k=1}^n a_j a_k \gamma(j-k), \quad (1.25)$$

using Property 1.1. Also, the value at $h = 0$, namely

$$\gamma(0) = E[(x_t - \mu)^2] \quad (1.26)$$

is the variance of the time series and the Cauchy–Schwarz inequality implies

$$|\gamma(h)| \leq \gamma(0).$$

A final useful property, noted in a previous example, is that the autocovariance function of a stationary series is symmetric around the origin; that is,

$$\gamma(h) = \gamma(-h) \quad (1.27)$$

for all h . This property follows because

$$\gamma((t+h)-t) = \text{cov}(x_{t+h}, x_t) = \text{cov}(x_t, x_{t+h}) = \gamma(t-(t+h)),$$

which shows how to use the notation as well as proving the result.

When several series are available, a notion of stationarity still applies with additional conditions.

Definition 1.10 Two time series, say, x_t and y_t , are said to be **jointly stationary** if they are each stationary, and the cross-covariance function

$$\gamma_{xy}(h) = \text{cov}(x_{t+h}, y_t) = E[(x_{t+h} - \mu_x)(y_t - \mu_y)] \quad (1.28)$$

is a function only of lag h .

Definition 1.11 The **cross-correlation function (CCF)** of jointly stationary time series x_t and y_t is defined as

$$\rho_{xy}(h) = \frac{\gamma_{xy}(h)}{\sqrt{\gamma_x(0)\gamma_y(0)}}. \quad (1.29)$$

Again, we have the result $-1 \leq \rho_{xy}(h) \leq 1$ which enables comparison with the extreme values -1 and 1 when looking at the relation between x_{t+h} and y_t . The cross-correlation function is not generally symmetric about zero, i.e., typically $\rho_{xy}(h) \neq \rho_{xy}(-h)$. This is an important concept; it should be clear that $\text{cov}(x_2, y_1)$ and $\text{cov}(x_1, y_2)$ need not be the same. It is the case, however, that

$$\rho_{xy}(h) = \rho_{yx}(-h), \quad (1.30)$$

which can be shown by manipulations similar to those used to show (1.27).

Example 1.23 Joint Stationarity

Consider the two series, x_t and y_t , formed from the sum and difference of two successive values of a white noise process, say,

$$x_t = w_t + w_{t-1} \quad \text{and} \quad y_t = w_t - w_{t-1},$$

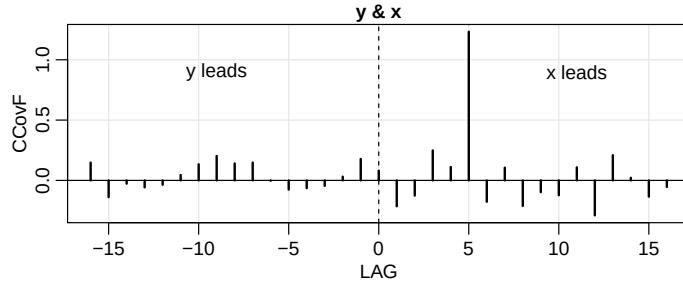


Fig. 1.13. Demonstration of the results of Example 1.24 when $\ell = 5$. The title shows which side leads.

where w_t are independent random variables with zero means and variance σ_w^2 . It is easy to show that $\gamma_x(0) = \gamma_y(0) = 2\sigma_w^2$ and $\gamma_x(1) = \gamma_x(-1) = \sigma_w^2$, $\gamma_y(1) = \gamma_y(-1) = -\sigma_w^2$. Also,

$$\gamma_{xy}(1) = \text{cov}(x_{t+1}, y_t) = \text{cov}(w_{t+1} + w_t, w_t - w_{t-1}) = \sigma_w^2$$

because only one term is nonzero. Similarly, $\gamma_{xy}(0) = 0$, $\gamma_{xy}(-1) = -\sigma_w^2$. We obtain, using (1.29),

$$\rho_{xy}(h) = \begin{cases} 0 & h = 0, \\ 1/2 & h = 1, \\ -1/2 & h = -1, \\ 0 & |h| \geq 2. \end{cases}$$

Clearly, the autocovariance and cross-covariance functions depend only on the lag separation, h , so the series are jointly stationary.

Example 1.24 Prediction Using Cross-Correlation

As a simple example of cross-correlation, consider the problem of determining possible leading or lagging relations between two series x_t and y_t . If the model

$$y_t = Ax_{t-\ell} + w_t$$

holds, the series x_t is said to *lead* y_t for $\ell > 0$ and is said to *lag* y_t for $\ell < 0$. Hence, the analysis of leading and lagging relations might be important in predicting the value of y_t from x_t . Assuming that the noise w_t is uncorrelated with the x_t series, the cross-covariance function can be computed as

$$\begin{aligned} \gamma_{yx}(h) &= \text{cov}(y_{t+h}, x_t) = \text{cov}(Ax_{t+h-\ell} + w_{t+h}, x_t) \\ &= \text{cov}(Ax_{t+h-\ell}, x_t) = A\gamma_x(h - \ell). \end{aligned}$$

Since (Cauchy–Schwarz) the largest absolute value of $\gamma_x(h - \ell)$ is $\gamma_x(0)$, i.e., when $h = \ell$, the cross-covariance function will look like the autocovariance of the input series x_t , and it will have a peak on the positive side if x_t leads y_t and a peak on the negative side if x_t lags y_t . Below is the R code of an example where x_t is white noise, $\ell = 5$, and with $\hat{\gamma}_{yx}(h)$ shown in Figure 1.13.

```
x = rnorm(100)
y = lag(x, -5) + rnorm(100)
ccf(y, x, ylab='CCovF', type='covariance')
```

The concept of weak stationarity forms the basis for much of the analysis performed with time series. The fundamental properties of the mean and autocovariance functions (1.22) and (1.23) are satisfied by many theoretical models that appear to generate plausible sample realizations. In [Example 1.9](#) and [Example 1.10](#), two series were generated that produced stationary looking realizations, and in [Example 1.20](#), we showed that the series in [Example 1.9](#) was, in fact, weakly stationary. Both examples are special cases of the so-called linear process.

Definition 1.12 A linear process, x_t , is defined to be a linear combination of white noise variates w_t , and is given by

$$x_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j w_{t-j}, \quad \sum_{j=-\infty}^{\infty} |\psi_j| < \infty. \quad (1.31)$$

For the linear process (see [Problem 1.11](#)), we may show that the autocovariance function is given by

$$\gamma_x(h) = \sigma_w^2 \sum_{j=-\infty}^{\infty} \psi_{j+h} \psi_j \quad (1.32)$$

for $h \geq 0$; recall that $\gamma_x(-h) = \gamma_x(h)$. This method exhibits the autocovariance function of the process in terms of the lagged products of the coefficients. We only need $\sum_{j=-\infty}^{\infty} \psi_j^2 < \infty$ for the process to have finite variance, but we will discuss this further in [Chapter 5](#). Note that, for [Example 1.9](#), we have $\psi_0 = \psi_{-1} = \psi_1 = 1/3$ and the result in [Example 1.20](#) comes out immediately. The autoregressive series in [Example 1.10](#) can also be put in this form, as can the general autoregressive moving average processes considered in [Chapter 3](#).

Notice that the linear process (1.31) is dependent on the future ($j < 0$), the present ($j = 0$), and the past ($j > 0$). For the purpose of forecasting, a future dependent model will be useless. Consequently, we will focus on processes that do not depend on the future. Such models are called *causal*, and a causal linear process has $\psi_j = 0$ for $j < 0$; we will discuss this further in [Chapter 3](#).

Finally, as previously mentioned, an important case in which a weakly stationary series is also strictly stationary is the normal or Gaussian series.

Definition 1.13 A process, $\{x_t\}$, is said to be a **Gaussian process** if the n -dimensional vectors $x = (x_{t_1}, x_{t_2}, \dots, x_{t_n})'$, for every collection of distinct time points t_1, t_2, \dots, t_n , and every positive integer n , have a multivariate normal distribution.

Defining the $n \times 1$ mean vector $E(x) \equiv \mu = (\mu_{t_1}, \mu_{t_2}, \dots, \mu_{t_n})'$ and the $n \times n$ covariance matrix as $\text{var}(x) \equiv \Gamma = \{\gamma(t_i, t_j); i, j = 1, \dots, n\}$, which is assumed to be positive definite, the multivariate normal density function can be written as

$$f(x) = (2\pi)^{-n/2} |\Gamma|^{-1/2} \exp \left\{ -\frac{1}{2} (x - \mu)' \Gamma^{-1} (x - \mu) \right\}, \quad (1.33)$$

for $x \in \mathbb{R}^n$, where $|\cdot|$ denotes the determinant.

We list some important items regarding linear and Gaussian processes.

- If a Gaussian time series, $\{x_t\}$, is weakly stationary, then μ_t is constant and $\gamma(t_i, t_j) = \gamma(|t_i - t_j|)$, so that the vector μ and the matrix Γ are independent of time. These facts imply that all the finite distributions, (1.33), of the series $\{x_t\}$ depend only on time lag and not on the actual times, and hence the series must be strictly stationary. In a sense, weak stationarity and normality go hand-in-hand in that we will base our analyses on the idea that it is enough for the first two moments to behave nicely. We use the multivariate normal density in the form given above as well as in a modified version, applicable to complex random variables throughout the text.
- A result called the *Wold Decomposition* (Theorem B.5) states that a stationary non-deterministic time series is a causal linear process (but with $\sum \psi_j^2 < \infty$). A linear process need not be Gaussian, but if a time series is Gaussian, then it is a causal linear process with $w_t \sim \text{iid } N(0, \sigma_w^2)$. Hence, **stationary Gaussian processes form the basis of modeling many time series.**
- It is not enough for the marginal distributions to be Gaussian for the process to be Gaussian. It is easy to construct a situation where X and Y are normal, but (X, Y) is not bivariate normal; e.g., let X and Z be independent normals and let $Y = Z$ if $XZ > 0$ and $Y = -Z$ if $XZ \leq 0$.

1.5 Estimation of Correlation

Although the theoretical autocorrelation and cross-correlation functions are useful for describing the properties of certain hypothesized models, most of the analyses must be performed using sampled data. This limitation means the sampled points x_1, x_2, \dots, x_n only are available for estimating the mean, autocovariance, and autocorrelation functions. From the point of view of classical statistics, this poses a problem because we will typically not have iid copies of x_t that are available for estimating the covariance and correlation functions. In the usual situation with only one realization, however, the assumption of stationarity becomes critical. Somehow, we must use averages over this single realization to estimate the population means and covariance functions.

Accordingly, if a time series is stationary, the mean function (1.22) $\mu_t = \mu$ is constant so that we can estimate it by the *sample mean*,

$$\bar{x} = \frac{1}{n} \sum_{t=1}^n x_t. \quad (1.34)$$

In our case, $E(\bar{x}) = \mu$, and the standard error of the estimate is the square root of $\text{var}(\bar{x})$, which can be computed using first principles (recall Property 1.1), and is given by

$$\begin{aligned}
\text{var}(\bar{x}) &= \text{var}\left(\frac{1}{n} \sum_{t=1}^n x_t\right) = \frac{1}{n^2} \text{cov}\left(\sum_{t=1}^n x_t, \sum_{s=1}^n x_s\right) \\
&= \frac{1}{n^2} \left(n\gamma_x(0) + (n-1)\gamma_x(1) + (n-2)\gamma_x(2) + \cdots + \gamma_x(n-1) \right. \\
&\quad \left. + (n-1)\gamma_x(-1) + (n-2)\gamma_x(-2) + \cdots + \gamma_x(1-n) \right) \\
&= \frac{1}{n} \sum_{h=-n}^n \left(1 - \frac{|h|}{n}\right) \gamma_x(h).
\end{aligned} \tag{1.35}$$

If the process is white noise, (1.35) reduces to the familiar σ_x^2/n recalling that $\gamma_x(0) = \sigma_x^2$. Note that, in the case of dependence, the standard error of \bar{x} may be smaller or larger than the white noise case depending on the nature of the correlation structure (see Problem 1.19)

The theoretical autocovariance function, (1.23), is estimated by the sample autocovariance function defined as follows.

Definition 1.14 *The sample autocovariance function is defined as*

$$\hat{\gamma}(h) = n^{-1} \sum_{t=1}^{n-h} (x_{t+h} - \bar{x})(x_t - \bar{x}), \tag{1.36}$$

with $\hat{\gamma}(-h) = \hat{\gamma}(h)$ for $h = 0, 1, \dots, n-1$.

The sum in (1.36) runs over a restricted range because x_{t+h} is not available for $t+h > n$. The estimator in (1.36) is preferred to the one that would be obtained by dividing by $n-h$ because (1.36) is a non-negative definite function. Recall that the autocovariance function of a stationary process is non-negative definite [(1.25); also, see Problem 1.25] ensuring that variances of linear combinations of the variates x_t will never be negative. And because a variance is never negative, the estimate of that variance

$$\widehat{\text{var}}(a_1 x_1 + \cdots + a_n x_n) = \sum_{j=1}^n \sum_{k=1}^n a_j a_k \hat{\gamma}(j-k),$$

should also be non-negative. The estimator in (1.36) guarantees this result, but no such guarantee exists if we divide by $n-h$. Note that neither dividing by n nor $n-h$ in (1.36) yields an unbiased estimator of $\gamma(h)$.

Definition 1.15 *The sample autocorrelation function is defined, analogously to (1.24), as*

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}. \tag{1.37}$$

The sample autocorrelation function has a sampling distribution that allows us to assess whether the data comes from a completely random or white series or whether correlations are statistically significant at some lags.

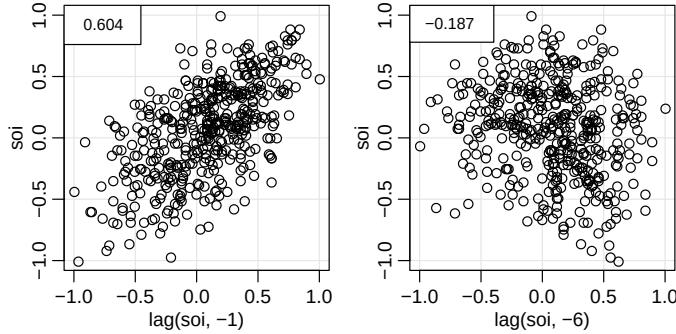


Fig. 1.14. Display for [Example 1.25](#). For the SOI series, the scatterplots show pairs of values one month apart (left) and six months apart (right). The estimated correlation is displayed in the box.

Example 1.25 Sample ACF and Scatterplots

Estimating autocorrelation is similar to estimating of correlation in the usual setup where we have pairs of observations, say (x_i, y_i) , for $i = 1, \dots, n$. For example, if we have time series data x_t for $t = 1, \dots, n$, then the pairs of observations for estimating $\rho(h)$ are the $n - h$ pairs given by $\{(x_t, x_{t+h}); t = 1, \dots, n - h\}$. [Figure 1.14](#) shows an example using the SOI series where $\hat{\rho}(1) = .604$ and $\hat{\rho}(6) = -.187$. The following code was used for [Figure 1.14](#).

```
r = round(acf(soi, 6, plot=FALSE)$acf[-1], 3)) # first 6 sample acf values
[1] 0.604 0.374 0.214 0.050 -0.107 -0.187
par(mfrow=c(1,2))
plot(lag(soi,-1), soi); legend('topleft', legend=r[1])
plot(lag(soi,-6), soi); legend('topleft', legend=r[6])
```

Property 1.2 Large-Sample Distribution of the ACF

Under general conditions,^{1.5} if x_t is white noise, then for n large, the sample ACF, $\hat{\rho}_x(h)$, for $h = 1, 2, \dots, H$, where H is fixed but arbitrary, is approximately normally distributed with zero mean and standard deviation given by

$$\sigma_{\hat{\rho}_x(h)} = \frac{1}{\sqrt{n}}. \quad (1.38)$$

Based on the previous result, we obtain a rough method of assessing whether peaks in $\hat{\rho}(h)$ are significant by determining whether the observed peak is outside the interval $\pm 2/\sqrt{n}$ (or plus/minus two standard errors); for a white noise sequence, approximately 95% of the sample ACFs should be within these limits. The applications of this property develop because many statistical modeling procedures depend on reducing a time series to a white noise series using various kinds of transformations. After such a procedure is applied, the plotted ACFs of the residuals should then lie roughly within the limits given above.

^{1.5} The general conditions are that x_t is iid with finite fourth moment. A sufficient condition for this to hold is that x_t is white Gaussian noise. Precise details are given in [Theorem A.7](#) in [Appendix A](#).

Example 1.26 A Simulated Time Series

To compare the sample ACF for various sample sizes to the theoretical ACF, consider a contrived set of data generated by tossing a fair coin, letting $x_t = 1$ when a head is obtained and $x_t = -1$ when a tail is obtained. Then, construct y_t as

$$y_t = 5 + x_t - .7x_{t-1}. \quad (1.39)$$

To simulate data, we consider two cases, one with a small sample size ($n = 10$) and another with a moderate sample size ($n = 100$).

```
set.seed(101010)
x1 = 2*rbinom(11, 1, .5) - 1    # simulated sequence of coin tosses
x2 = 2*rbinom(101, 1, .5) - 1
y1 = 5 + filter(x1, sides=1, filter=c(1,-.7))[-1]
y2 = 5 + filter(x2, sides=1, filter=c(1,-.7))[-1]
plot.ts(y1, type='s'); plot.ts(y2, type='s')  # plot both series (not shown)
c(mean(y1), mean(y2))                      # the sample means
[1] 5.080  5.002
acf(y1, lag.max=4, plot=FALSE)  # 1/sqrt(10) = .32
  Autocorrelations of series 'y1', by lag
    0     1     2     3     4
  1.000 -0.688  0.425 -0.306 -0.007
acf(y2, lag.max=4, plot=FALSE)  # 1/sqrt(100) = .1
  Autocorrelations of series 'y2', by lag
    0     1     2     3     4
  1.000 -0.480 -0.002 -0.004  0.000
# Note that the sample ACF at lag zero is always 1 (Why?).
```

The theoretical ACF can be obtained from the model (1.39) using the fact that the mean of x_t is zero and the variance of x_t is one. It can be shown that

$$\rho_y(1) = \frac{-0.7}{1 + 0.7^2} = -0.47$$

and $\rho_y(h) = 0$ for $|h| > 1$ (Problem 1.24). It is interesting to compare the theoretical ACF with sample ACFs for the realization where $n = 10$ and the other realization where $n = 100$; note the increased variability in the smaller size sample.

Example 1.27 ACF of a Speech Signal

Computing the sample ACF as in the previous example can be thought of as matching the time series h units in the future, say, x_{t+h} against itself, x_t . Figure 1.15 shows the ACF of the speech series of Figure 1.3. The original series appears to contain a sequence of repeating short signals. The ACF confirms this behavior, showing repeating peaks spaced at about 106-109 points. Autocorrelation functions of the short signals appear, spaced at the intervals mentioned above. The distance between the repeating signals is known as the *pitch period* and is a fundamental parameter of interest in systems that encode and decipher speech. Because the series is sampled at 10,000 points per second, the pitch period appears to be between .0106 and .0109 seconds. To compute the sample ACF in R, use `acf(speech, 250)`.

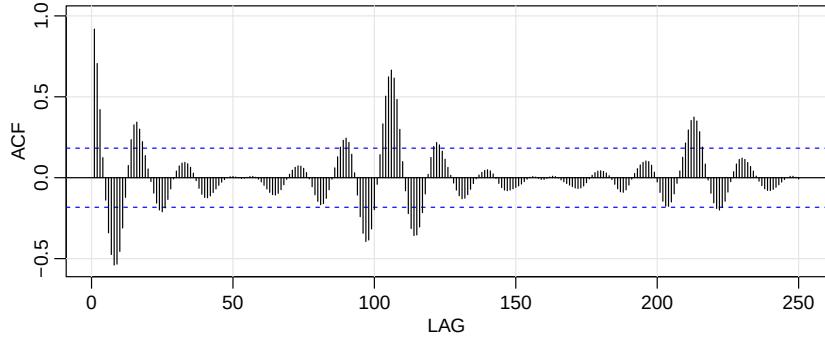


Fig. 1.15. ACF of the speech series.

Definition 1.16 The estimators for the cross-covariance function, $\hat{\gamma}_{xy}(h)$, as given in (1.28) and the cross-correlation, $\hat{\rho}_{xy}(h)$, in (1.11) are given, respectively, by the **sample cross-covariance function**

$$\hat{\gamma}_{xy}(h) = n^{-1} \sum_{t=1}^{n-h} (x_{t+h} - \bar{x})(y_t - \bar{y}), \quad (1.40)$$

where $\hat{\gamma}_{xy}(-h) = \hat{\gamma}_{yx}(h)$ determines the function for negative lags, and the **sample cross-correlation function**

$$\hat{\rho}_{xy}(h) = \frac{\hat{\gamma}_{xy}(h)}{\sqrt{\hat{\gamma}_x(0)\hat{\gamma}_y(0)}}. \quad (1.41)$$

The sample cross-correlation function can be examined graphically as a function of lag h to search for leading or lagging relations in the data using the property mentioned in [Example 1.24](#) for the theoretical cross-covariance function. Because $-1 \leq \hat{\rho}_{xy}(h) \leq 1$, the practical importance of peaks can be assessed by comparing their magnitudes with their theoretical maximum values. Furthermore, for x_t and y_t independent linear processes of the form (1.31), we have the following property.

Property 1.3 Large-Sample Distribution of Cross-Correlation

The large sample distribution of $\hat{\rho}_{xy}(h)$ is normal with mean zero and

$$\sigma_{\hat{\rho}_{xy}} = \frac{1}{\sqrt{n}} \quad (1.42)$$

if at least one of the processes is independent white noise (see [Theorem A.8](#)).

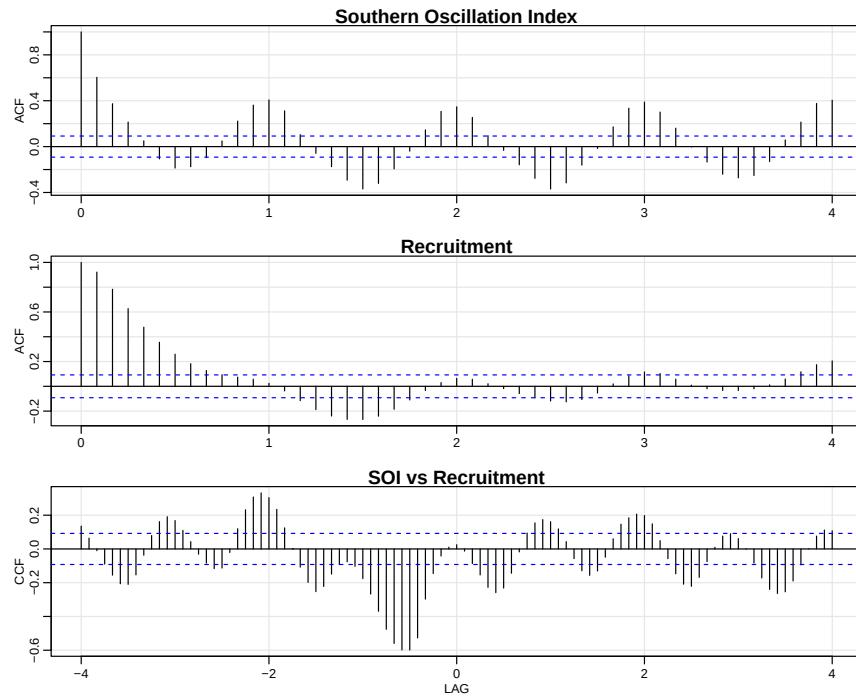


Fig. 1.16. Sample ACFs of the SOI series (top) and of the Recruitment series (middle), and the sample CCF of the two series (bottom); negative lags indicate SOI leads Recruitment. The lag axes are in terms of seasons (12 months).

Example 1.28 SOI and Recruitment Correlation Analysis

The autocorrelation and cross-correlation functions are also useful for analyzing the joint behavior of two stationary series whose behavior may be related in some unspecified way. In Example 1.5 (see Figure 1.5), we have considered simultaneous monthly readings of the SOI and the number of new fish (Recruitment) computed from a model. Figure 1.16 shows the autocorrelation and cross-correlation functions (ACFs and CCF) for these two series. Both of the ACFs exhibit periodicities corresponding to the correlation between values separated by 12 units. Observations 12 months or one year apart are strongly positively correlated, as are observations at multiples such as 24, 36, 48, Observations separated by six months are negatively correlated, showing that positive excursions tend to be associated with negative excursions six months removed.

The sample CCF in Figure 1.16, however, shows some departure from the cyclic component of each series and there is an obvious peak at $h = -6$. This result implies that SOI measured at time $t - 6$ months is associated with the Recruitment series at time t . We could say the SOI leads the Recruitment series by six months. The sign of the CCF is negative, leading to the conclusion that the two series move in different directions; that is, increases in SOI lead to decreases in Recruitment.

and vice versa. We will discover in [Chapter 2](#) that there is a relationship between the series, but the relationship is nonlinear. The dashed lines shown on the plots indicate $\pm 2/\sqrt{453}$ [see (1.42)], but since neither series is noise, these lines do not apply. To reproduce [Figure 1.16](#) in R, use the following commands:

```
par(mfrow=c(3,1))
acf(soi, 48, main="Southern Oscillation Index")
acf(rec, 48, main="Recruitment")
ccf(soi, rec, 48, main="SOI vs Recruitment", ylab="CCF")
```

Example 1.29 Prewhitening and Cross Correlation Analysis

Although we do not have all the tools necessary yet, it is worthwhile to discuss the idea of prewhitening a series prior to a cross-correlation analysis. [The basic idea is simple; in order to use Property 1.3, at least one of the series must be white noise.](#) If this is not the case, there is no simple way to tell if a cross-correlation estimate is significantly different from zero. Hence, in [Example 1.28](#), we were only guessing at the linear dependence relationship between SOI and Recruitment.

For example, in [Figure 1.17](#) we generated two series, x_t and y_t , for $t = 1, \dots, 120$ independently as

$$x_t = 2 \cos(2\pi t \frac{1}{12}) + w_{t1} \quad \text{and} \quad y_t = 2 \cos(2\pi [t+5] \frac{1}{12}) + w_{t2}$$

where $\{w_{t1}, w_{t2}; t = 1, \dots, 120\}$ are all independent standard normals. The series are made to resemble SOI and Recruitment. The generated data are shown in the top row of the figure. The middle row of [Figure 1.17](#) shows the sample ACF of each series, each of which exhibits the cyclic nature of each series. The bottom row (left) of [Figure 1.17](#) shows the sample CCF between x_t and y_t , which appears to show cross-correlation even though the series are independent. The bottom row (right) also displays the sample CCF between x_t and the prewhitened y_t , which shows that the two sequences are uncorrelated. By prewhitening y_t , we mean that the signal has been removed from the data by running a regression of y_t on $\cos(2\pi t)$ and $\sin(2\pi t)$ [see [Example 2.10](#)] and then putting $\tilde{y}_t = y_t - \hat{y}_t$, where \hat{y}_t are the predicted values from the regression.

The following code will reproduce [Figure 1.17](#).

```
set.seed(1492)
num=120; t=1:num
X = ts(2*cos(2*pi*t/12) + rnorm(num), freq=12)
Y = ts(2*cos(2*pi*(t+5)/12) + rnorm(num), freq=12)
Yw = resid( lm(Y~ cos(2*pi*t/12) + sin(2*pi*t/12), na.action=NULL) )
par(mfrow=c(3,2), mgp=c(1.6,.6,0), mar=c(3,3,1,1) )
plot(X)
plot(Y)
acf(X,48, ylab='ACF(X)')
acf(Y,48, ylab='ACF(Y)')
ccf(X,Y,24, ylab='CCF(X,Y)')
ccf(X,Yw,24, ylab='CCF(X,Yw)', ylim=c(-.6,.6))
```

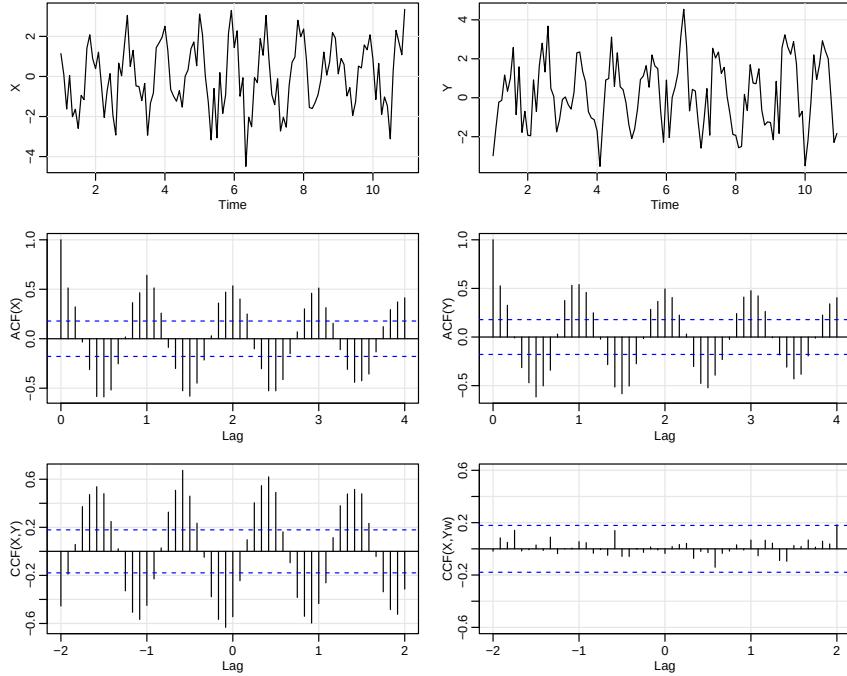


Fig. 1.17. Display for *Example 1.29*. Top row: The generated series. Middle row: The sample ACF of each series. Bottom row: The sample CCF of the series (left) and the sample CCF of the first series with the prewhitened second series (right).

1.6 Vector-Valued and Multidimensional Series

We frequently encounter situations in which the relationships between a number of jointly measured time series are of interest. For example, in the previous sections, we considered discovering the relationships between the SOI and Recruitment series. Hence, it will be useful to consider the notion of a *vector time series* $x_t = (x_{t1}, x_{t2}, \dots, x_{tp})'$, which contains as its components p univariate time series. We denote the $p \times 1$ column vector of the observed series as x_t . The row vector x_t' is its transpose. For the stationary case, the $p \times 1$ mean vector

$$\mu = E(x_t) \quad (1.43)$$

of the form $\mu = (\mu_{t1}, \mu_{t2}, \dots, \mu_{tp})'$ and the $p \times p$ autocovariance matrix

$$\Gamma(h) = E[(x_{t+h} - \mu)(x_t - \mu)'] \quad (1.44)$$

can be defined, where the elements of the matrix $\Gamma(h)$ are the cross-covariance functions

$$\gamma_{ij}(h) = E[(x_{t+h,i} - \mu_i)(x_{tj} - \mu_j)] \quad (1.45)$$

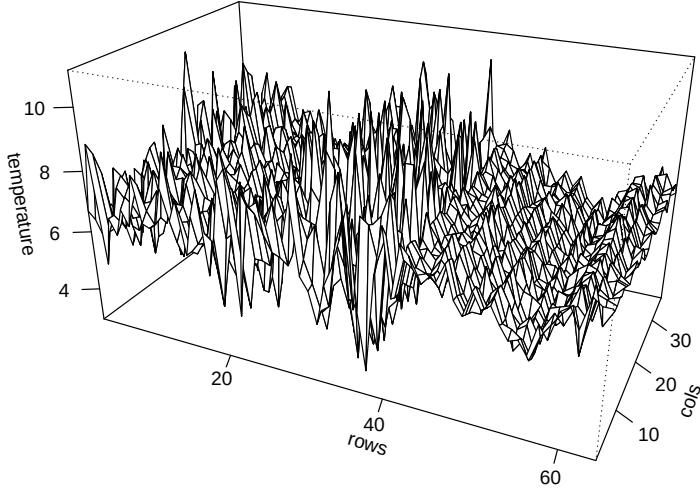


Fig. 1.18. Two-dimensional time series of temperature measurements taken on a rectangular field (64×36 with 17-foot spacing). Data are from Bazza et al. (1988).

for $i, j = 1, \dots, p$. Because $\gamma_{ij}(h) = \gamma_{ji}(-h)$, it follows that

$$\Gamma(-h) = \Gamma'(h). \quad (1.46)$$

Now, the *sample autocovariance matrix* of the vector series x_t is the $p \times p$ matrix of sample cross-covariances, defined as

$$\hat{\Gamma}(h) = n^{-1} \sum_{t=1}^{n-h} (x_{t+h} - \bar{x})(x_t - \bar{x})', \quad (1.47)$$

where

$$\bar{x} = n^{-1} \sum_{t=1}^n x_t \quad (1.48)$$

denotes the $p \times 1$ *sample mean vector*. The symmetry property of the theoretical autocovariance (1.46) extends to the sample autocovariance (1.47), which is defined for negative values by taking

$$\hat{\Gamma}(-h) = \hat{\Gamma}(h)'. \quad (1.49)$$

In many applied problems, an observed series may be indexed by more than time alone. For example, the position in space of an experimental unit might be described by two coordinates, say, s_1 and s_2 . We may proceed in these cases by defining a *multidimensional process* x_s as a function of the $r \times 1$ vector $s = (s_1, s_2, \dots, s_r)'$, where s_i denotes the coordinate of the i th index.

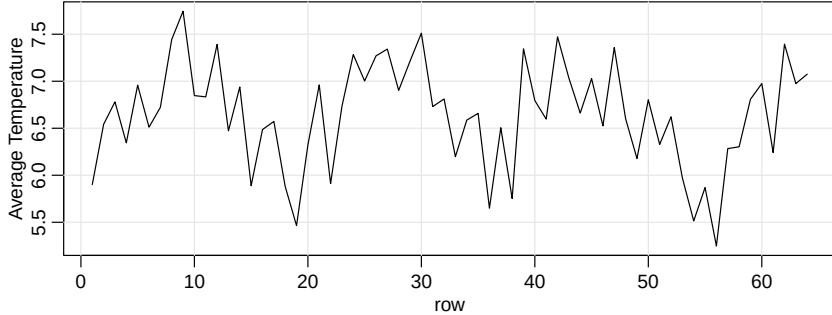


Fig. 1.19. Row averages of the two-dimensional soil temperature profile. $\bar{x}_{s_1,\cdot} = \sum_{s_2} x_{s_1,s_2}/36$.

Example 1.30 Soil Surface Temperatures

As an example, the two-dimensional ($r = 2$) temperature series x_{s_1,s_2} in Figure 1.18 is indexed by a row number s_1 and a column number s_2 that represent positions on a 64×36 spatial grid set out on an agricultural field. The value of the temperature measured at row s_1 and column s_2 , is denoted by $x_s = x_{s_1,s_2}$. We can note from the two-dimensional plot that a distinct change occurs in the character of the two-dimensional surface starting at about row 40, where the oscillations along the row axis become fairly stable and periodic. For example, averaging over the 36 columns, we may compute an average value for each s_1 as in Figure 1.19. It is clear that the noise present in the first part of the two-dimensional series is nicely averaged out, and we see a clear and consistent temperature signal.

To generate Figure 1.18 and Figure 1.19 in R, use the following commands:

```
persp(1:64, 1:36, soiltemp, phi=25, theta=25, scale=FALSE, expand=4,
      ticktype="detailed", xlab="rows", ylab="cols", zlab="temperature")
plot.ts(rowMeans(soiltemp), xlab="row", ylab="Average Temperature")
```

The *autocovariance function* of a stationary multidimensional process, x_s , can be defined as a function of the multidimensional lag vector, say, $h = (h_1, h_2, \dots, h_r)'$, as

$$\gamma(h) = E[(x_{s+h} - \mu)(x_s - \mu)], \quad (1.50)$$

where

$$\mu = E(x_s) \quad (1.51)$$

does not depend on the spatial coordinate s . For the two dimensional temperature process, (1.50) becomes

$$\gamma(h_1, h_2) = E[(x_{s_1+h_1, s_2+h_2} - \mu)(x_{s_1, s_2} - \mu)], \quad (1.52)$$

which is a function of lag, both in the row (h_1) and column (h_2) directions.

The *multidimensional sample autocovariance function* is defined as

$$\hat{\gamma}(h) = (S_1 S_2 \cdots S_r)^{-1} \sum_{s_1} \sum_{s_2} \cdots \sum_{s_r} (x_{s+h} - \bar{x})(x_s - \bar{x}), \quad (1.53)$$

where $s = (s_1, s_2, \dots, s_r)'$ and the range of summation for each argument is $1 \leq s_i \leq S_i - h_i$, for $i = 1, \dots, r$. The mean is computed over the r -dimensional array, that is,

$$\bar{x} = (S_1 S_2 \cdots S_r)^{-1} \sum_{s_1} \sum_{s_2} \cdots \sum_{s_r} x_{s_1, s_2, \dots, s_r}, \quad (1.54)$$

where the arguments s_i are summed over $1 \leq s_i \leq S_i$. The multidimensional sample autocorrelation function follows, as usual, by taking the scaled ratio

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}. \quad (1.55)$$

Example 1.31 Sample ACF of the Soil Temperature Series

The autocorrelation function of the two-dimensional (2d) temperature process can be written in the form

$$\hat{\rho}(h_1, h_2) = \frac{\hat{\gamma}(h_1, h_2)}{\hat{\gamma}(0, 0)},$$

where

$$\hat{\gamma}(h_1, h_2) = (S_1 S_2)^{-1} \sum_{s_1} \sum_{s_2} (x_{s_1+h_1, s_2+h_2} - \bar{x})(x_{s_1, s_2} - \bar{x})$$

[Figure 1.20](#) shows the autocorrelation function for the temperature data, and we note the systematic periodic variation that appears along the rows. The autocovariance over columns seems to be strongest for $h_1 = 0$, implying columns may form replicates of some underlying process that has a periodicity over the rows. This idea can be investigated by examining the mean series over columns as shown in [Figure 1.19](#).

The easiest way (that we know of) to calculate a 2d ACF in R is by using the fast Fourier transform (FFT) as shown below. Unfortunately, the material needed to understand this approach is given in [Chapter 4, Section 4.3](#). The 2d autocovariance function is obtained in two steps and is contained in `cs` below; $\hat{\gamma}(0, 0)$ is the (1,1) element so that $\hat{\rho}(h_1, h_2)$ is obtained by dividing each element by that value. The 2d ACF is contained in `rs` below, and the rest of the code is simply to arrange the results to yield a nice display.

```
fs = Mod(fft(soiltemp-mean(soiltemp)))^2/(64*36)
cs = Re(fft(fs, inverse=TRUE)/sqrt(64*36)) # ACovF
rs = cs/cs[1,1] # ACF
rs2 = cbind(rs[1:41,21:2], rs[1:41,1:21])
rs3 = rbind(rs2[41:2,], rs2)
par(mar = c(1,2.5,0,0)+.1)
persp(-40:40, -20:20, rs3, phi=30, theta=30, expand=30, scale="FALSE",
      ticktype="detailed", xlab="row lags", ylab="column lags",
      zlab="ACF")
```

The sampling requirements for multidimensional processes are rather severe because values must be available over some uniform grid in order to compute the ACF. In some areas of application, such as in soil science, we may prefer to sample a limited number of rows or *transects* and hope these are essentially replicates of the

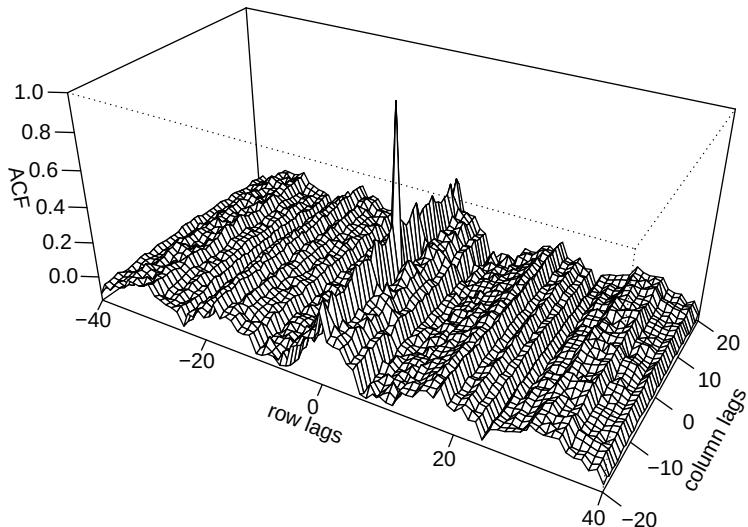


Fig. 1.20. Two-dimensional autocorrelation function for the soil temperature data.

basic underlying phenomenon of interest. One-dimensional methods can then be applied. When observations are irregular in time space, modifications to the estimators need to be made. Systematic approaches to the problems introduced by irregularly spaced observations have been developed by Journel and Huijbregts (1978) or Cressie (1993). We shall not pursue such methods in detail here, but it is worth noting that the introduction of the *variogram*

$$2V_x(h) = \text{var}\{x_{s+h} - x_s\} \quad (1.56)$$

and its sample estimator

$$2\hat{V}_x(h) = \frac{1}{N(h)} \sum_s (x_{s+h} - x_s)^2 \quad (1.57)$$

play key roles, where $N(h)$ denotes both the number of points located within h , and the sum runs over the points in the neighborhood. Clearly, substantial indexing difficulties will develop from estimators of the kind, and often it will be difficult to find non-negative definite estimators for the covariance function. **Problem 1.27** investigates the relation between the variogram and the autocovariance function in the stationary case.

Problems

Section 1.1

1.1 To compare the earthquake and explosion signals, plot the data displayed in Figure 1.7 on the same graph using different colors or different line types and comment on the results. (The R code in Example 1.11 may be of help on how to add lines to existing plots.)

1.2 Consider a signal-plus-noise model of the general form $x_t = s_t + w_t$, where w_t is Gaussian white noise with $\sigma_w^2 = 1$. Simulate and plot $n = 200$ observations from each of the following two models.

(a) $x_t = s_t + w_t$, for $t = 1, \dots, 200$, where

$$s_t = \begin{cases} 0, & t = 1, \dots, 100 \\ 10 \exp\left\{-\frac{(t-100)}{20}\right\} \cos(2\pi t/4), & t = 101, \dots, 200. \end{cases}$$

Hint:

```
s = c(rep(0,100), 10*exp(-(1:100)/20)*cos(2*pi*1:100/4))
x = s + rnorm(200)
plot.ts(x)
```

(b) $x_t = s_t + w_t$, for $t = 1, \dots, 200$, where

$$s_t = \begin{cases} 0, & t = 1, \dots, 100 \\ 10 \exp\left\{-\frac{(t-100)}{200}\right\} \cos(2\pi t/4), & t = 101, \dots, 200. \end{cases}$$

(c) Compare the general appearance of the series (a) and (b) with the earthquake series and the explosion series shown in Figure 1.7. In addition, plot (or sketch) and compare the signal modulators (a) $\exp\{-t/20\}$ and (b) $\exp\{-t/200\}$, for $t = 1, 2, \dots, 100$.

Section 1.2

1.3 (a) Generate $n = 100$ observations from the autoregression

$$x_t = -0.9x_{t-2} + w_t$$

with $\sigma_w = 1$, using the method described in Example 1.10. Next, apply the moving average filter

$$v_t = (x_t + x_{t-1} + x_{t-2} + x_{t-3})/4$$

to x_t , the data you generated. Now plot x_t as a line and superimpose v_t as a dashed line. Comment on the behavior of x_t and how applying the moving average filter changes that behavior. [Hints: Use `v = filter(x, rep(1/4, 4), sides = 1)` for the filter and note that the R code in Example 1.11 may be of help on how to add lines to existing plots.]

(b) Repeat (a) but with

$$x_t = \cos(2\pi t/4).$$

(c) Repeat (b) but with added $N(0, 1)$ noise,

$$x_t = \cos(2\pi t/4) + w_t.$$

(d) Compare and contrast (a)–(c); i.e., how does the moving average change each series.

Section 1.3

1.4 Show that the autocovariance function can be written as

$$\gamma(s, t) = E[(x_s - \mu_s)(x_t - \mu_t)] = E(x_s x_t) - \mu_s \mu_t,$$

where $E[x_t] = \mu_t$.

1.5 For the two series, x_t , in **Problem 1.2** (a) and (b):

- (a) Compute and plot the mean functions $\mu_x(t)$, for $t = 1, \dots, 200$.
- (b) Calculate the autocovariance functions, $\gamma_x(s, t)$, for $s, t = 1, \dots, 200$.

Section 1.4

1.6 Consider the time series

$$x_t = \beta_1 + \beta_2 t + w_t,$$

where β_1 and β_2 are known constants and w_t is a white noise process with variance σ_w^2 .

- (a) Determine whether x_t is stationary.
- (b) Show that the process $y_t = x_t - x_{t-1}$ is stationary.
- (c) Show that the mean of the moving average

$$v_t = \frac{1}{2q+1} \sum_{j=-q}^q x_{t-j}$$

is $\beta_1 + \beta_2 t$, and give a simplified expression for the autocovariance function.

1.7 For a moving average process of the form

$$x_t = w_{t-1} + 2w_t + w_{t+1},$$

where w_t are independent with zero means and variance σ_w^2 , determine the autocovariance and autocorrelation functions as a function of lag $h = s - t$ and plot the ACF as a function of h .

1.8 Consider the random walk with drift model

$$x_t = \delta + x_{t-1} + w_t,$$

for $t = 1, 2, \dots$, with $x_0 = 0$, where w_t is white noise with variance σ_w^2 .

- (a) Show that the model can be written as $x_t = \delta t + \sum_{k=1}^t w_k$.
- (b) Find the mean function and the autocovariance function of x_t .
- (c) Argue that x_t is not stationary.
- (d) Show $\rho_x(t-1, t) = \sqrt{\frac{t-1}{t}} \rightarrow 1$ as $t \rightarrow \infty$. What is the implication of this result?
- (e) Suggest a transformation to make the series stationary, and prove that the transformed series is stationary. (Hint: See Problem 1.6b.)

1.9 A time series with a periodic component can be constructed from

$$x_t = U_1 \sin(2\pi\omega_0 t) + U_2 \cos(2\pi\omega_0 t),$$

where U_1 and U_2 are independent random variables with zero means and $E(U_1^2) = E(U_2^2) = \sigma^2$. The constant ω_0 determines the period or time it takes the process to make one complete cycle. Show that this series is weakly stationary with autocovariance function

$$\gamma(h) = \sigma^2 \cos(2\pi\omega_0 h).$$

1.10 Suppose we would like to predict a single stationary series x_t with zero mean and autocorrelation function $\gamma(h)$ at some time in the future, say, $t + \ell$, for $\ell > 0$.

- (a) If we predict using only x_t and some scale multiplier A , show that the mean-square prediction error

$$MSE(A) = E[(x_{t+\ell} - Ax_t)^2]$$

is minimized by the value

$$A = \rho(\ell).$$

- (b) Show that the minimum mean-square prediction error is

$$MSE(A) = \gamma(0)[1 - \rho^2(\ell)].$$

- (c) Show that if $x_{t+\ell} = Ax_t$, then $\rho(\ell) = 1$ if $A > 0$, and $\rho(\ell) = -1$ if $A < 0$.

1.11 Consider the linear process defined in (1.31).

- (a) Verify that the autocovariance function of the process is given by (1.32). Use the result to verify your answer to Problem 1.7. Hint: For $h \geq 0$, $\text{cov}(x_{t+h}, x_t) = \text{cov}(\sum_k \psi_k w_{t+h-k}, \sum_j \psi_j w_{t-j})$. For each $j \in \mathbb{Z}$, the only “survivor” will be when $k = h + j$.
- (b) Show that x_t exists as a limit in mean square (see Appendix A).

1.12 For two weakly stationary series x_t and y_t , verify (1.30).

1.13 Consider the two series

$$x_t = w_t$$

$$y_t = w_t - \theta w_{t-1} + u_t,$$

where w_t and u_t are independent white noise series with variances σ_w^2 and σ_u^2 , respectively, and θ is an unspecified constant.

- (a) Express the ACF, $\rho_y(h)$, for $h = 0, \pm 1, \pm 2, \dots$ of the series y_t as a function of σ_w^2 , σ_u^2 , and θ .
- (b) Determine the CCF, $\rho_{xy}(h)$ relating x_t and y_t .
- (c) Show that x_t and y_t are jointly stationary.

1.14 Let x_t be a stationary normal process with mean μ_x and autocovariance function $\gamma(h)$. Define the nonlinear time series

$$y_t = \exp\{x_t\}.$$

- (a) Express the mean function $E(y_t)$ in terms of μ_x and $\gamma(0)$. The moment generating function of a normal random variable x with mean μ and variance σ^2 is

$$M_x(\lambda) = E[\exp\{\lambda x\}] = \exp\left\{\mu\lambda + \frac{1}{2}\sigma^2\lambda^2\right\}.$$

- (b) Determine the autocovariance function of y_t . The sum of the two normal random variables $x_{t+h} + x_t$ is still a normal random variable.

1.15 Let w_t , for $t = 0, \pm 1, \pm 2, \dots$ be a normal white noise process, and consider the series

$$x_t = w_t w_{t-1}.$$

Determine the mean and autocovariance function of x_t , and state whether it is stationary.

1.16 Consider the series

$$x_t = \sin(2\pi Ut),$$

$t = 1, 2, \dots$, where U has a uniform distribution on the interval $(0, 1)$.

- (a) Prove x_t is weakly stationary.
- (b) Prove x_t is not strictly stationary.

1.17 Suppose we have the linear process x_t generated by

$$x_t = w_t - \theta w_{t-1},$$

$t = 0, 1, 2, \dots$, where $\{w_t\}$ is independent and identically distributed with characteristic function $\phi_w(\cdot)$, and θ is a fixed constant. [Replace "characteristic function" with "moment generating function" if instructed to do so.]

- (a) Express the joint characteristic function of x_1, x_2, \dots, x_n , say,

$$\phi_{x_1, x_2, \dots, x_n}(\lambda_1, \lambda_2, \dots, \lambda_n),$$

in terms of $\phi_w(\cdot)$.

- (b) Deduce from (a) that x_t is strictly stationary.

1.18 Suppose that x_t is a linear process of the form (1.31). Prove

$$\sum_{h=-\infty}^{\infty} |\gamma(h)| < \infty.$$

Section 1.5

1.19 Suppose $x_t = \mu + w_t + \theta w_{t-1}$, where $w_t \sim wn(0, \sigma_w^2)$.

- (a) Show that mean function is $E(x_t) = \mu$.
- (b) Show that the autocovariance function of x_t is given by $\gamma_x(0) = \sigma_w^2(1 + \theta^2)$, $\gamma_x(\pm 1) = \sigma_w^2\theta$, and $\gamma_x(h) = 0$ otherwise.
- (c) Show that x_t is stationary for all values of $\theta \in \mathbb{R}$.
- (d) Use (1.35) to calculate $\text{var}(\bar{x})$ for estimating μ when (i) $\theta = 1$, (ii) $\theta = 0$, and (iii) $\theta = -1$
- (e) In time series, the sample size n is typically large, so that $\frac{(n-1)}{n} \approx 1$. With this as a consideration, comment on the results of part (d); in particular, how does the accuracy in the estimate of the mean μ change for the three different cases?

1.20 (a) Simulate a series of $n = 500$ Gaussian white noise observations as in [Example 1.8](#) and compute the sample ACF, $\hat{\rho}(h)$, to lag 20. Compare the sample ACF you obtain to the actual ACF, $\rho(h)$. [Recall [Example 1.19](#).]
 (b) Repeat part (a) using only $n = 50$. How does changing n affect the results?

1.21 (a) Simulate a series of $n = 500$ moving average observations as in [Example 1.9](#) and compute the sample ACF, $\hat{\rho}(h)$, to lag 20. Compare the sample ACF you obtain to the actual ACF, $\rho(h)$. [Recall [Example 1.20](#).]
 (b) Repeat part (a) using only $n = 50$. How does changing n affect the results?

1.22 Although the model in [Problem 1.2\(a\)](#) is not stationary (Why?), the sample ACF can be informative. For the data you generated in that problem, calculate and plot the sample ACF, and then comment.

1.23 Simulate a series of $n = 500$ observations from the signal-plus-noise model presented in [Example 1.12](#) with $\sigma_w^2 = 1$. Compute the sample ACF to lag 100 of the data you generated and comment.

1.24 For the time series y_t described in [Example 1.26](#), verify the stated result that $\rho_y(1) = -.47$ and $\rho_y(h) = 0$ for $h > 1$.

1.25 A real-valued function $g(t)$, defined on the integers, is non-negative definite if and only if

$$\sum_{i=1}^n \sum_{j=1}^n a_i g(t_i - t_j) a_j \geq 0$$

for all positive integers n and for all vectors $a = (a_1, a_2, \dots, a_n)'$ and $t = (t_1, t_2, \dots, t_n)'$. For the matrix $G = \{g(t_i - t_j); i, j = 1, 2, \dots, n\}$, this implies that $a'G a \geq 0$ for all vectors a . It is called positive definite if we can replace ' \geq ' with ' $>$ ' for all $a \neq 0$, the zero vector.

- (a) Prove that $\gamma(h)$, the autocovariance function of a stationary process, is a non-negative definite function.
- (b) Verify that the sample autocovariance $\hat{\gamma}(h)$ is a non-negative definite function.

Section 1.6

1.26 Consider a collection of time series $x_{1t}, x_{2t}, \dots, x_{Nt}$ that are observing some common signal μ_t observed in noise processes $e_{1t}, e_{2t}, \dots, e_{Nt}$, with a model for the j -th observed series given by

$$x_{jt} = \mu_t + e_{jt}.$$

Suppose the noise series have zero means and are uncorrelated for different j . The common autocovariance functions of all series are given by $\gamma_e(s, t)$. Define the sample mean

$$\bar{x}_t = \frac{1}{N} \sum_{j=1}^N x_{jt}.$$

- (a) Show that $E[\bar{x}_t] = \mu_t$.
- (b) Show that $E[(\bar{x}_t - \mu)^2] = N^{-1} \gamma_e(t, t)$.
- (c) How can we use the results in estimating the common signal?

1.27 A concept used in *geostatistics*, see Journel and Huijbregts (1978) or Cressie (1993), is that of the *variogram*, defined for a spatial process x_s , $s = (s_1, s_2)$, for $s_1, s_2 = 0, \pm 1, \pm 2, \dots$, as

$$V_x(h) = \frac{1}{2} E[(x_{s+h} - x_s)^2],$$

where $h = (h_1, h_2)$, for $h_1, h_2 = 0, \pm 1, \pm 2, \dots$. Show that, for a stationary process, the variogram and autocovariance functions can be related through

$$V_x(h) = \gamma(0) - \gamma(h),$$

where $\gamma(h)$ is the usual lag h covariance function and $0 = (0, 0)$. Note the easy extension to any spatial dimension.

The following problems require the material given in Appendix A

1.28 Suppose $x_t = \beta_0 + \beta_1 t$, where β_0 and β_1 are constants. Prove as $n \rightarrow \infty$, $\hat{\rho}_x(h) \rightarrow 1$ for fixed h , where $\hat{\rho}_x(h)$ is the ACF (1.37).

1.29 (a) Suppose x_t is a weakly stationary time series with mean zero and with absolutely summable autocovariance function, $\gamma(h)$, such that

$$\sum_{h=-\infty}^{\infty} \gamma(h) = 0.$$

Prove that $\sqrt{n} \bar{x} \xrightarrow{P} 0$, where \bar{x} is the sample mean (1.34).

(b) Give an example of a process that satisfies the conditions of part (a). What is special about this process?

1.30 Let x_t be a linear process of the form (A.43)–(A.44). If we define

$$\tilde{\gamma}(h) = n^{-1} \sum_{t=1}^n (x_{t+h} - \mu_x)(x_t - \mu_x),$$

show that

$$n^{1/2}(\tilde{\gamma}(h) - \hat{\gamma}(h)) = o_p(1).$$

Hint: The Markov Inequality

$$\Pr\{|x| \geq \epsilon\} < \frac{\text{E}|x|}{\epsilon}$$

can be helpful for the cross-product terms.

1.31 For a linear process of the form

$$x_t = \sum_{j=0}^{\infty} \phi^j w_{t-j},$$

where $\{w_t\}$ satisfies the conditions of Theorem A.7 and $|\phi| < 1$, show that

$$\sqrt{n} \frac{(\hat{\rho}_x(1) - \rho_x(1))}{\sqrt{1 - \rho_x^2(1)}} \xrightarrow{d} N(0, 1),$$

and construct a 95% confidence interval for ϕ when $\hat{\rho}_x(1) = .64$ and $n = 100$.

1.32 Let $\{x_t; t = 0, \pm 1, \pm 2, \dots\}$ be iid($0, \sigma^2$).

- (a) For $h \geq 1$ and $k \geq 1$, show that $x_t x_{t+h}$ and $x_s x_{s+k}$ are uncorrelated for all $s \neq t$.
- (b) For fixed $h \geq 1$, show that the $h \times 1$ vector

$$\sigma^{-2} n^{-1/2} \sum_{t=1}^n (x_t x_{t+1}, \dots, x_t x_{t+h})' \xrightarrow{d} (z_1, \dots, z_h)'$$

where z_1, \dots, z_h are iid $N(0, 1)$ random variables. [Hint: Use the Cramér-Wold device.]

(c) Show, for each $h \geq 1$,

$$n^{-1/2} \left[\sum_{t=1}^n x_t x_{t+h} - \sum_{t=1}^{n-h} (x_t - \bar{x})(x_{t+h} - \bar{x}) \right] \xrightarrow{P} 0 \quad \text{as } n \rightarrow \infty$$

where $\bar{x} = n^{-1} \sum_{t=1}^n x_t$.

(d) Noting that $n^{-1} \sum_{t=1}^n x_t^2 \xrightarrow{P} \sigma^2$ by the WLLN, conclude that

$$n^{1/2} [\hat{\rho}(1), \dots, \hat{\rho}(h)]' \xrightarrow{d} (z_1, \dots, z_h)'$$

where $\hat{\rho}(h)$ is the sample ACF of the data x_1, \dots, x_n .

Chapter 2

Time Series Regression and Exploratory Data Analysis

In this chapter we introduce classical multiple linear regression in a time series context, model selection, exploratory data analysis for preprocessing nonstationary time series (for example trend removal), the concept of differencing and the backshift operator, variance stabilization, and nonparametric smoothing of time series.

2.1 Classical Regression in the Time Series Context

We begin our discussion of linear regression in the time series context by assuming some output or *dependent* time series, say, x_t , for $t = 1, \dots, n$, is being influenced by a collection of possible inputs or *independent* series, say, $z_{t1}, z_{t2}, \dots, z_{tq}$, where we first regard the inputs as fixed and known. This assumption, necessary for applying conventional linear regression, will be relaxed later on. We express this relation through the *linear regression model*

$$x_t = \beta_0 + \beta_1 z_{t1} + \beta_2 z_{t2} + \dots + \beta_q z_{tq} + w_t, \quad (2.1)$$

where $\beta_0, \beta_1, \dots, \beta_q$ are unknown fixed regression coefficients, and $\{w_t\}$ is a random error or noise process consisting of independent and identically distributed (iid) normal variables with mean zero and variance σ_w^2 . For time series regression, it is rarely the case that the noise is white, and we will need to eventually relax that assumption. A more general setting within which to embed mean square estimation and linear regression is given in [Appendix B](#), where we introduce Hilbert spaces and the Projection Theorem.

Example 2.1 Estimating a Linear Trend

Consider the monthly price (per pound) of a chicken in the US from mid-2001 to mid-2016 (180 months), say x_t , shown in [Figure 2.1](#). There is an obvious upward trend in the series, and we might use simple linear regression to estimate that trend by fitting the model

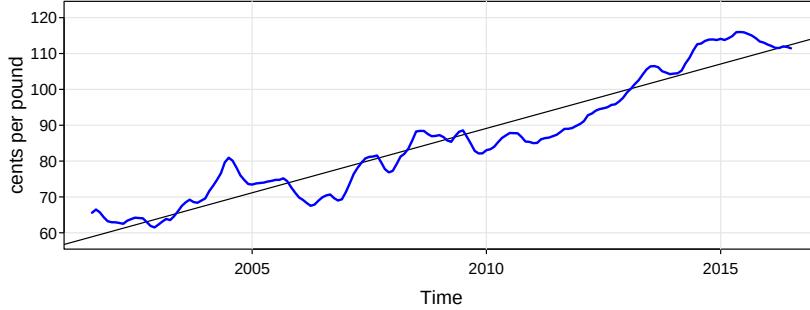


Fig. 2.1. The price of chicken: monthly whole bird spot price, Georgia docks, US cents per pound, August 2001 to July 2016, with fitted linear trend line.

$$x_t = \beta_0 + \beta_1 z_t + w_t, \quad z_t = 2001\frac{7}{12}, 2001\frac{8}{12}, \dots, 2016\frac{6}{12}.$$

This is in the form of the regression model (2.1) with $q = 1$. Note that we are making the assumption that the errors, w_t , are an iid normal sequence, which may not be true; the problem of autocorrelated errors is discussed in detail in [Chapter 3](#).

In ordinary least squares (OLS), we minimize the error sum of squares

$$Q = \sum_{t=1}^n w_t^2 = \sum_{t=1}^n (x_t - [\beta_0 + \beta_1 z_t])^2$$

with respect to β_i for $i = 0, 1$. In this case we can use simple calculus to evaluate $\partial Q / \partial \beta_i = 0$ for $i = 0, 1$, to obtain two equations to solve for the β s. The OLS estimates of the coefficients are explicit and given by

$$\hat{\beta}_1 = \frac{\sum_{t=1}^n (x_t - \bar{x})(z_t - \bar{z})}{\sum_{t=1}^n (z_t - \bar{z})^2} \quad \text{and} \quad \hat{\beta}_0 = \bar{x} - \hat{\beta}_1 \bar{z},$$

where $\bar{x} = \sum_t x_t / n$ and $\bar{z} = \sum_t z_t / n$ are the respective sample means.

Using R, we obtained the estimated slope coefficient of $\hat{\beta}_1 = 3.59$ (with a standard error of .08) yielding a significant estimated increase of about 3.6 cents per year. Finally, [Figure 2.1](#) shows the data with the estimated trend line superimposed. R code with partial output:

```
summary(fit <- lm(chicken~time(chicken), na.action=NULL))
      Estimate Std. Error t.value
(Intercept) -7131.02     162.41   -43.9
time(chicken)    3.59      0.08    44.4
---
Residual standard error: 4.7 on 178 degrees of freedom
plot(chicken, ylab="cents per pound")
abline(fit)      # add the fitted line
```

The multiple linear regression model described by (2.1) can be conveniently written in a more general notation by defining the column vectors $\mathbf{z}_t = (1, z_{t1}, z_{t2}, \dots, z_{tq})'$

and $\beta = (\beta_0, \beta_1, \dots, \beta_q)'$, where ' denotes transpose, so (2.1) can be written in the alternate form

$$x_t = \beta_0 + \beta_1 z_{t1} + \dots + \beta_q z_{tq} + w_t = \beta' z_t + w_t. \quad (2.2)$$

where $w_t \sim \text{iid } N(0, \sigma_w^2)$. As in the previous example, OLS estimation finds the coefficient vector β that minimizes the error sum of squares

$$Q = \sum_{t=1}^n w_t^2 = \sum_{t=1}^n (x_t - \beta' z_t)^2, \quad (2.3)$$

with respect to $\beta_0, \beta_1, \dots, \beta_q$. This minimization can be accomplished by differentiating (2.3) with respect to the vector β or by using the properties of projections. Either way, the solution must satisfy $\sum_{t=1}^n (x_t - \hat{\beta}' z_t) z_t' = 0$. This procedure gives the *normal equations*

$$\left(\sum_{t=1}^n z_t z_t' \right) \hat{\beta} = \sum_{t=1}^n z_t x_t. \quad (2.4)$$

If $\sum_{t=1}^n z_t z_t'$ is non-singular, the least squares estimate of β is

$$\hat{\beta} = \left(\sum_{t=1}^n z_t z_t' \right)^{-1} \sum_{t=1}^n z_t x_t.$$

The minimized error sum of squares (2.3), denoted *SSE*, can be written as

$$SSE = \sum_{t=1}^n (x_t - \hat{\beta}' z_t)^2. \quad (2.5)$$

The ordinary least squares estimators are unbiased, i.e., $E(\hat{\beta}) = \beta$, and have the smallest variance within the class of linear unbiased estimators.

If the errors w_t are normally distributed, $\hat{\beta}$ is also the maximum likelihood estimator for β and is normally distributed with

$$\text{cov}(\hat{\beta}) = \sigma_w^2 C, \quad (2.6)$$

where

$$C = \left(\sum_{t=1}^n z_t z_t' \right)^{-1} \quad (2.7)$$

is a convenient notation. An unbiased estimator for the variance σ_w^2 is

$$s_w^2 = MSE = \frac{SSE}{n - (q + 1)}, \quad (2.8)$$

where *MSE* denotes the *mean squared error*. Under the normal assumption,

$$t = \frac{(\hat{\beta}_i - \beta_i)}{s_w \sqrt{c_{ii}}} \quad (2.9)$$

Table 2.1. Analysis of Variance for Regression

Source	df	Sum of Squares	Mean Square	F
$z_{t,r+1:q}$	$q - r$	$SSR = SSE_r - SSE$	$MSR = SSR/(q - r)$	$F = \frac{MSR}{MSE}$
Error	$n - (q + 1)$	SSE	$MSE = SSE/(n - q - 1)$	

has the t-distribution with $n - (q + 1)$ degrees of freedom; c_{ii} denotes the i -th diagonal element of C , as defined in (2.7). This result is often used for individual tests of the null hypothesis $H_0: \beta_i = 0$ for $i = 1, \dots, q$.

Various competing models are often of interest to isolate or select the best subset of independent variables. Suppose a proposed model specifies that only a subset $r < q$ independent variables, say, $z_{t,1:r} = \{z_{t1}, z_{t2}, \dots, z_{tr}\}$ is influencing the dependent variable x_t . The reduced model is

$$x_t = \beta_0 + \beta_1 z_{t1} + \dots + \beta_r z_{tr} + w_t \quad (2.10)$$

where $\beta_1, \beta_2, \dots, \beta_r$ are a subset of coefficients of the original q variables.

The null hypothesis in this case is $H_0: \beta_{r+1} = \dots = \beta_q = 0$. We can test the reduced model (2.10) against the full model (2.2) by comparing the error sums of squares under the two models using the F-statistic

$$F = \frac{(SSE_r - SSE)/(q - r)}{SSE/(n - q - 1)} = \frac{MSR}{MSE}, \quad (2.11)$$

where SSE_r is the error sum of squares under the reduced model (2.10). Note that $SSE_r \geq SSE$ because the full model has more parameters. If $H_0: \beta_{r+1} = \dots = \beta_q = 0$ is true, then $SSE_r \approx SSE$ because the estimates of those β s will be close to 0. Hence, we do not believe H_0 if $SSR = SSE_r - SSE$ is big. Under the null hypothesis, (2.11) has a central F-distribution with $q - r$ and $n - q - 1$ degrees of freedom when (2.10) is the correct model.

These results are often summarized in an *Analysis of Variance (ANOVA)* table as given in Table 2.1 for this particular case. The difference in the numerator is often called the regression sum of squares (SSR). The null hypothesis is rejected at level α if $F > F_{n-q-1}^{q-r}(\alpha)$, the $1 - \alpha$ percentile of the F distribution with $q - r$ numerator and $n - q - 1$ denominator degrees of freedom.

A special case of interest is the null hypothesis $H_0: \beta_1 = \dots = \beta_q = 0$. In this case $r = 0$, and the model in (2.10) becomes

$$x_t = \beta_0 + w_t.$$

We may measure the proportion of variation accounted for by all the variables using

$$R^2 = \frac{SSE_0 - SSE}{SSE_0}, \quad (2.12)$$

where the residual sum of squares under the reduced model is

$$SSE_0 = \sum_{t=1}^n (x_t - \bar{x})^2. \quad (2.13)$$

In this case SSE_0 is the sum of squared deviations from the mean \bar{x} and is otherwise known as the adjusted total sum of squares. The measure R^2 is called the *coefficient of determination*.

The techniques discussed in the previous paragraph can be used to test various models against one another using the F test given in (2.11). These tests have been used in the past in a stepwise manner, where variables are added or deleted when the values from the F -test either exceed or fail to exceed some predetermined levels. The procedure, called *stepwise multiple regression*, is useful in arriving at a set of useful variables. An alternative is to focus on a procedure for *model selection* that does not proceed sequentially, but simply evaluates each model on its own merits. Suppose we consider a normal regression model with k coefficients and denote the *maximum likelihood estimator* for the variance as

$$\hat{\sigma}_k^2 = \frac{SSE(k)}{n}, \quad (2.14)$$

where $SSE(k)$ denotes the residual sum of squares under the model with k regression coefficients. Then, Akaike (1969, 1973, 1974) suggested measuring the goodness of fit for this particular model by balancing the error of the fit against the number of parameters in the model; we define the following.^{2.1}

Definition 2.1 Akaike's Information Criterion (AIC)

$$AIC = \log \hat{\sigma}_k^2 + \frac{n + 2k}{n}, \quad (2.15)$$

where $\hat{\sigma}_k^2$ is given by (2.14) and k is the number of parameters in the model.

The value of k yielding the minimum AIC specifies the best model. The idea is roughly that minimizing $\hat{\sigma}_k^2$ would be a reasonable objective, except that it decreases monotonically as k increases. Therefore, we ought to penalize the error variance by a term proportional to the number of parameters. The choice for the penalty term given by (2.15) is not the only one, and a considerable literature is available advocating different penalty terms. A corrected form, suggested by Sugiura (1978), and expanded by Hurvich and Tsai (1989), can be based on small-sample distributional results for the linear regression model (details are provided in [Problem 2.4](#) and [Problem 2.5](#)). The corrected form is defined as follows.

Definition 2.2 AIC, Bias Corrected (AICc)

$$AICc = \log \hat{\sigma}_k^2 + \frac{n + k}{n - k - 2}, \quad (2.16)$$

^{2.1} Formally, AIC is defined as $-2 \log L_k + 2k$ where L_k is the maximized likelihood and k is the number of parameters in the model. For the normal regression problem, AIC can be reduced to the form given by (2.15). AIC is an estimate of the Kullback-Leibler discrepancy between a true model and a candidate model; see [Problem 2.4](#) and [Problem 2.5](#) for further details.

where $\hat{\sigma}_k^2$ is given by (2.14), k is the number of parameters in the model, and n is the sample size.

We may also derive a correction term based on Bayesian arguments, as in Schwarz (1978), which leads to the following.

Definition 2.3 Bayesian Information Criterion (BIC)

$$\text{BIC} = \log \hat{\sigma}_k^2 + \frac{k \log n}{n}, \quad (2.17)$$

using the same notation as in [Definition 2.2](#).

BIC is also called the Schwarz Information Criterion (SIC); see also Rissanen (1978) for an approach yielding the same statistic based on a minimum description length argument. Notice that the penalty term in BIC is much larger than in AIC, consequently, BIC tends to choose smaller models. Various simulation studies have tended to verify that BIC does well at getting the correct order in large samples, whereas AICc tends to be superior in smaller samples where the relative number of parameters is large; see McQuarrie and Tsai (1998) for detailed comparisons. In fitting regression models, two measures that have been used in the past are adjusted R-squared, which is essentially s_w^2 , and Mallows C_p , Mallows (1973), which we do not consider in this context.

Example 2.2 Pollution, Temperature and Mortality

The data shown in [Figure 2.2](#) are extracted series from a study by Shumway et al. (1988) of the possible effects of temperature and pollution on weekly mortality in Los Angeles County. Note the strong seasonal components in all of the series, corresponding to winter-summer variations and the downward trend in the cardiovascular mortality over the 10-year period.

A scatterplot matrix, shown in [Figure 2.3](#), indicates a possible linear relation between mortality and the pollutant particulates and a possible relation to temperature. Note the curvilinear shape of the temperature mortality curve, indicating that higher temperatures as well as lower temperatures are associated with increases in cardiovascular mortality.

Based on the scatterplot matrix, we entertain, tentatively, four models where M_t denotes cardiovascular mortality, T_t denotes temperature and P_t denotes the particulate levels. They are

$$M_t = \beta_0 + \beta_1 t + w_t \quad (2.18)$$

$$M_t = \beta_0 + \beta_1 t + \beta_2(T_t - T.) + w_t \quad (2.19)$$

$$M_t = \beta_0 + \beta_1 t + \beta_2(T_t - T.) + \beta_3(T_t - T.)^2 + w_t \quad (2.20)$$

$$M_t = \beta_0 + \beta_1 t + \beta_2(T_t - T.) + \beta_3(T_t - T.)^2 + \beta_4 P_t + w_t \quad (2.21)$$

where we adjust temperature for its mean, $T. = 74.26$, to avoid collinearity problems. It is clear that (2.18) is a trend only model, (2.19) is linear temperature, (2.20)

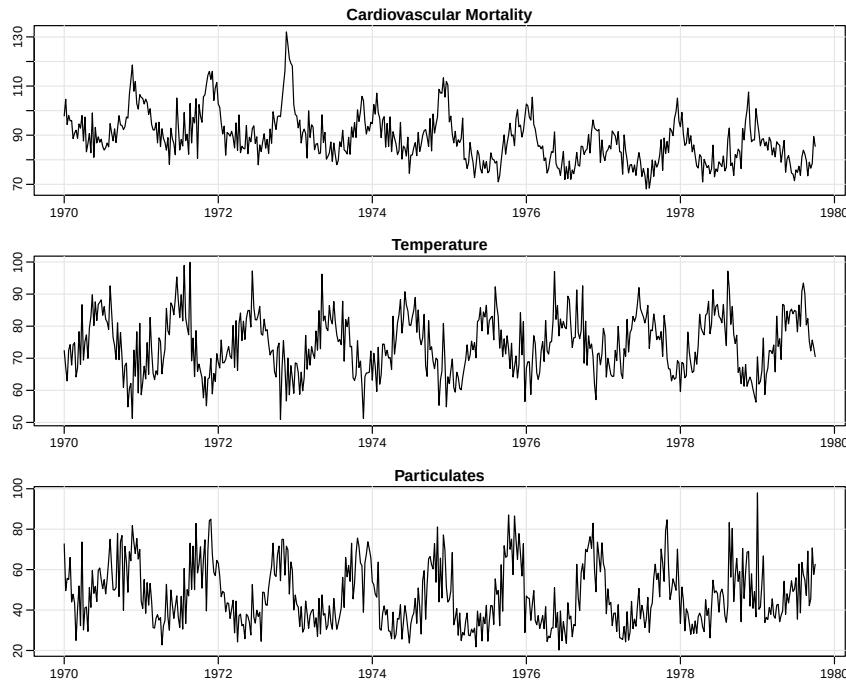


Fig. 2.2. Average weekly cardiovascular mortality (top), temperature (middle) and particulate pollution (bottom) in Los Angeles County. There are 508 six-day smoothed averages obtained by filtering daily values over the 10 year period 1970-1979.

Table 2.2. Summary Statistics for Mortality Models

Model	<i>k</i>	SSE	df	MSE	<i>R</i> ²	AIC	BIC
(2.18)	2	40,020	506	79.0	.21	5.38	5.40
(2.19)	3	31,413	505	62.2	.38	5.14	5.17
(2.20)	4	27,985	504	55.5	.45	5.03	5.07
(2.21)	5	20,508	503	40.8	.60	4.72	4.77

is curvilinear temperature and (2.21) is curvilinear temperature and pollution. We summarize some of the statistics given for this particular case in Table 2.2.

We note that each model does substantially better than the one before it and that the model including temperature, temperature squared, and particulates does the best, accounting for some 60% of the variability and with the best value for AIC and BIC (because of the large sample size, AIC and AICc are nearly the same). Note that one can compare any two models using the residual sums of squares and (2.11). Hence, a model with only trend could be compared to the full model, $H_0: \beta_2 = \beta_3 = \beta_4 = 0$, using $q = 4, r = 1, n = 508$, and

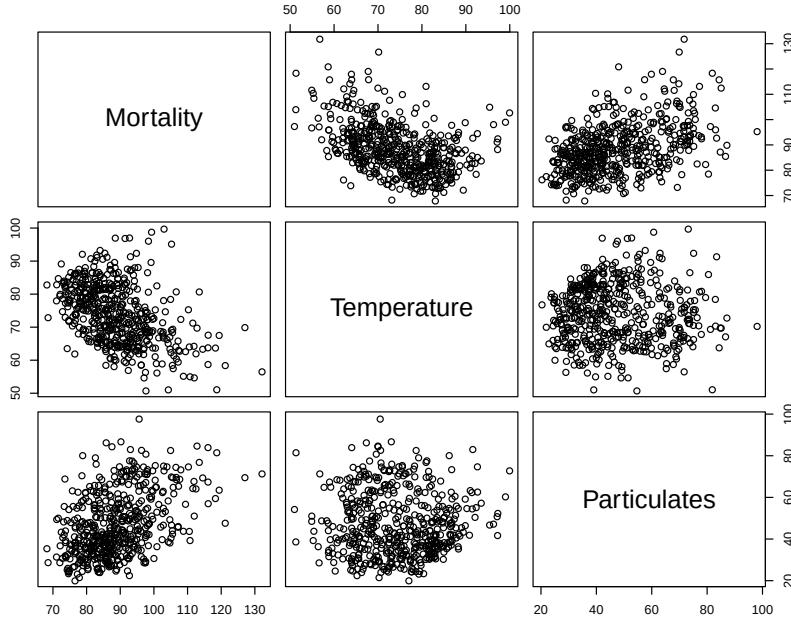


Fig. 2.3. Scatterplot matrix showing relations between mortality, temperature, and pollution.

$$F_{3,503} = \frac{(40,020 - 20,508)/3}{20,508/503} = 160,$$

which exceeds $F_{3,503}(.001) = 5.51$. We obtain the best prediction model,

$$\begin{aligned}\hat{M}_t = & 2831.5 - 1.396_{(.10)} t - .472_{(.032)}(T_t - 74.26) \\ & + .023_{(.003)}(T_t - 74.26)^2 + .255_{(.019)} P_t,\end{aligned}$$

for mortality, where the standard errors, computed from (2.6)–(2.8), are given in parentheses. As expected, a negative trend is present in time as well as a negative coefficient for adjusted temperature. The quadratic effect of temperature can clearly be seen in the scatterplots of Figure 2.3. Pollution weights positively and can be interpreted as the incremental contribution to daily deaths per unit of particulate pollution. It would still be essential to check the residuals $\hat{w}_t = M_t - \hat{M}_t$ for autocorrelation (of which there is a substantial amount), but we defer this question to Section 3.8 when we discuss regression with correlated errors.

Below is the R code to plot the series, display the scatterplot matrix, fit the final regression model (2.21), and compute the corresponding values of AIC, AICc and BIC.^{2.2} Finally, the use of `na.action` in `lm()` is to retain the time series attributes for the residuals and fitted values.

^{2.2} The easiest way to extract AIC and BIC from an `lm()` run in R is to use the command `AIC()` or `BIC()`. Our definitions differ from R by terms that do not change from model to model. In the example, we show how to obtain (2.15) and (2.17) from the R output. It is more difficult to obtain AICc.

```

par(mfrow=c(3,1)) # plot the data
plot(cmort, main="Cardiovascular Mortality", xlab="", ylab="")
plot(temp, main="Temperature", xlab="", ylab="")
plot(part, main="Particulates", xlab="", ylab="")
dev.new() # open a new graphic device
ts.plot(cmort,temp,part, col=1:3) # all on same plot (not shown)
dev.new()
pairs(cbind(Mortality=cmort, Temperature=temp, Particulates=part))
temp = temp-mean(temp) # center temperature
temp2 = temp^2
trend = time(cmort) # time
fit = lm(cmort~ trend + temp + temp2 + part, na.action=NULL)
summary(fit) # regression results
summary(aov(fit)) # ANOVA table (compare to next line)
summary(aov(lm(cmort~cbind(trend, temp, temp2, part)))) # Table 2.1
num = length(cmort) # sample size
AIC(fit)/num - log(2*pi) # AIC
BIC(fit)/num - log(2*pi) # BIC
(AICc = log(sum(resid(fit)^2)/num) + (num+5)/(num-5-2)) # AICc

```

As previously mentioned, it is possible to include lagged variables in time series regression models and we will continue to discuss this type of problem throughout the text. This concept is explored further in [Problem 2.2](#) and [Problem 2.10](#). The following is a simple example of lagged regression.

Example 2.3 Regression With Lagged Variables

In [Example 1.28](#), we discovered that the Southern Oscillation Index (SOI) measured at time $t - 6$ months is associated with the Recruitment series at time t , indicating that the SOI leads the Recruitment series by six months. Although there is evidence that the relationship is not linear (this is discussed further in [Example 2.8](#) and [Example 2.9](#)), consider the following regression,

$$R_t = \beta_0 + \beta_1 S_{t-6} + w_t, \quad (2.22)$$

where R_t denotes Recruitment for month t and S_{t-6} denotes SOI six months prior. Assuming the w_t sequence is white, the fitted model is

$$\hat{R}_t = 65.79 - 44.28_{(2.78)} S_{t-6} \quad (2.23)$$

with $\hat{\sigma}_w = 22.5$ on 445 degrees of freedom. This result indicates the strong predictive ability of SOI for Recruitment six months in advance. Of course, it is still essential to check the model assumptions, but again we defer this until later.

Performing lagged regression in R is a little difficult because the series must be aligned prior to running the regression. The easiest way to do this is to create a data frame (that we call `fish`) using `ts.intersect`, which aligns the lagged series.

```

fish = ts.intersect(rec, soiL6=lag(soi, -6), dframe=TRUE)
summary(fit1 <- lm(rec~soiL6, data=fish, na.action=NULL))

```

The headache of aligning the lagged series can be avoided by using the R package `dynlm`, which must be downloaded and installed.

```

library(dynlm)
summary(fit2 <- dynlm(rec~ L(soi, 6)))

```

We note that `fit2` is similar to the `fit1` object, but the time series attributes are retained without any additional commands.

2.2 Exploratory Data Analysis

In general, it is necessary for time series data to be stationary so that averaging lagged products over time, as in the previous section, will be a sensible thing to do. With time series data, it is the dependence between the values of the series that is important to measure; we must, at least, be able to estimate autocorrelations with precision. It would be difficult to measure that dependence if the dependence structure is not regular or is changing at every time point. Hence, to achieve any meaningful statistical analysis of time series data, it will be crucial that, if nothing else, the mean and the autocovariance functions satisfy the conditions of stationarity (for at least some reasonable stretch of time) stated in [Definition 1.7](#). Often, this is not the case, and we will mention some methods in this section for playing down the effects of nonstationarity so the stationary properties of the series may be studied.

A number of our examples came from clearly nonstationary series. The Johnson & Johnson series in [Figure 1.1](#) has a mean that increases exponentially over time, and the increase in the magnitude of the fluctuations around this trend causes changes in the covariance function; the variance of the process, for example, clearly increases as one progresses over the length of the series. Also, the global temperature series shown in [Figure 1.2](#) contains some evidence of a trend over time; human-induced global warming advocates seize on this as empirical evidence to advance the hypothesis that temperatures are increasing.

Perhaps the easiest form of nonstationarity to work with is the *trend stationary* model wherein the process has stationary behavior around a trend. We may write this type of model as

$$x_t = \mu_t + y_t \quad (2.24)$$

where x_t are the observations, μ_t denotes the trend, and y_t is a stationary process. Quite often, strong trend will obscure the behavior of the stationary process, y_t , as we shall see in numerous examples. Hence, there is some advantage to removing the trend as a first step in an exploratory analysis of such time series. The steps involved are to obtain a reasonable estimate of the trend component, say $\hat{\mu}_t$, and then work with the residuals

$$\hat{y}_t = x_t - \hat{\mu}_t. \quad (2.25)$$

Example 2.4 Detrending Chicken Prices

Here we suppose the model is of the form of (2.24),

$$x_t = \mu_t + y_t,$$

where, as we suggested in the analysis of the chicken price data presented in [Example 2.1](#), a straight line might be useful for detrending the data; i.e.,

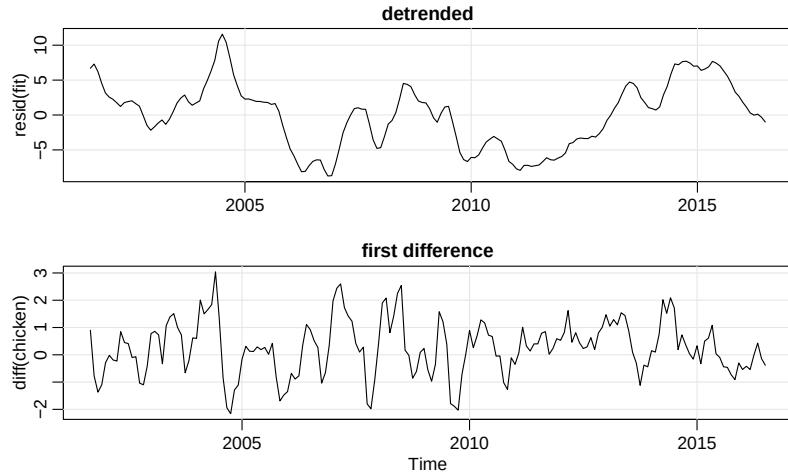


Fig. 2.4. Detrended (top) and differenced (bottom) chicken price series. The original data are shown in [Figure 2.1](#).

$$\mu_t = \beta_0 + \beta_1 t.$$

In that example, we estimated the trend using ordinary least squares and found

$$\hat{\mu}_t = -7131 + 3.59 t$$

where we are using t instead of z_t for time. [Figure 2.1](#) shows the data with the estimated trend line superimposed. To obtain the detrended series we simply subtract $\hat{\mu}_t$ from the observations, x_t , to obtain the detrended series^{2.3}

$$\hat{y}_t = x_t + 7131 - 3.59 t.$$

The top graph of [Figure 2.4](#) shows the detrended series. [Figure 2.5](#) shows the ACF of the original data (top panel) as well as the ACF of the detrended data (middle panel).

In [Example 1.11](#) and the corresponding [Figure 1.10](#) we saw that a random walk might also be a good model for trend. That is, rather than modeling trend as fixed (as in [Example 2.4](#)), we might model trend as a stochastic component using the random walk with drift model,

$$\mu_t = \delta + \mu_{t-1} + w_t, \tag{2.26}$$

where w_t is white noise and is independent of y_t . If the appropriate model is (2.24), then differencing the data, x_t , yields a stationary process; that is,

^{2.3} Because the error term, y_t , is not assumed to be iid, the reader may feel that weighted least squares is called for in this case. The problem is, we do not know the behavior of y_t and that is precisely what we are trying to assess at this stage. A notable result by Grenander and Rosenblatt (1957, Ch 7), however, is that under mild conditions on y_t , for polynomial regression or periodic regression, asymptotically, ordinary least squares is equivalent to weighted least squares with regard to efficiency.

$$\begin{aligned}x_t - x_{t-1} &= (\mu_t + y_t) - (\mu_{t-1} + y_{t-1}) \\&= \delta + w_t + y_t - y_{t-1}.\end{aligned}\tag{2.27}$$

It is easy to show $z_t = y_t - y_{t-1}$ is stationary using [Property 1.1](#). That is, because y_t is stationary,

$$\begin{aligned}\gamma_z(h) &= \text{cov}(z_{t+h}, z_t) = \text{cov}(y_{t+h} - y_{t+h-1}, y_t - y_{t-1}) \\&= 2\gamma_y(h) - \gamma_y(h+1) - \gamma_y(h-1)\end{aligned}$$

is independent of time; we leave it as an exercise ([Problem 2.7](#)) to show that $x_t - x_{t-1}$ in (2.27) is stationary.

One advantage of differencing over detrending to remove trend is that no parameters are estimated in the differencing operation. One disadvantage, however, is that differencing does not yield an estimate of the stationary process y_t as can be seen in (2.27). If an estimate of y_t is essential, then detrending may be more appropriate. If the goal is to coerce the data to stationarity, then differencing may be more appropriate. Differencing is also a viable tool if the trend is fixed, as in [Example 2.4](#). That is, e.g., if $\mu_t = \beta_0 + \beta_1 t$ in the model (2.24), differencing the data produces stationarity (see [Problem 2.6](#)):

$$x_t - x_{t-1} = (\mu_t + y_t) - (\mu_{t-1} + y_{t-1}) = \beta_1 + y_t - y_{t-1}.$$

Because differencing plays a central role in time series analysis, it receives its own notation. The first difference is denoted as

$$\nabla x_t = x_t - x_{t-1}.\tag{2.28}$$

As we have seen, the first difference eliminates a linear trend. A second difference, that is, the difference of (2.28), can eliminate a quadratic trend, and so on. In order to define higher differences, we need a variation in notation that we will use often in our discussion of ARIMA models in [Chapter 3](#).

Definition 2.4 We define the **backshift operator** by

$$Bx_t = x_{t-1}$$

and extend it to powers $B^2 x_t = B(Bx_t) = Bx_{t-1} = x_{t-2}$, and so on. Thus,

$$B^k x_t = x_{t-k}.\tag{2.29}$$

The idea of an inverse operator can also be given if we require $B^{-1}B = 1$, so that

$$x_t = B^{-1}Bx_t = B^{-1}x_{t-1}.$$

That is, B^{-1} is the *forward-shift operator*. In addition, it is clear that we may rewrite (2.28) as

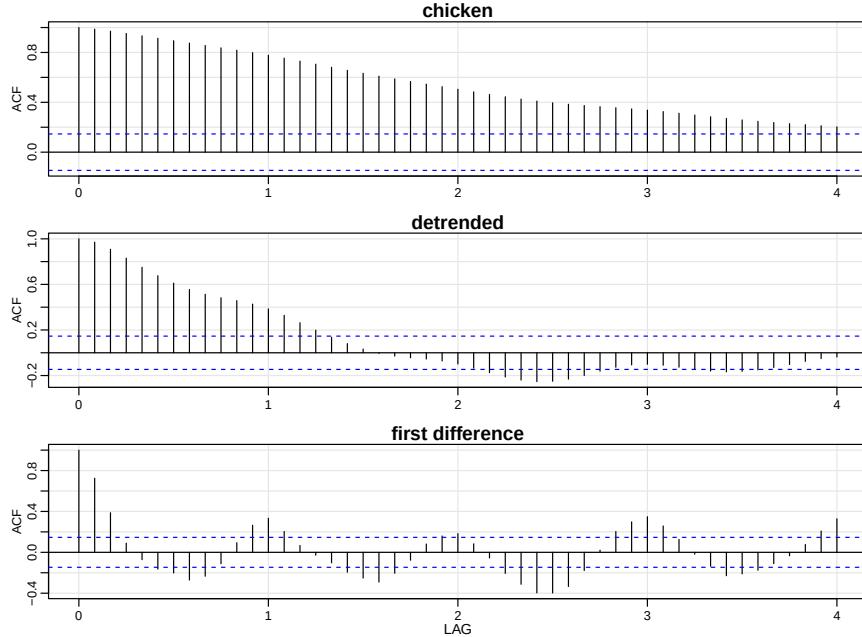


Fig. 2.5. Sample ACFs of chicken prices (top), and of the detrended (middle) and the differenced (bottom) series. Compare the top plot with the sample ACF of a straight line: `acf(1:100)`.

$$\nabla x_t = (1 - B)x_t, \quad (2.30)$$

and we may extend the notion further. For example, the second difference becomes

$$\nabla^2 x_t = (1 - B)^2 x_t = (1 - 2B + B^2)x_t = x_t - 2x_{t-1} + x_{t-2} \quad (2.31)$$

by the linearity of the operator. To check, just take the difference of the first difference $\nabla(\nabla x_t) = \nabla(x_t - x_{t-1}) = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2})$.

Definition 2.5 Differences of order d are defined as

$$\nabla^d = (1 - B)^d, \quad (2.32)$$

where we may expand the operator $(1 - B)^d$ algebraically to evaluate for higher integer values of d . When $d = 1$, we drop it from the notation.

The first difference (2.28) is an example of a *linear filter* applied to eliminate a trend. Other filters, formed by averaging values near x_t , can produce adjusted series that eliminate other kinds of unwanted fluctuations, as in Chapter 4. The differencing technique is an important component of the ARIMA model of Box and Jenkins (1970) (see also Box et al., 1994), to be discussed in Chapter 3.

Example 2.5 Differencing Chicken Prices

The first difference of the chicken prices series, also shown in Figure 2.4, produces different results than removing trend by detrending via regression. For example, the differenced series does not contain the long (five-year) cycle we observe in the detrended series. The ACF of this series is also shown in Figure 2.5. In this case, the differenced series exhibits an annual cycle that was obscured in the original or detrended data.

The R code to reproduce Figure 2.4 and Figure 2.5 is as follows.

```
fit = lm(chicken~time(chicken), na.action=NULL) # regress chicken on time
par(mfrow=c(2,1))
plot(resid(fit), type="o", main="detrended")
plot(diff(chicken), type="o", main="first difference")
par(mfrow=c(3,1)) # plot ACFs
acf(chicken, 48, main="chicken")
acf(resid(fit), 48, main="detrended")
acf(diff(chicken), 48, main="first difference")
```

Example 2.6 Differencing Global Temperature

The global temperature series shown in Figure 1.2 appears to behave more as a random walk than a trend stationary series. Hence, rather than detrend the data, it would be more appropriate to use differencing to coerce it into stationarity. The detreded data are shown in Figure 2.6 along with the corresponding sample ACF. In this case it appears that the differenced process shows minimal autocorrelation, which may imply the global temperature series is nearly a random walk with drift. It is interesting to note that if the series is a random walk with drift, the mean of the differenced series, which is an estimate of the drift, is about .008, or an increase of about one degree centigrade per 100 years.

The R code to reproduce Figure 2.4 and Figure 2.5 is as follows.

```
par(mfrow=c(2,1))
plot(diff(globtemp), type="o")
mean(diff(globtemp)) # drift estimate = .008
acf(diff(gtemp), 48)
```

An alternative to differencing is a less-severe operation that still assumes stationarity of the underlying time series. This alternative, called *fractional differencing*, extends the notion of the difference operator (2.32) to fractional powers $-.5 < d < .5$, which still define stationary processes. Granger and Joyeux (1980) and Hosking (1981) introduced long memory time series, which corresponds to the case when $0 < d < .5$. This model is often used for environmental time series arising in hydrology. We will discuss long memory processes in more detail in Section 5.1.

Often, obvious aberrations are present that can contribute nonstationary as well as nonlinear behavior in observed time series. In such cases, *transformations* may be useful to equalize the variability over the length of a single series. A particularly useful transformation is

$$y_t = \log x_t, \tag{2.33}$$

which tends to suppress larger fluctuations that occur over portions of the series where the underlying values are larger. Other possibilities are *power transformations* in the

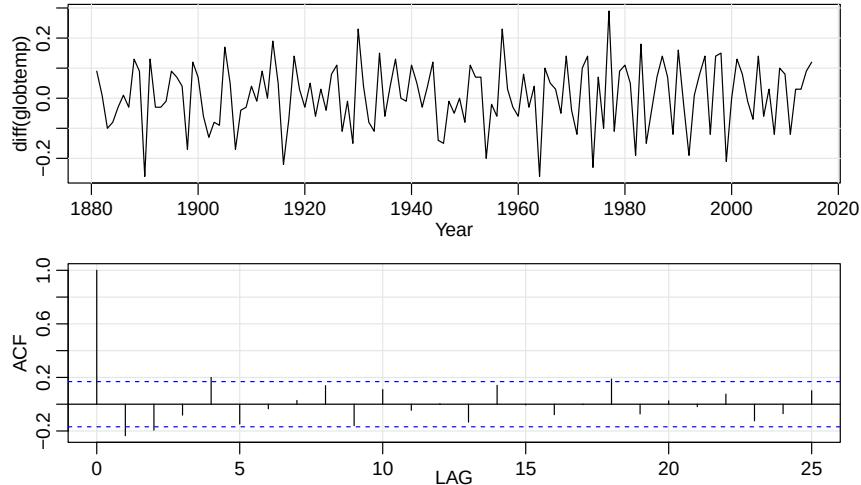


Fig. 2.6. Differenced global temperature series and its sample ACF.

Box–Cox family of the form

$$y_t = \begin{cases} (x_t^\lambda - 1)/\lambda & \lambda \neq 0, \\ \log x_t & \lambda = 0. \end{cases} \quad (2.34)$$

Methods for choosing the power λ are available (see Johnson and Wichern, 1992, §4.7) but we do not pursue them here. Often, transformations are also used to improve the approximation to normality or to improve linearity in predicting the value of one series from another.

Example 2.7 Paleoclimatic Glacial Varves

Melting glaciers deposit yearly layers of sand and silt during the spring melting seasons, which can be reconstructed yearly over a period ranging from the time deglaciation began in New England (about 12,600 years ago) to the time it ended (about 6,000 years ago). Such sedimentary deposits, called *varves*, can be used as proxies for paleoclimatic parameters, such as temperature, because, in a warm year, more sand and silt are deposited from the receding glacier. Figure 2.7 shows the thicknesses of the yearly varves collected from one location in Massachusetts for 634 years, beginning 11,834 years ago. For further information, see Shumway and Verosub (1992). Because the variation in thicknesses increases in proportion to the amount deposited, a logarithmic transformation could remove the nonstationarity observable in the variance as a function of time. Figure 2.7 shows the original and transformed varves, and it is clear that this improvement has occurred. We may also plot the histogram of the original and transformed data, as in Problem 2.8, to argue that the approximation to normality is improved. The ordinary first differences (2.30) are also computed in Problem 2.8, and we note that the first differences have

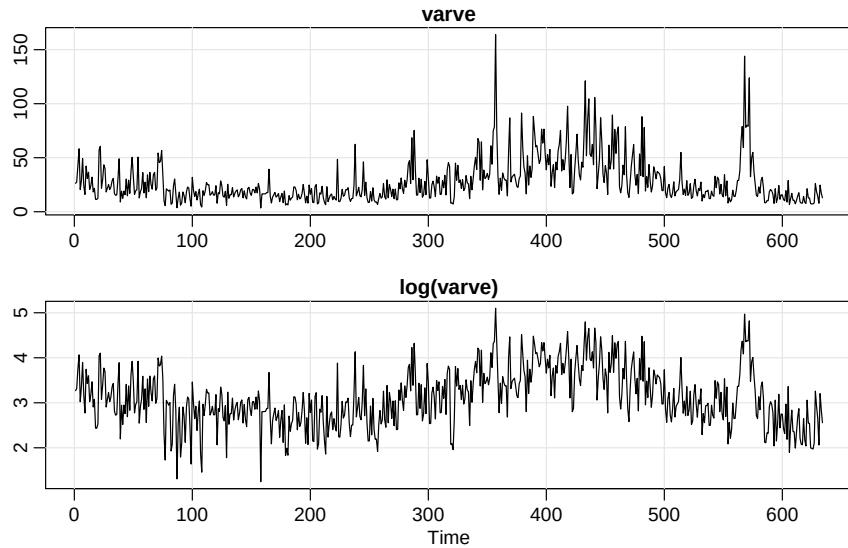


Fig. 2.7. Glacial varve thicknesses (top) from Massachusetts for $n = 634$ years compared with log transformed thicknesses (bottom).

a significant negative correlation at lag $h = 1$. Later, in [Chapter 5](#), we will show that perhaps the varve series has long memory and will propose using fractional differencing. [Figure 2.7](#) was generated in R as follows:

```
par(mfrow=c(2, 1))
plot(varve, main="varve", ylab="")
plot(log(varve), main="log(varve)", ylab="" )
```

Next, we consider another preliminary data processing technique that is used for the purpose of visualizing the relations between series at different lags, namely, *scatterplot matrices*. In the definition of the ACF, we are essentially interested in relations between x_t and x_{t-h} ; the autocorrelation function tells us whether a substantial linear relation exists between the series and its own lagged values. The ACF gives a profile of the linear correlation at all possible lags and shows which values of h lead to the best predictability. The restriction of this idea to linear predictability, however, may mask a possible nonlinear relation between current values, x_t , and past values, x_{t-h} . This idea extends to two series where one may be interested in examining scatterplots of y_t versus x_{t-h} .

Example 2.8 Scatterplot Matrices, SOI and Recruitment

To check for nonlinear relations of this form, it is convenient to display a lagged scatterplot matrix, as in [Figure 2.8](#), that displays values of the SOI, S_t , on the vertical axis plotted against S_{t-h} on the horizontal axis. The sample autocorrelations are displayed in the upper right-hand corner and superimposed on the scatterplots are locally weighted scatterplot smoothing (lowess) lines that can be used to help

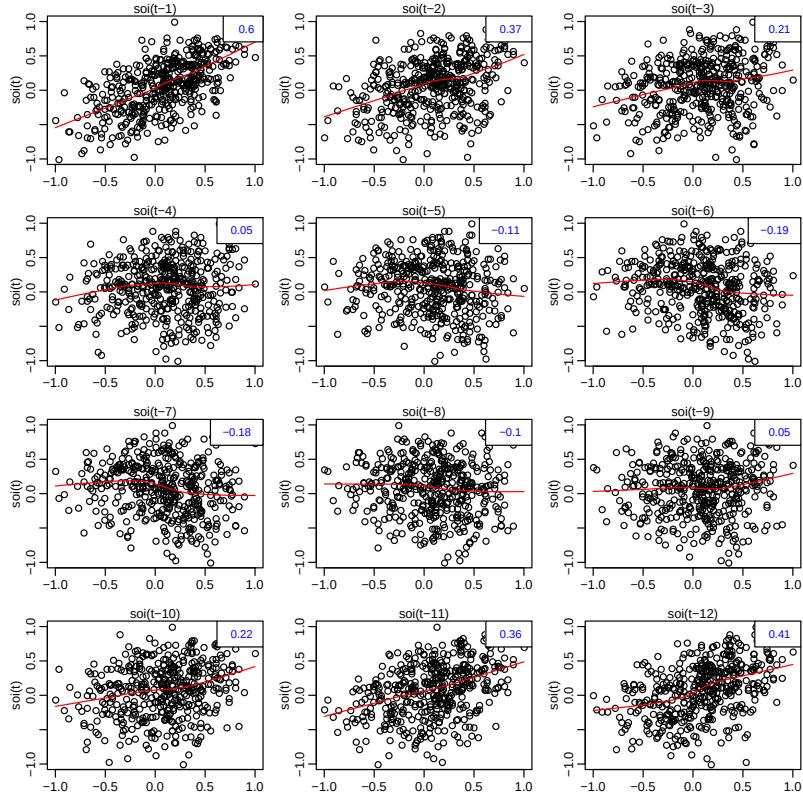


Fig. 2.8. Scatterplot matrix relating current SOI values, S_t , to past SOI values, S_{t-h} , at lags $h = 1, 2, \dots, 12$. The values in the upper right corner are the sample autocorrelations and the lines are a lowess fit.

discover any nonlinearities. We discuss smoothing in the next section, but for now, think of lowess as a robust method for fitting local regression.

In Figure 2.8, we notice that the lowess fits are approximately linear, so that the sample autocorrelations are meaningful. Also, we see strong positive linear relations at lags $h = 1, 2, 11, 12$, that is, between S_t and $S_{t-1}, S_{t-2}, S_{t-11}, S_{t-12}$, and a negative linear relation at lags $h = 6, 7$. These results match up well with peaks noticed in the ACF in Figure 1.16.

Similarly, we might want to look at values of one series, say Recruitment, denoted R_t plotted against another series at various lags, say the SOI, S_{t-h} , to look for possible nonlinear relations between the two series. Because, for example, we might wish to predict the Recruitment series, R_t , from current or past values of the SOI series, S_{t-h} , for $h = 0, 1, 2, \dots$ it would be worthwhile to examine the scatterplot matrix. Figure 2.9 shows the lagged scatterplot of the Recruitment series R_t on the

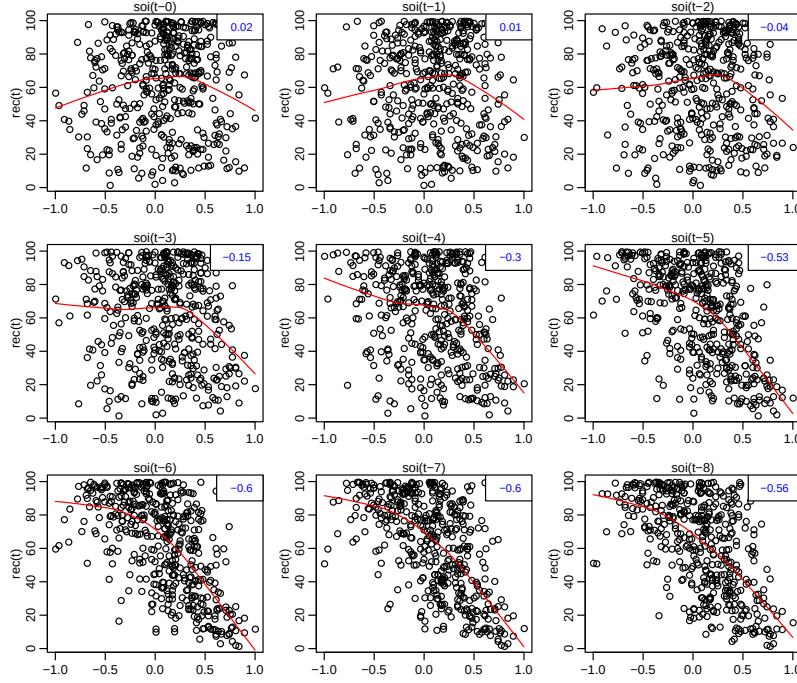


Fig. 2.9. Scatterplot matrix of the Recruitment series, R_t , on the vertical axis plotted against the SOI series, S_{t-h} , on the horizontal axis at lags $h = 0, 1, \dots, 8$. The values in the upper right corner are the sample cross-correlations and the lines are a lowess fit.

vertical axis plotted against the SOI index S_{t-h} on the horizontal axis. In addition, the figure exhibits the sample cross-correlations as well as lowess fits.

Figure 2.9 shows a fairly strong nonlinear relationship between Recruitment, R_t , and the SOI series at $S_{t-5}, S_{t-6}, S_{t-7}, S_{t-8}$, indicating the SOI series tends to lead the Recruitment series and the coefficients are negative, implying that increases in the SOI lead to decreases in the Recruitment. The nonlinearity observed in the scatterplots (with the help of the superimposed lowess fits) indicates that the behavior between Recruitment and the SOI is different for positive values of SOI than for negative values of SOI.

Simple scatterplot matrices for one series can be obtained in R using the `lag.plot` command. Figure 2.8 and Figure 2.9 may be reproduced using the following scripts provided with `astsa`:

```
lag1.plot(soi, 12)      # Figure 2.8
lag2.plot(soi, rec, 8)  # Figure 2.9
```

Example 2.9 Regression with Lagged Variables (cont)

In Example 2.3 we regressed Recruitment on lagged SOI,

$$R_t = \beta_0 + \beta_1 S_{t-6} + w_t.$$

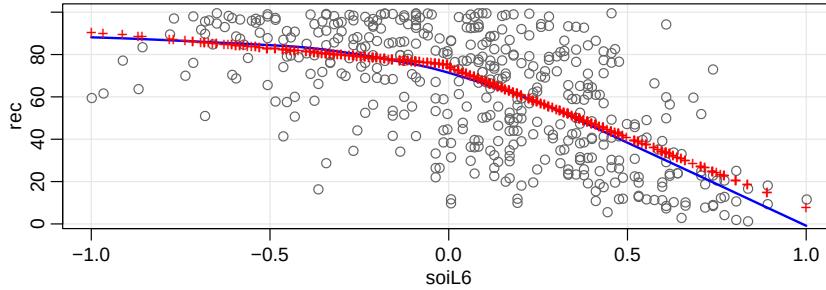


Fig. 2.10. Display for [Example 2.9](#): Plot of Recruitment (R_t) vs SOI lagged 6 months (S_{t-6}) with the fitted values of the regression as points (+) and a lowess fit (—).

However, in [Example 2.8](#), we saw that the relationship is nonlinear and different when SOI is positive or negative. In this case, we may consider adding a dummy variable to account for this change. In particular, we fit the model

$$R_t = \beta_0 + \beta_1 S_{t-6} + \beta_2 D_{t-6} + \beta_3 D_{t-6} S_{t-6} + w_t,$$

where D_t is a dummy variable that is 0 if $S_t < 0$ and 1 otherwise. This means that

$$R_t = \begin{cases} \beta_0 + \beta_1 S_{t-6} + w_t & \text{if } S_{t-6} < 0, \\ (\beta_0 + \beta_2) + (\beta_1 + \beta_3) S_{t-6} + w_t & \text{if } S_{t-6} \geq 0. \end{cases}$$

The result of the fit is given in the R code below. [Figure 2.10](#) shows R_t vs S_{t-6} with the fitted values of the regression and a lowess fit superimposed. The piecewise regression fit is similar to the lowess fit, but we note that the residuals are not white noise (see the code below). This is followed up in [Example 3.45](#).

```
dummy = ifelse(soi<0, 0, 1)
fish = ts.intersect(rec, soil6=lag(soi,-6), dL6=lag(dummy,-6), dframe=TRUE)
summary(fit <- lm(rec~soil6*dL6, data=fish, na.action=NULL))
Coefficients:
            Estimate Std. Error t.value
(Intercept)  74.479    2.865   25.998
soil6        -15.358    7.401   -2.075
dL6          -1.139    3.711   -0.307
soil6:dL6    -51.244    9.523   -5.381
---
Residual standard error: 21.84 on 443 degrees of freedom
Multiple R-squared:  0.4024
F-statistic: 99.43 on 3 and 443 DF
attach(fish)
plot(soil6, rec)
lines(lowess(soil6, rec), col=4, lwd=2)
points(soil6, fitted(fit), pch='+', col=2)
plot(resid(fit)) # not shown ...
acf(resid(fit)) # ... but obviously not noise
```

As a final exploratory tool, we discuss assessing periodic behavior in time series data using regression analysis. In [Example 1.12](#), we briefly discussed the problem of

identifying cyclic or periodic signals in time series. A number of the time series we have seen so far exhibit periodic behavior. For example, the data from the pollution study example shown in [Figure 2.2](#) exhibit strong yearly cycles. The Johnson & Johnson data shown in [Figure 1.1](#) make one cycle every year (four quarters) on top of an increasing trend and the speech data in [Figure 1.2](#) is highly repetitive. The monthly SOI and Recruitment series in [Figure 1.6](#) show strong yearly cycles, which obscures the slower El Niño cycle.

Example 2.10 Using Regression to Discover a Signal in Noise

In [Example 1.12](#), we generated $n = 500$ observations from the model

$$x_t = A \cos(2\pi\omega t + \phi) + w_t, \quad (2.35)$$

where $\omega = 1/50$, $A = 2$, $\phi = .6\pi$, and $\sigma_w = 5$; the data are shown on the bottom panel of [Figure 1.11](#). At this point we assume the frequency of oscillation $\omega = 1/50$ is known, but A and ϕ are unknown parameters. In this case the parameters appear in [\(2.35\)](#) in a nonlinear way, so we use a trigonometric identity^{2.4} and write

$$A \cos(2\pi\omega t + \phi) = \beta_1 \cos(2\pi\omega t) + \beta_2 \sin(2\pi\omega t),$$

where $\beta_1 = A \cos(\phi)$ and $\beta_2 = -A \sin(\phi)$. Now the model [\(2.35\)](#) can be written in the usual linear regression form given by (no intercept term is needed here)

$$x_t = \beta_1 \cos(2\pi t/50) + \beta_2 \sin(2\pi t/50) + w_t. \quad (2.36)$$

Using linear regression, we find $\hat{\beta}_1 = -.74_{(.33)}$, $\hat{\beta}_2 = -1.99_{(.33)}$ with $\hat{\sigma}_w = 5.18$; the values in parentheses are the standard errors. We note the actual values of the coefficients for this example are $\beta_1 = 2 \cos(.6\pi) = -.62$, and $\beta_2 = -2 \sin(.6\pi) = -1.90$. It is clear that we are able to detect the signal in the noise using regression, even though the signal-to-noise ratio is small. [Figure 2.11](#) shows data generated by [\(2.35\)](#) with the fitted line superimposed.

To reproduce the analysis and [Figure 2.11](#) in R, use the following:

```
set.seed(90210)          # so you can reproduce these results
x = 2*cos(2*pi*1:500/50 + .6*pi) + rnorm(500,0,5)
z1 = cos(2*pi*1:500/50)
z2 = sin(2*pi*1:500/50)
summary(fit <- lm(x~0+z1+z2)) # zero to exclude the intercept
Coefficients:
            Estimate Std. Error t value
z1     -0.7442    0.3274  -2.273
z2     -1.9949    0.3274 - 6.093
Residual standard error: 5.177 on 498 degrees of freedom
par(mfrow=c(2,1))
plot.ts(x)
plot.ts(x, col=8, ylab=expression(hat(x)))
lines(fitted(fit), col=2)
```

We will discuss this and related approaches in more detail in [Chapter 4](#).

^{2.4} $\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta)$.

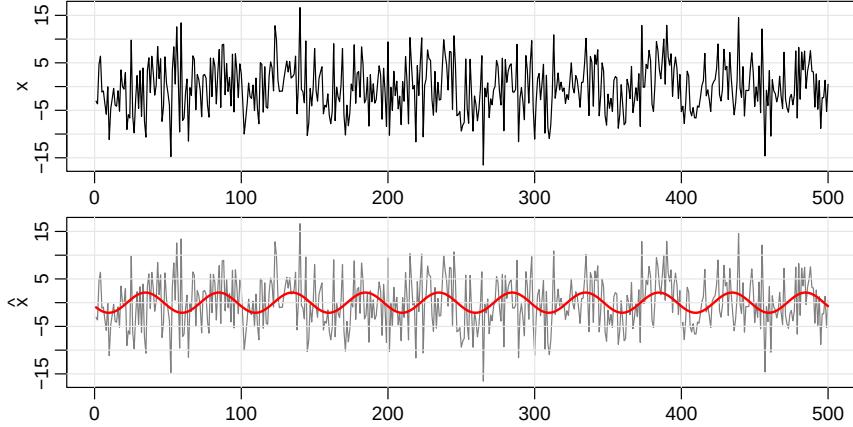


Fig. 2.11. Data generated by (2.35) [top] and the fitted line superimposed on the data [bottom].

2.3 Smoothing in the Time Series Context

In [Section 1.2](#), we introduced the concept of filtering or smoothing a time series, and in [Example 1.9](#), we discussed using a moving average to smooth white noise. This method is useful in discovering certain traits in a time series, such as long-term trend and seasonal components. In particular, if x_t represents the observations, then

$$m_t = \sum_{j=-k}^k a_j x_{t-j}, \quad (2.37)$$

where $a_j = a_{-j} \geq 0$ and $\sum_{j=-k}^k a_j = 1$ is a symmetric moving average of the data.

Example 2.11 Moving Average Smoother

For example, [Figure 2.12](#) shows the monthly SOI series discussed in [Example 1.5](#) smoothed using (2.37) with weights $a_0 = a_{\pm 1} = \dots = a_{\pm 5} = 1/12$, and $a_{\pm 6} = 1/24$; $k = 6$. This particular method removes (filters out) the obvious annual temperature cycle and helps emphasize the El Niño cycle. To reproduce [Figure 2.12](#) in R:

```
wgts = c(.5, rep(1,11), .5)/12
soif = filter(soi, sides=2, filter=wgts)
plot(soi)
lines(soif, lwd=2, col=4)
par(fig = c(.65, 1, .65, 1), new = TRUE) # the insert
nwgts = c(rep(0,20), wgts, rep(0,20))
plot(nwgts, type="l", ylim = c(-.02,.1), xaxt='n', yaxt='n', ann=FALSE)
```

Although the moving average smoother does a good job in highlighting the El Niño effect, it might be considered too choppy. We can obtain a smoother fit using the normal distribution for the weights, instead of boxcar-type weights of (2.37).

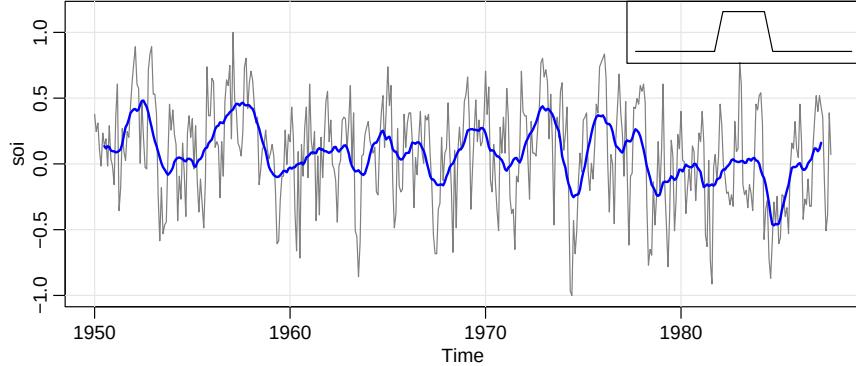


Fig. 2.12. Moving average smoother of SOI. The insert shows the shape of the moving average (“boxcar”) kernel [not drawn to scale] described in (2.39).

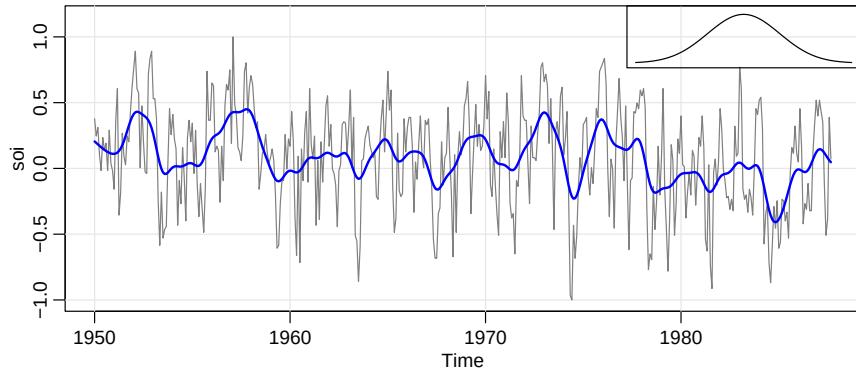


Fig. 2.13. Kernel smoother of SOI. The insert shows the shape of the normal kernel [not drawn to scale].

Example 2.12 Kernel Smoothing

Kernel smoothing is a moving average smoother that uses a weight function, or kernel, to average the observations. Figure 2.13 shows kernel smoothing of the SOI series, where m_t is now

$$m_t = \sum_{i=1}^n w_i(t) x_i, \quad (2.38)$$

where

$$w_i(t) = K\left(\frac{t-i}{b}\right) / \sum_{j=1}^n K\left(\frac{t-j}{b}\right) \quad (2.39)$$

are the weights and $K(\cdot)$ is a kernel function. This estimator, which was originally explored by Parzen (1962) and Rosenblatt (1956b), is often called the Nadaraya–Watson estimator (Watson, 1966). In this example, and typically, the normal kernel, $K(z) = \frac{1}{\sqrt{2\pi}} \exp(-z^2/2)$, is used.

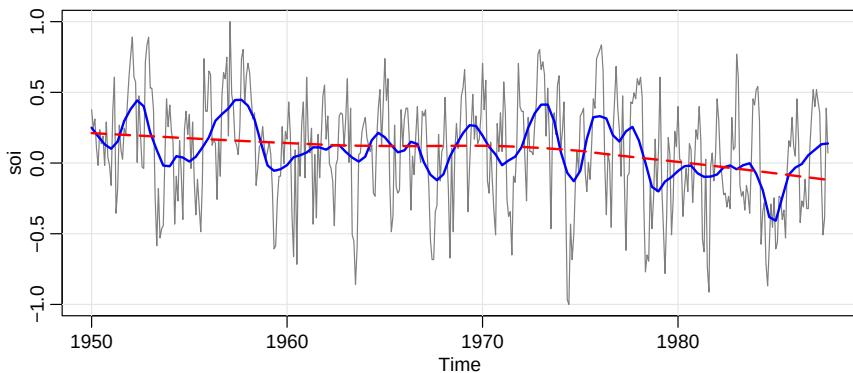


Fig. 2.14. Locally weighted scatterplot smoothers (lowess) of the SOI series.

To implement this in R, use the `ksmooth` function where a bandwidth can be chosen. The wider the bandwidth, b , the smoother the result. From the R `ksmooth` help file: The kernels are scaled so that their quartiles (viewed as probability densities) are at $\pm 0.25 \times \text{bandwidth}$. For the standard normal distribution, the quartiles are ± 0.674 . In our case, we are smoothing over time, which is of the form $t/12$ for the SOI time series. In Figure 2.13, we used the value of $b = 1$ to correspond to approximately smoothing a little over one year. Figure 2.13 can be reproduced in R as follows.

```
plot(soi)
lines(ksmooth(time(soi), soi, "normal", bandwidth=1), lwd=2, col=4)
par(fig = c(.65, 1, .65, 1), new = TRUE) # the insert
gauss = function(x) { 1/sqrt(2*pi) * exp(-(x^2)/2) }
x = seq(from = -3, to = 3, by = 0.001)
plot(x, gauss(x), type ="l", ylim=c(-.02,.45), xaxt='n', yaxt='n', ann=FALSE)
```

Example 2.13 Lowess

Another approach to smoothing a time plot is nearest neighbor regression. The technique is based on k -nearest neighbors regression, wherein one uses only the data $\{x_{t-k/2}, \dots, x_t, \dots, x_{t+k/2}\}$ to predict x_t via regression, and then sets $m_t = \hat{x}_t$.

Lowess is a method of smoothing that is rather complex, but the basic idea is close to nearest neighbor regression. Figure 2.14 shows smoothing of SOI using the R function `lowess` (see Cleveland, 1979). First, a certain proportion of nearest neighbors to x_t are included in a weighting scheme; values closer to x_t in time get more weight. Then, a robust weighted regression is used to predict x_t and obtain the smoothed values m_t . The larger the fraction of nearest neighbors included, the smoother the fit will be. In Figure 2.14, one smoother uses 5% of the data to obtain an estimate of the El Niño cycle of the data.

In addition, a (negative) trend in SOI would indicate the long-term warming of the Pacific Ocean. To investigate this, we used lowess with the default smoother span of `f=2/3` of the data. Figure 2.14 can be reproduced in R as follows.

```
plot(soi)
lines(lowess(soi, f=.05), lwd=2, col=4) # El Nino cycle
```

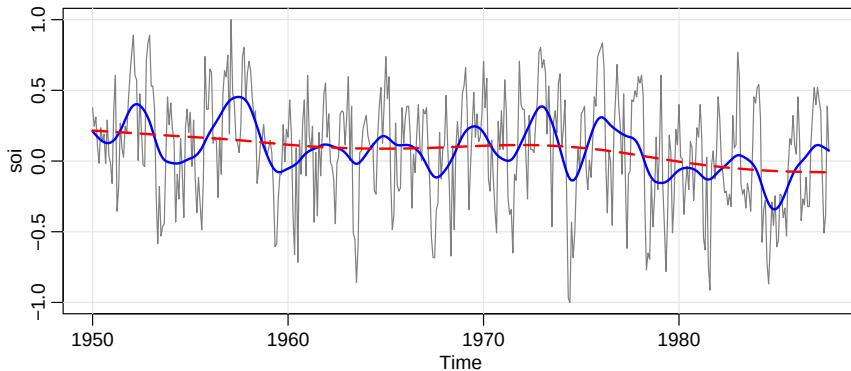


Fig. 2.15. Smoothing splines fit to the SOI series.

```
lines(lowess(soi), lty=2, lwd=2, col=2) # trend (with default span)
```

Example 2.14 Smoothing Splines

An obvious way to smooth data would be to fit a polynomial regression in terms of time. For example, a cubic polynomial would have $x_t = m_t + w_t$ where

$$m_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3.$$

We could then fit m_t via ordinary least squares.

An extension of polynomial regression is to first divide time $t = 1, \dots, n$, into k intervals, $[t_0 = 1, t_1], [t_1 + 1, t_2], \dots, [t_{k-1} + 1, t_k = n]$; the values t_0, t_1, \dots, t_k are called *knots*. Then, in each interval, one fits a polynomial regression, typically the order is 3, and this is called *cubic splines*.

A related method is *smoothing splines*, which minimizes a compromise between the fit and the degree of smoothness given by

$$\sum_{t=1}^n [x_t - m_t]^2 + \lambda \int (m_t'')^2 dt, \quad (2.40)$$

where m_t is a cubic spline with a knot at each t and primes denote differentiation. The degree of smoothness is controlled by $\lambda > 0$.

Think of taking a long drive where m_t is the position of your car at time t . In this case, m_t'' is instantaneous acceleration/deceleration, and $\int (m_t'')^2 dt$ is a measure of the total amount of acceleration and deceleration on your trip. A smooth drive would be one where a constant velocity is maintained (i.e., $m_t'' = 0$). A choppy ride would be when the driver is constantly accelerating and decelerating, such as beginning drivers tend to do.

If $\lambda = 0$, we don't care how choppy the ride is, and this leads to $m_t = x_t$, the data, which are not smooth. If $\lambda = \infty$, we insist on no acceleration or deceleration ($m_t'' = 0$); in this case, our drive must be at constant velocity, $m_t = c + vt$, and

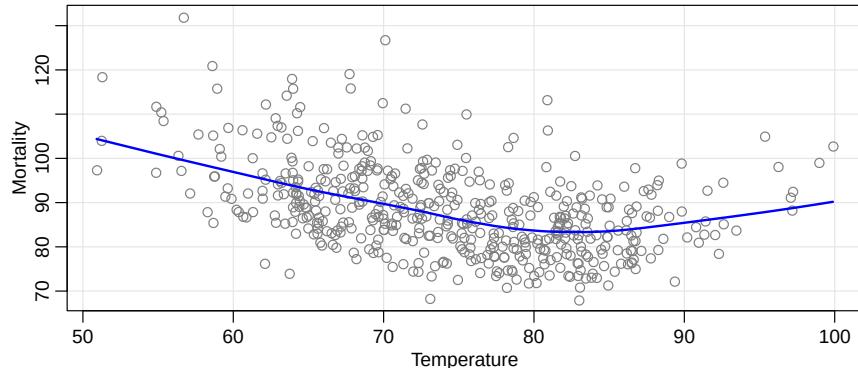


Fig. 2.16. Smooth of mortality as a function of temperature using lowess.

consequently very smooth. Thus, λ is seen as a trade-off between linear regression (completely smooth) and the data itself (no smoothness). The larger the value of λ , the smoother the fit.

In R, the smoothing parameter is called `spar` and it is monotonically related to λ ; type `?smooth.spline` to view the help file for details. Figure 2.15 shows smoothing spline fits on the SOI series using `spar=.5` to emphasize the El Niño cycle, and `spar=1` to emphasize the trend. The figure can be reproduced in R as follows.

```
plot(soi)
lines(smooth.spline(time(soi), soi, spar=.5), lwd=2, col=4)
lines(smooth.spline(time(soi), soi, spar= 1), lty=2, lwd=2, col=2)
```

Example 2.15 Smoothing One Series as a Function of Another

In addition to smoothing time plots, smoothing techniques can be applied to smoothing a time series as a function of another time series. We have already seen this idea used in Example 2.8 when we used lowess to visualize the nonlinear relationship between Recruitment and SOI at various lags. In this example, we smooth the scatterplot of two contemporaneously measured time series, mortality as a function of temperature. In Example 2.2, we discovered a nonlinear relationship between mortality and temperature. Continuing along these lines, Figure 2.16 show a scatterplot of mortality, M_t , and temperature, T_t , along with M_t smoothed as a function of T_t using lowess. Note that mortality increases at extreme temperatures, but in an asymmetric way; mortality is higher at colder temperatures than at hotter temperatures. The minimum mortality rate seems to occur at approximately 83° F.

Figure 2.16 can be reproduced in R as follows using the defaults.

```
plot(temp, cmort, xlab="Temperature", ylab="Mortality")
lines(lowess(temp, cmort))
```

Problems

Section 2.1

2.1 A Structural Model For the Johnson & Johnson data, say y_t , shown in Figure 1.1, let $x_t = \log(y_t)$. In this problem, we are going to fit a special type of structural model, $x_t = T_t + S_t + N_t$ where T_t is a trend component, S_t is a seasonal component, and N_t is noise. In our case, time t is in quarters (1960.00, 1960.25, ...) so one unit of time is a year.

- (a) Fit the regression model

$$x_t = \underbrace{\beta t}_{\text{trend}} + \underbrace{\alpha_1 Q_1(t) + \alpha_2 Q_2(t) + \alpha_3 Q_3(t) + \alpha_4 Q_4(t)}_{\text{seasonal}} + \underbrace{w_t}_{\text{noise}}$$

where $Q_i(t) = 1$ if time t corresponds to quarter $i = 1, 2, 3, 4$, and zero otherwise. The $Q_i(t)$'s are called indicator variables. We will assume for now that w_t is a Gaussian white noise sequence. Hint: Detailed code is given in [Code R.4](#), the last example of [Section R.4](#).

- (b) If the model is correct, what is the estimated average annual increase in the logged earnings per share?
(c) If the model is correct, does the average logged earnings rate increase or decrease from the third quarter to the fourth quarter? And, by what percentage does it increase or decrease?
(d) What happens if you include an intercept term in the model in (a)? Explain why there was a problem.
(e) Graph the data, x_t , and superimpose the fitted values, say \hat{x}_t , on the graph. Examine the residuals, $x_t - \hat{x}_t$, and state your conclusions. Does it appear that the model fits the data well (do the residuals look white)?

2.2 For the mortality data examined in [Example 2.2](#):

- (a) Add another component to the regression in [\(2.21\)](#) that accounts for the particulate count four weeks prior; that is, add P_{t-4} to the regression in [\(2.21\)](#). State your conclusion.
(b) Draw a scatterplot matrix of M_t, T_t, P_t and P_{t-4} and then calculate the pairwise correlations between the series. Compare the relationship between M_t and P_t versus M_t and P_{t-4} .

2.3 In this problem, we explore the difference between a random walk and a trend stationary process.

- (a) Generate *four* series that are random walk with drift, [\(1.4\)](#), of length $n = 100$ with $\delta = .01$ and $\sigma_w = 1$. Call the data x_t for $t = 1, \dots, 100$. Fit the regression $x_t = \beta t + w_t$ using least squares. Plot the data, the true mean function (i.e., $\mu_t = .01 t$) and the fitted line, $\hat{x}_t = \hat{\beta} t$, on the same graph. Hint: The following R code may be useful.

```

par(mfrow=c(2,2), mar=c(2.5,2.5,0,0)+.5, mgp=c(1.6,.6,0)) # set up
for (i in 1:4){
  x = ts(cumsum(rnorm(100,.01,1))) # data
  regx = lm(x~0+time(x), na.action=NULL) # regression
  plot(x, ylab='Random Walk w Drift') # plots
  abline(a=0, b=.01, col=2, lty=2) # true mean (red - dashed)
  abline(regx, col=4) # fitted line (blue - solid)
}

```

- (b) Generate *four* series of length $n = 100$ that are linear trend plus noise, say $y_t = .01t + w_t$, where t and w_t are as in part (a). Fit the regression $y_t = \beta t + w_t$ using least squares. Plot the data, the true mean function (i.e., $\mu_t = .01t$) and the fitted line, $\hat{y}_t = \hat{\beta}t$, on the same graph.
- (c) Comment (what did you learn from this assignment).

2.4 Kullback-Leibler Information Given the random $n \times 1$ vector y , we define the information for discriminating between two densities in the same family, indexed by a parameter θ , say $f(y; \theta_1)$ and $f(y; \theta_2)$, as

$$I(\theta_1; \theta_2) = n^{-1} E_1 \log \frac{f(y; \theta_1)}{f(y; \theta_2)}, \quad (2.41)$$

where E_1 denotes expectation with respect to the density determined by θ_1 . For the Gaussian regression model, the parameters are $\theta = (\beta', \sigma^2)'$. Show that

$$I(\theta_1; \theta_2) = \frac{1}{2} \left(\frac{\sigma_1^2}{\sigma_2^2} - \log \frac{\sigma_1^2}{\sigma_2^2} - 1 \right) + \frac{1}{2} \frac{(\beta_1 - \beta_2)' Z' Z (\beta_1 - \beta_2)}{n \sigma_2^2}. \quad (2.42)$$

2.5 Model Selection Both selection criteria (2.15) and (2.16) are derived from information theoretic arguments, based on the well-known *Kullback-Leibler discrimination information* numbers (see Kullback and Leibler, 1951, Kullback, 1958). We give an argument due to Hurvich and Tsai (1989). We think of the measure (2.42) as measuring the discrepancy between the two densities, characterized by the parameter values $\theta'_1 = (\beta'_1, \sigma_1^{2'})'$ and $\theta'_2 = (\beta'_2, \sigma_2^{2'})'$. Now, if the true value of the parameter vector is θ_1 , we argue that the best model would be one that minimizes the discrepancy between the theoretical value and the sample, say $I(\theta_1; \hat{\theta})$. Because θ_1 will not be known, Hurvich and Tsai (1989) considered finding an unbiased estimator for $E_1[I(\beta_1, \sigma_1^2; \hat{\beta}, \hat{\sigma}^2)]$, where

$$I(\beta_1, \sigma_1^2; \hat{\beta}, \hat{\sigma}^2) = \frac{1}{2} \left(\frac{\sigma_1^2}{\hat{\sigma}^2} - \log \frac{\sigma_1^2}{\hat{\sigma}^2} - 1 \right) + \frac{1}{2} \frac{(\beta_1 - \hat{\beta})' Z' Z (\beta_1 - \hat{\beta})}{n \hat{\sigma}^2}$$

and β is a $k \times 1$ regression vector. Show that

$$E_1[I(\beta_1, \sigma_1^2; \hat{\beta}, \hat{\sigma}^2)] = \frac{1}{2} \left(-\log \sigma_1^2 + E_1 \log \hat{\sigma}^2 + \frac{n+k}{n-k-2} - 1 \right), \quad (2.43)$$

using the distributional properties of the regression coefficients and error variance. An unbiased estimator for $E_1 \log \hat{\sigma}^2$ is $\log \hat{\sigma}^2$. Hence, we have shown that the expectation

of the above discrimination information is as claimed. As models with differing dimensions k are considered, only the second and third terms in (2.43) will vary and we only need unbiased estimators for those two terms. This gives the form of AICc quoted in (2.16) in the chapter. You will need the two distributional results

$$\frac{n\hat{\sigma}^2}{\sigma_1^2} \sim \chi_{n-k}^2 \quad \text{and} \quad \frac{(\hat{\beta} - \beta_1)'Z'Z(\hat{\beta} - \beta_1)}{\sigma_1^2} \sim \chi_k^2$$

The two quantities are distributed independently as chi-squared distributions with the indicated degrees of freedom. If $x \sim \chi_n^2$, $E(1/x) = 1/(n-2)$.

Section 2.2

2.6 Consider a process consisting of a linear trend with an additive noise term consisting of independent random variables w_t with zero means and variances σ_w^2 , that is,

$$x_t = \beta_0 + \beta_1 t + w_t,$$

where β_0, β_1 are fixed constants.

- (a) Prove x_t is nonstationary.
- (b) Prove that the first difference series $\nabla x_t = x_t - x_{t-1}$ is stationary by finding its mean and autocovariance function.
- (c) Repeat part (b) if w_t is replaced by a general stationary process, say y_t , with mean function μ_y and autocovariance function $\gamma_y(h)$.

2.7 Show (2.27) is stationary.

2.8 The glacial varve record plotted in Figure 2.7 exhibits some nonstationarity that can be improved by transforming to logarithms and some additional nonstationarity that can be corrected by differencing the logarithms.

- (a) Argue that the glacial varves series, say x_t , exhibits heteroscedasticity by computing the sample variance over the first half and the second half of the data. Argue that the transformation $y_t = \log x_t$ stabilizes the variance over the series. Plot the histograms of x_t and y_t to see whether the approximation to normality is improved by transforming the data.
- (b) Plot the series y_t . Do any time intervals, of the order 100 years, exist where one can observe behavior comparable to that observed in the global temperature records in Figure 1.2?
- (c) Examine the sample ACF of y_t and comment.
- (d) Compute the difference $u_t = y_t - y_{t-1}$, examine its time plot and sample ACF, and argue that differencing the logged varve data produces a reasonably stationary series. Can you think of a practical interpretation for u_t ? Hint: Recall Footnote 1.2.

- (e) Based on the sample ACF of the differenced transformed series computed in (c), argue that a generalization of the model given by [Example 1.26](#) might be reasonable. Assume

$$u_t = \mu + w_t + \theta w_{t-1}$$

is stationary when the inputs w_t are assumed independent with mean 0 and variance σ_w^2 . Show that

$$\gamma_u(h) = \begin{cases} \sigma_w^2(1 + \theta^2) & \text{if } h = 0, \\ \theta \sigma_w^2 & \text{if } h = \pm 1, \\ 0 & \text{if } |h| > 1. \end{cases}$$

- (f) Based on part (e), use $\hat{\rho}_u(1)$ and the estimate of the variance of u_t , $\hat{\gamma}_u(0)$, to derive estimates of θ and σ_w^2 . This is an application of the method of moments from classical statistics, where estimators of the parameters are derived by equating sample moments to theoretical moments.

- 2.9** In this problem, we will explore the periodic nature of S_t , the SOI series displayed in [Figure 1.5](#).

- (a) Detrend the series by fitting a regression of S_t on time t . Is there a significant trend in the sea surface temperature? Comment.
- (b) Calculate the periodogram for the detrended series obtained in part (a). Identify the frequencies of the two main peaks (with an obvious one at the frequency of one cycle every 12 months). What is the probable El Niño cycle indicated by the minor peak?

Section 2.3

- 2.10** Consider the two weekly time series [oil](#) and [gas](#). The oil series is in dollars per barrel, while the gas series is in cents per gallon.

- (a) Plot the data on the same graph. Which of the simulated series displayed in [Section 1.2](#) do these series most resemble? Do you believe the series are stationary (explain your answer)?
- (b) In economics, it is often the percentage change in price (termed *growth rate* or *return*), rather than the absolute price change, that is important. Argue that a transformation of the form $y_t = \nabla \log x_t$ might be applied to the data, where x_t is the oil or gas price series. *Hint:* Recall [Footnote 1.2](#).
- (c) Transform the data as described in part (b), plot the data on the same graph, look at the sample ACFs of the transformed data, and comment.
- (d) Plot the CCF of the transformed data and comment. The small, but significant values when [gas](#) leads [oil](#) might be considered as feedback.
- (e) Exhibit scatterplots of the oil and gas growth rate series for up to three weeks of lead time of oil prices; include a nonparametric smoother in each plot and comment on the results (e.g., Are there outliers? Are the relationships linear?).

- (f) There have been a number of studies questioning whether gasoline prices respond more quickly when oil prices are rising than when oil prices are falling (“asymmetry”). We will attempt to explore this question here with simple lagged regression; we will ignore some obvious problems such as outliers and autocorrelated errors, so this will not be a definitive analysis. Let G_t and O_t denote the gas and oil growth rates.

- (i) Fit the regression (and comment on the results)

$$G_t = \alpha_1 + \alpha_2 I_t + \beta_1 O_t + \beta_2 O_{t-1} + w_t,$$

where $I_t = 1$ if $O_t \geq 0$ and 0 otherwise (I_t is the indicator of no growth or positive growth in oil price). Hint:

```
poil = diff(log(oil))
pgas = diff(log(gas))
indi = ifelse(poil < 0, 0, 1)
mess = ts.intersect(pgas, poil, poill = lag(poil,-1), indi)
summary(fit <- lm(pgas~ poil + poill + indi, data=mess))
```

- (ii) What is the fitted model when there is negative growth in oil price at time t ? What is the fitted model when there is no or positive growth in oil price? Do these results support the asymmetry hypothesis?
- (iii) Analyze the residuals from the fit and comment.

- 2.11** Use two different smoothing techniques described in Section 2.3 to estimate the trend in the global temperature series `globtemp`. Comment.

Chapter 3

ARIMA Models

Classical regression is often insufficient for explaining all of the interesting dynamics of a time series. For example, the ACF of the residuals of the simple linear regression fit to the price of chicken data (see [Example 2.4](#)) reveals additional structure in the data that regression did not capture. Instead, the introduction of correlation that may be generated through lagged linear relations leads to proposing the *autoregressive (AR)* and *autoregressive moving average (ARMA)* models that were presented in Whittle (1951). Adding nonstationary models to the mix leads to the *autoregressive integrated moving average (ARIMA)* model popularized in the landmark work by Box and Jenkins (1970). The *Box–Jenkins method* for identifying ARIMA models is given in this chapter along with techniques for *parameter estimation* and *forecasting* for these models. A partial theoretical justification of the use of ARMA models is discussed in [Section B.4](#).

3.1 Autoregressive Moving Average Models

The classical regression model of [Chapter 2](#) was developed for the static case, namely, we only allow the dependent variable to be influenced by current values of the independent variables. In the time series case, it is desirable to allow the dependent variable to be influenced by the past values of the independent variables and possibly by its own past values. If the present can be plausibly modeled in terms of only the past values of the independent inputs, we have the enticing prospect that forecasting will be possible.

INTRODUCTION TO AUTOREGRESSIVE MODELS

Autoregressive models are based on the idea that the current value of the series, x_t , can be explained as a function of p past values, $x_{t-1}, x_{t-2}, \dots, x_{t-p}$, where p determines the number of steps into the past needed to forecast the current value. As a typical case, recall [Example 1.10](#) in which data were generated using the model

$$x_t = x_{t-1} - .90x_{t-2} + w_t,$$

where w_t is white Gaussian noise with $\sigma_w^2 = 1$. We have now assumed the current value is a particular *linear* function of past values. The regularity that persists in [Figure 1.9](#) gives an indication that forecasting for such a model might be a distinct possibility, say, through some version such as

$$x_{n+1}^n = x_n - .90x_{n-1},$$

where the quantity on the left-hand side denotes the forecast at the next period $n + 1$ based on the observed data, x_1, x_2, \dots, x_n . We will make this notion more precise in our discussion of forecasting ([Section 3.4](#)).

The extent to which it might be possible to forecast a real data series from its own past values can be assessed by looking at the autocorrelation function and the lagged scatterplot matrices discussed in [Chapter 2](#). For example, the lagged scatterplot matrix for the Southern Oscillation Index (SOI), shown in [Figure 2.8](#), gives a distinct indication that lags 1 and 2, for example, are linearly associated with the current value. The ACF shown in [Figure 1.16](#) shows relatively large positive values at lags 1, 2, 12, 24, and 36 and large negative values at 18, 30, and 42. We note also the possible relation between the SOI and Recruitment series indicated in the scatterplot matrix shown in [Figure 2.9](#). We will indicate in later sections on transfer function and vector AR modeling how to handle the dependence on values taken by other series.

The preceding discussion motivates the following definition.

Definition 3.1 An autoregressive model of order p , abbreviated **AR**(p), is of the form

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t, \quad (3.1)$$

where x_t is stationary, $w_t \sim \text{wn}(0, \sigma_w^2)$, and $\phi_1, \phi_2, \dots, \phi_p$ are constants ($\phi_p \neq 0$). The mean of x_t in (3.1) is zero. If the mean, μ , of x_t is not zero, replace x_t by $x_t - \mu$ in (3.1).

$$x_t - \mu = \phi_1(x_{t-1} - \mu) + \phi_2(x_{t-2} - \mu) + \cdots + \phi_p(x_{t-p} - \mu) + w_t,$$

or write

$$x_t = \alpha + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t, \quad (3.2)$$

where $\alpha = \mu(1 - \phi_1 - \cdots - \phi_p)$.

We note that (3.2) is similar to the regression model of [Section 2.1](#), and hence the term auto (or self) regression. Some technical difficulties, however, develop from applying that model because the regressors, x_{t-1}, \dots, x_{t-p} , are random components, whereas z_t was assumed to be fixed. A useful form follows by using the backshift operator (2.29) to write the AR(p) model, (3.1), as

$$(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)x_t = w_t, \quad (3.3)$$

or even more concisely as

$$\phi(B)x_t = w_t. \quad (3.4)$$

The properties of $\phi(B)$ are important in solving (3.4) for x_t . This leads to the following definition.

Definition 3.2 The autoregressive operator is defined to be

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p. \quad (3.5)$$

Example 3.1 The AR(1) Model

We initiate the investigation of AR models by considering the first-order model, AR(1), given by $x_t = \phi x_{t-1} + w_t$. Iterating backwards k times, we get

$$\begin{aligned} x_t &= \phi x_{t-1} + w_t = \phi(\phi x_{t-2} + w_{t-1}) + w_t \\ &= \phi^2 x_{t-2} + \phi w_{t-1} + w_t \\ &\vdots \\ &= \phi^k x_{t-k} + \sum_{j=0}^{k-1} \phi^j w_{t-j}. \end{aligned}$$

This method suggests that, by continuing to iterate backward, and provided that $|\phi| < 1$ and $\sup_t \text{var}(x_t) < \infty$, we can represent an AR(1) model as a linear process given by^{3.1}

$$x_t = \sum_{j=0}^{\infty} \phi^j w_{t-j}. \quad (3.6)$$

Representation (3.6) is called the stationary solution of the model. In fact, by simple substitution,

$$\underbrace{\sum_{j=0}^{\infty} \phi^j w_{t-j}}_{x_t} = \phi \underbrace{\left(\sum_{k=0}^{\infty} \phi^k w_{t-1-k} \right)}_{x_{t-1}} + w_t.$$

The AR(1) process defined by (3.6) is stationary with mean

$$E(x_t) = \sum_{j=0}^{\infty} \phi^j E(w_{t-j}) = 0,$$

and autocovariance function,

$$\begin{aligned} \gamma(h) &= \text{cov}(x_{t+h}, x_t) = E \left[\left(\sum_{j=0}^{\infty} \phi^j w_{t+h-j} \right) \left(\sum_{k=0}^{\infty} \phi^k w_{t-k} \right) \right] \\ &= E \left[\left(w_{t+h} + \cdots + \phi^h w_t + \phi^{h+1} w_{t-1} + \cdots \right) (w_t + \phi w_{t-1} + \cdots) \right] \quad (3.7) \\ &= \sigma_w^2 \sum_{j=0}^{\infty} \phi^{h+j} \phi^j = \sigma_w^2 \phi^h \sum_{j=0}^{\infty} \phi^{2j} = \frac{\sigma_w^2 \phi^h}{1 - \phi^2}, \quad h \geq 0. \end{aligned}$$

^{3.1} Note that $\lim_{k \rightarrow \infty} E \left(x_t - \sum_{j=0}^{k-1} \phi^j w_{t-j} \right)^2 = \lim_{k \rightarrow \infty} \phi^{2k} E \left(x_{t-k}^2 \right) = 0$, so (3.6) exists in the mean square sense (see Appendix A for a definition).

Recall that $\gamma(h) = \gamma(-h)$, so we will only exhibit the autocovariance function for $h \geq 0$. From (3.7), the ACF of an AR(1) is

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)} = \phi^h, \quad h \geq 0, \quad (3.8)$$

and $\rho(h)$ satisfies the recursion

$$\rho(h) = \phi \rho(h-1), \quad h = 1, 2, \dots . \quad (3.9)$$

We will discuss the ACF of a general AR(p) model in [Section 3.3](#).

Example 3.2 The Sample Path of an AR(1) Process

[Figure 3.1](#) shows a time plot of two AR(1) processes, one with $\phi = .9$ and one with $\phi = -.9$; in both cases, $\sigma_w^2 = 1$. In the first case, $\rho(h) = .9^h$, for $h \geq 0$, so observations close together in time are positively correlated with each other. This result means that observations at contiguous time points will tend to be close in value to each other; this fact shows up in the top of [Figure 3.1](#) as a very smooth sample path for x_t . Now, contrast this with the case in which $\phi = -.9$, so that $\rho(h) = (-.9)^h$, for $h \geq 0$. This result means that observations at contiguous time points are negatively correlated but observations two time points apart are positively correlated. This fact shows up in the bottom of [Figure 3.1](#), where, for example, if an observation, x_t , is positive, the next observation, x_{t+1} , is typically negative, and the next observation, x_{t+2} , is typically positive. Thus, in this case, the sample path is very choppy.

The following R code can be used to obtain a figure similar to [Figure 3.1](#):

```
par(mfrow=c(2,1))
plot(arima.sim(list(order=c(1,0,0), ar=.9), n=100), ylab="x",
         main=(expression(AR(1)~~~phi==+.9)))
plot(arima.sim(list(order=c(1,0,0), ar=-.9), n=100), ylab="x",
         main=(expression(AR(1)~~~phi==-.9)))
```

Example 3.3 Explosive AR Models and Causality

In [Example 1.18](#), it was discovered that the random walk $x_t = x_{t-1} + w_t$ is not stationary. We might wonder whether there is a stationary AR(1) process with $|\phi| > 1$. Such processes are called explosive because the values of the time series quickly become large in magnitude. Clearly, because $|\phi|^j$ increases without bound as $j \rightarrow \infty$, $\sum_{j=0}^{k-1} \phi^j w_{t-j}$ will not converge (in mean square) as $k \rightarrow \infty$, so the intuition used to get (3.6) will not work directly. We can, however, modify that argument to obtain a stationary model as follows. Write $x_{t+1} = \phi x_t + w_{t+1}$, in which case,

$$\begin{aligned} x_t &= \phi^{-1} x_{t+1} - \phi^{-1} w_{t+1} = \phi^{-1} (\phi^{-1} x_{t+2} - \phi^{-1} w_{t+2}) - \phi^{-1} w_{t+1} \\ &\vdots \\ &= \phi^{-k} x_{t+k} - \sum_{j=1}^{k-1} \phi^{-j} w_{t+j}, \end{aligned} \quad (3.10)$$

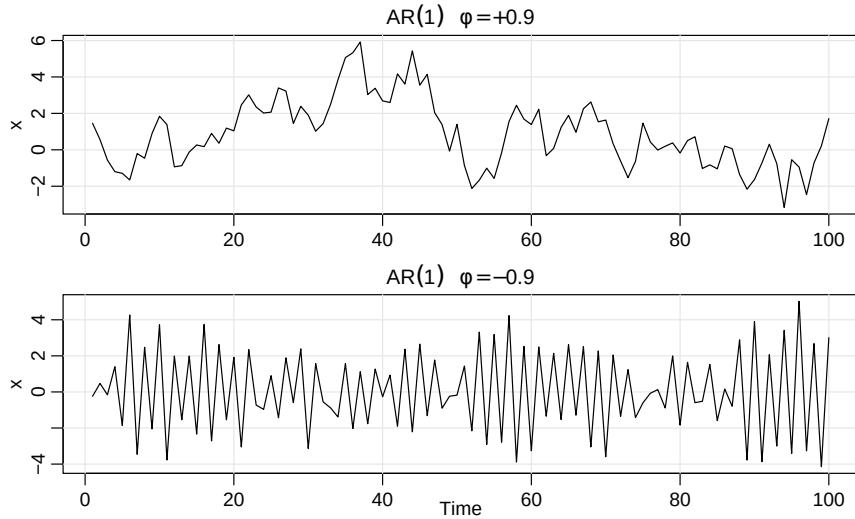


Fig. 3.1. Simulated AR(1) models: $\phi = .9$ (top); $\phi = -.9$ (bottom).

by iterating forward k steps. Because $|\phi|^{-1} < 1$, this result suggests the stationary future dependent AR(1) model

$$x_t = - \sum_{j=1}^{\infty} \phi^{-j} w_{t+j}. \quad (3.11)$$

The reader can verify that this is stationary and of the AR(1) form $x_t = \phi x_{t-1} + w_t$. Unfortunately, this model is useless because it requires us to know the future to be able to predict the future. When a process does not depend on the future, such as the AR(1) when $|\phi| < 1$, we will say the process is *causal*. In the explosive case of this example, the process is stationary, but it is also future dependent, and not causal.

Example 3.4 Every Explosion Has a Cause

Excluding explosive models from consideration is not a problem because the models have causal counterparts. For example, if

$$x_t = \phi x_{t-1} + w_t \quad \text{with} \quad |\phi| > 1$$

and $w_t \sim \text{iid } N(0, \sigma_w^2)$, then using (3.11), $\{x_t\}$ is a non-causal stationary Gaussian process with $E(x_t) = 0$ and

$$\begin{aligned} \gamma_x(h) &= \text{cov}(x_{t+h}, x_t) = \text{cov}\left(- \sum_{j=1}^{\infty} \phi^{-j} w_{t+h+j}, - \sum_{k=1}^{\infty} \phi^{-k} w_{t+k}\right) \\ &= \sigma_w^2 \phi^{-2} \phi^{-h} / (1 - \phi^{-2}). \end{aligned}$$

Thus, using (3.7), the causal process defined by

$$y_t = \phi^{-1} y_{t-1} + v_t$$

where $v_t \sim \text{iid } N(0, \sigma_w^2 \phi^{-2})$ is stochastically equal to the x_t process (i.e., all finite distributions of the processes are the same). For example, if $x_t = 2x_{t-1} + w_t$ with $\sigma_w^2 = 1$, then $y_t = \frac{1}{2}y_{t-1} + v_t$ with $\sigma_v^2 = 1/4$ is an equivalent causal process (see Problem 3.3). This concept generalizes to higher orders, but it is easier to show using Chapter 4 techniques; see Example 4.8.

The technique of iterating backward to get an idea of the stationary solution of AR models works well when $p = 1$, but not for larger orders. A general technique is that of matching coefficients. Consider the AR(1) model in operator form

$$\phi(B)x_t = w_t, \quad (3.12)$$

where $\phi(B) = 1 - \phi B$, and $|\phi| < 1$. Also, write the model in equation (3.6) using operator form as

$$x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j} = \psi(B)w_t, \quad (3.13)$$

where $\psi(B) = \sum_{j=0}^{\infty} \psi_j B^j$ and $\psi_j = \phi^j$. Suppose we did not know that $\psi_j = \phi^j$. We could substitute $\psi(B)w_t$ from (3.13) for x_t in (3.12) to obtain

$$\phi(B)\psi(B)w_t = w_t. \quad (3.14)$$

The coefficients of B on the left-hand side of (3.14) must be equal to those on right-hand side of (3.14), which means

$$(1 - \phi B)(1 + \psi_1 B + \psi_2 B^2 + \cdots + \psi_j B^j + \cdots) = 1. \quad (3.15)$$

Reorganizing the coefficients in (3.15),

$$1 + (\psi_1 - \phi)B + (\psi_2 - \psi_1\phi)B^2 + \cdots + (\psi_j - \psi_{j-1}\phi)B^j + \cdots = 1,$$

we see that for each $j = 1, 2, \dots$, the coefficient of B^j on the left must be zero because it is zero on the right. The coefficient of B on the left is $(\psi_1 - \phi)$, and equating this to zero, $\psi_1 - \phi = 0$, leads to $\psi_1 = \phi$. Continuing, the coefficient of B^2 is $(\psi_2 - \psi_1\phi)$, so $\psi_2 = \phi^2$. In general,

$$\psi_j = \psi_{j-1}\phi,$$

with $\psi_0 = 1$, which leads to the solution $\psi_j = \phi^j$.

Another way to think about the operations we just performed is to consider the AR(1) model in operator form, $\phi(B)x_t = w_t$. Now multiply both sides by $\phi^{-1}(B)$ (assuming the inverse operator exists) to get

$$\phi^{-1}(B)\phi(B)x_t = \phi^{-1}(B)w_t,$$

or

$$x_t = \phi^{-1}(B)w_t.$$

We know already that

$$\phi^{-1}(B) = 1 + \phi B + \phi^2 B^2 + \cdots + \phi^j B^j + \cdots,$$

that is, $\phi^{-1}(B)$ is $\psi(B)$ in (3.13). Thus, we notice that working with operators is like working with polynomials. That is, consider the polynomial $\phi(z) = 1 - \phi z$, where z is a complex number and $|\phi| < 1$. Then,

$$\phi^{-1}(z) = \frac{1}{(1 - \phi z)} = 1 + \phi z + \phi^2 z^2 + \cdots + \phi^j z^j + \cdots, \quad |z| \leq 1,$$

and the coefficients of B^j in $\phi^{-1}(B)$ are the same as the coefficients of z^j in $\phi^{-1}(z)$. In other words, we may treat the backshift operator, B , as a complex number, z . These results will be generalized in our discussion of ARMA models. We will find the polynomials corresponding to the operators useful in exploring the general properties of ARMA models.

INTRODUCTION TO MOVING AVERAGE MODELS

As an alternative to the autoregressive representation in which the x_t on the left-hand side of the equation are assumed to be combined linearly, the moving average model of order q , abbreviated as MA(q), assumes the white noise w_t on the right-hand side of the defining equation are combined linearly to form the observed data.

Definition 3.3 *The moving average model of order q , or MA(q) model, is defined to be*

$$x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \cdots + \theta_q w_{t-q}, \quad (3.16)$$

where $w_t \sim \text{wn}(0, \sigma_w^2)$, and $\theta_1, \theta_2, \dots, \theta_q$ ($\theta_q \neq 0$) are parameters.^{3.2}

The system is the same as the infinite moving average defined as the linear process (3.13), where $\psi_0 = 1$, $\psi_j = \theta_j$, for $j = 1, \dots, q$, and $\psi_j = 0$ for other values. We may also write the MA(q) process in the equivalent form

$$x_t = \theta(B)w_t, \quad (3.17)$$

using the following definition.

Definition 3.4 *The moving average operator is*

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q. \quad (3.18)$$

Unlike the autoregressive process, the moving average process is stationary for any values of the parameters $\theta_1, \dots, \theta_q$; details of this result are provided in Section 3.3.

^{3.2} Some texts and software packages write the MA model with negative coefficients; that is, $x_t = w_t - \theta_1 w_{t-1} - \theta_2 w_{t-2} - \cdots - \theta_q w_{t-q}$.

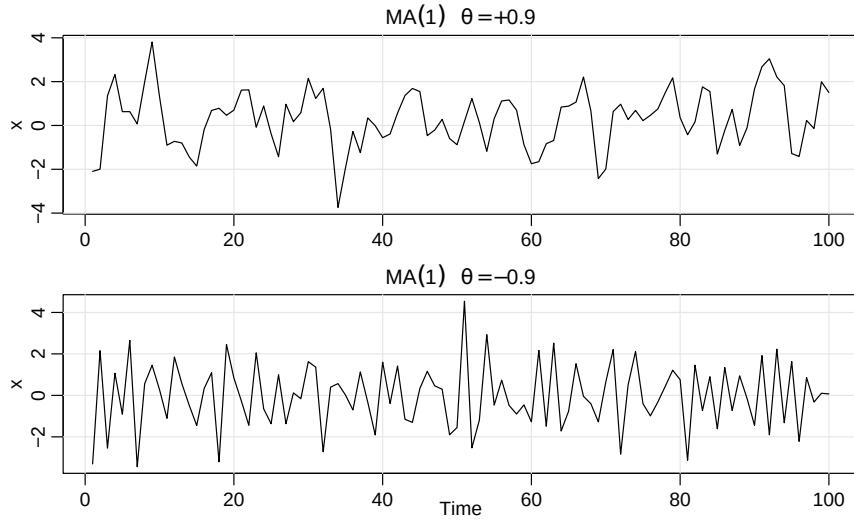


Fig. 3.2. Simulated MA(1) models: $\theta = .9$ (top); $\theta = -.9$ (bottom).

Example 3.5 The MA(1) Process

Consider the MA(1) model $x_t = w_t + \theta w_{t-1}$. Then, $E(x_t) = 0$,

$$\gamma(h) = \begin{cases} (1 + \theta^2)\sigma_w^2 & h = 0, \\ \theta\sigma_w^2 & h = 1, \\ 0 & h > 1, \end{cases}$$

and the ACF is

$$\rho(h) = \begin{cases} \frac{\theta}{(1+\theta^2)} & h = 1, \\ 0 & h > 1. \end{cases}$$

Note $|\rho(1)| \leq 1/2$ for all values of θ (Problem 3.1). Also, x_t is correlated with x_{t-1} , but not with x_{t-2}, x_{t-3}, \dots . Contrast this with the case of the AR(1) model in which the correlation between x_t and x_{t-k} is never zero. When $\theta = .9$, for example, x_t and x_{t-1} are positively correlated, and $\rho(1) = .497$. When $\theta = -.9$, x_t and x_{t-1} are negatively correlated, $\rho(1) = -.497$. Figure 3.2 shows a time plot of these two processes with $\sigma_w^2 = 1$. The series for which $\theta = .9$ is smoother than the series for which $\theta = -.9$.

A figure similar to Figure 3.2 can be created in R as follows:

```
par(mfrow = c(2,1))
plot(arima.sim(list(order=c(0,0,1), ma=.9), n=100), ylab="x",
      main=(expression(MA(1)~~~theta==+.5)))
plot(arima.sim(list(order=c(0,0,1), ma=-.9), n=100), ylab="x",
      main=(expression(MA(1)~~~theta==-.5)))
```

Example 3.6 Non-uniqueness of MA Models and Invertibility

Using Example 3.5, we note that for an MA(1) model, $\rho(h)$ is the same for θ and $\frac{1}{\theta}$; try 5 and $\frac{1}{5}$, for example. In addition, the pair $\sigma_w^2 = 1$ and $\theta = 5$ yield the same autocovariance function as the pair $\sigma_w^2 = 25$ and $\theta = 1/5$, namely,

$$\gamma(h) = \begin{cases} 26 & h = 0, \\ 5 & h = 1, \\ 0 & h > 1. \end{cases}$$

Thus, the MA(1) processes

$$x_t = w_t + \frac{1}{5}w_{t-1}, \quad w_t \sim \text{iid } N(0, 25)$$

and

$$y_t = v_t + 5v_{t-1}, \quad v_t \sim \text{iid } N(0, 1)$$

are the same because of normality (i.e., all finite distributions are the same). We can only observe the time series, x_t or y_t , and not the noise, w_t or v_t , so we cannot distinguish between the models. Hence, we will have to choose only one of them. For convenience, by mimicking the criterion of causality for AR models, we will choose the model with an infinite AR representation. Such a process is called an *invertible* process.

To discover which model is the invertible model, we can reverse the roles of x_t and w_t (because we are mimicking the AR case) and write the MA(1) model as $w_t = -\theta w_{t-1} + x_t$. Following the steps that led to (3.6), if $|\theta| < 1$, then $w_t = \sum_{j=0}^{\infty} (-\theta)^j x_{t-j}$, which is the desired infinite AR representation of the model. Hence, given a choice, we will choose the model with $\sigma_w^2 = 25$ and $\theta = 1/5$ because it is invertible.

As in the AR case, the polynomial, $\theta(z)$, corresponding to the moving average operators, $\theta(B)$, will be useful in exploring general properties of MA processes. For example, following the steps of equations (3.12)–(3.15), we can write the MA(1) model as $x_t = \theta(B)w_t$, where $\theta(B) = 1 + \theta B$. If $|\theta| < 1$, then we can write the model as $\pi(B)x_t = w_t$, where $\pi(B) = \theta^{-1}(B)$. Let $\theta(z) = 1 + \theta z$, for $|z| \leq 1$, then $\pi(z) = \theta^{-1}(z) = 1/(1 + \theta z) = \sum_{j=0}^{\infty} (-\theta)^j z^j$, and we determine that $\pi(B) = \sum_{j=0}^{\infty} (-\theta)^j B^j$.

AUTOREGRESSIVE MOVING AVERAGE MODELS

We now proceed with the general development of autoregressive, moving average, and mixed *autoregressive moving average* (ARMA), models for stationary time series.

Definition 3.5 A time series $\{x_t; t = 0, \pm 1, \pm 2, \dots\}$ is **ARMA**(p, q) if it is stationary and

$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q}, \quad (3.19)$$

with $\phi_p \neq 0$, $\theta_q \neq 0$, and $\sigma_w^2 > 0$. The parameters p and q are called the autoregressive and the moving average orders, respectively. If x_t has a nonzero mean μ , we set $\alpha = \mu(1 - \phi_1 - \dots - \phi_p)$ and write the model as

$$x_t = \alpha + \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \cdots + \theta_q w_{t-q}, \quad (3.20)$$

where $w_t \sim wn(0, \sigma_w^2)$.

As previously noted, when $q = 0$, the model is called an autoregressive model of order p , AR(p), and when $p = 0$, the model is called a moving average model of order q , MA(q). To aid in the investigation of ARMA models, it will be useful to write them using the AR operator, (3.5), and the MA operator, (3.18). In particular, the ARMA(p, q) model in (3.19) can then be written in concise form as

$$\phi(B)x_t = \theta(B)w_t. \quad (3.21)$$

The concise form of the model points to a potential problem in that we can unnecessarily complicate the model by multiplying both sides by another operator, say

$$\eta(B)\phi(B)x_t = \eta(B)\theta(B)w_t,$$

without changing the dynamics. Consider the following example.

Example 3.7 Parameter Redundancy

Consider a white noise process $x_t = w_t$. If we multiply both sides of the equation by $\eta(B) = 1 - .5B$, then the model becomes $(1 - .5B)x_t = (1 - .5B)w_t$, or

$$x_t = .5x_{t-1} - .5w_{t-1} + w_t, \quad (3.22)$$

which looks like an ARMA(1, 1) model. Of course, x_t is still white noise; nothing has changed in this regard [i.e., $x_t = w_t$ is the solution to (3.22)], but we have hidden the fact that x_t is white noise because of the *parameter redundancy* or over-parameterization.

The consideration of parameter redundancy will be crucial when we discuss estimation for general ARMA models. As this example points out, we might fit an ARMA(1, 1) model to white noise data and find that the parameter estimates are significant. If we were unaware of parameter redundancy, we might claim the data are correlated when in fact they are not (Problem 3.20). Although we have not yet discussed estimation, we present the following demonstration of the problem. We generated 150 iid normals and then fit an ARMA(1, 1) to the data. Note that $\hat{\phi} = -.96$ and $\hat{\theta} = .95$, and both are significant. Below is the R code (note that the estimate called ‘intercept’ is really the estimate of the mean).

```
set.seed(8675309)      # Jenny, I got your number
x = rnorm(150, mean=5) # generate iid N(5, 1)s
arima(x, order=c(1,0,1)) # estimation
Coefficients:
            ar1      ma1  intercept<= misnomer
            -0.9595  0.9527   5.0462
            s.e.    0.1688  0.1750   0.0727
```

Thus, forgetting the mean estimate, the fitted model looks like

$$(1 + .96B)x_t = (1 + .95B)w_t,$$

which we should recognize as an over-parametrized model.

Example 3.3, **Example 3.6**, and **Example 3.7** point to a number of problems with the general definition of ARMA(p, q) models, as given by (3.19), or, equivalently, by (3.21). To summarize, we have seen the following problems:

- (i) parameter redundant models,
- (ii) stationary AR models that depend on the future, and
- (iii) MA models that are not unique.

To overcome these problems, we will require some additional restrictions on the model parameters. First, we make the following definitions.

Definition 3.6 *The AR and MA polynomials are defined as*

$$\phi(z) = 1 - \phi_1 z - \cdots - \phi_p z^p, \quad \phi_p \neq 0, \quad (3.23)$$

and

$$\theta(z) = 1 + \theta_1 z + \cdots + \theta_q z^q, \quad \theta_q \neq 0, \quad (3.24)$$

respectively, where z is a complex number.

To address the first problem, we will henceforth refer to an ARMA(p, q) model to mean that it is in its simplest form. That is, in addition to the original definition given in equation (3.19), we will also require that $\phi(z)$ and $\theta(z)$ have no common factors. So, the process, $x_t = .5x_{t-1} - .5w_{t-1} + w_t$, discussed in **Example 3.7** is not referred to as an ARMA(1, 1) process because, in its reduced form, x_t is white noise.

To address the problem of future-dependent models, we formally introduce the concept of *causality*.

Definition 3.7 *An ARMA(p, q) model is said to be causal, if the time series $\{x_t; t = 0, \pm 1, \pm 2, \dots\}$ can be written as a one-sided linear process:*

$$x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j} = \psi(B)w_t, \quad (3.25)$$

where $\psi(B) = \sum_{j=0}^{\infty} \psi_j B^j$, and $\sum_{j=0}^{\infty} |\psi_j| < \infty$; we set $\psi_0 = 1$.

In **Example 3.3**, the AR(1) process, $x_t = \phi x_{t-1} + w_t$, is causal only when $|\phi| < 1$. Equivalently, the process is causal only when the root of $\phi(z) = 1 - \phi z$ is bigger than one in absolute value. That is, the root, say, z_0 , of $\phi(z)$ is $z_0 = 1/\phi$ (because $\phi(z_0) = 0$) and $|z_0| > 1$ because $|\phi| < 1$. In general, we have the following property.

Property 3.1 Causality of an ARMA(p, q) Process

An ARMA(p, q) model is causal if and only if $\phi(z) \neq 0$ for $|z| \leq 1$. The coefficients of the linear process given in (3.25) can be determined by solving

$$\psi(z) = \sum_{j=0}^{\infty} \psi_j z^j = \frac{\theta(z)}{\phi(z)}, \quad |z| \leq 1.$$

Another way to phrase **Property 3.1** is that *an ARMA process is causal only when the roots of $\phi(z)$ lie outside the unit circle*; that is, $\phi(z) = 0$ only when $|z| > 1$. Finally, to address the problem of uniqueness discussed in **Example 3.6**, we choose the model that allows an infinite autoregressive representation.

Definition 3.8 An ARMA(p, q) model is said to be **invertible**, if the time series $\{x_t; t = 0, \pm 1, \pm 2, \dots\}$ can be written as

$$\pi(B)x_t = \sum_{j=0}^{\infty} \pi_j x_{t-j} = w_t, \quad (3.26)$$

where $\pi(B) = \sum_{j=0}^{\infty} \pi_j B^j$, and $\sum_{j=0}^{\infty} |\pi_j| < \infty$; we set $\pi_0 = 1$.

Analogous to **Property 3.1**, we have the following property.

Property 3.2 Invertibility of an ARMA(p, q) Process

An ARMA(p, q) model is invertible if and only if $\theta(z) \neq 0$ for $|z| \leq 1$. The coefficients π_j of $\pi(B)$ given in (3.26) can be determined by solving

$$\pi(z) = \sum_{j=0}^{\infty} \pi_j z^j = \frac{\phi(z)}{\theta(z)}, \quad |z| \leq 1.$$

Another way to phrase **Property 3.2** is that *an ARMA process is invertible only when the roots of $\theta(z)$ lie outside the unit circle*; that is, $\theta(z) = 0$ only when $|z| > 1$. The proof of **Property 3.1** is given in **Section B.2** (the proof of **Property 3.2** is similar). The following examples illustrate these concepts.

Example 3.8 Parameter Redundancy, Causality, Invertibility

Consider the process

$$x_t = .4x_{t-1} + .45x_{t-2} + w_t + w_{t-1} + .25w_{t-2},$$

or, in operator form,

$$(1 - .4B - .45B^2)x_t = (1 + B + .25B^2)w_t.$$

At first, x_t appears to be an ARMA(2, 2) process. But notice that

$$\phi(B) = 1 - .4B - .45B^2 = (1 + .5B)(1 - .9B)$$

and

$$\theta(B) = (1 + B + .25B^2) = (1 + .5B)^2$$

have a common factor that can be canceled. After cancellation, the operators are $\phi(B) = (1 - .9B)$ and $\theta(B) = (1 + .5B)$, so the model is an ARMA(1, 1) model, $(1 - .9B)x_t = (1 + .5B)w_t$, or

$$x_t = .9x_{t-1} + .5w_{t-1} + w_t. \quad (3.27)$$

The model is causal because $\phi(z) = (1 - .9z) = 0$ when $z = 10/9$, which is outside the unit circle. The model is also invertible because the root of $\theta(z) = (1 + .5z)$ is $z = -2$, which is outside the unit circle.

To write the model as a linear process, we can obtain the ψ -weights using **Property 3.1**, $\phi(z)\psi(z) = \theta(z)$, or

$$(1 - .9z)(1 + \psi_1 z + \psi_2 z^2 + \cdots + \psi_j z^j + \cdots) = 1 + .5z.$$

Rearranging, we get

$$1 + (\psi_1 - .9)z + (\psi_2 - .9\psi_1)z^2 + \cdots + (\psi_j - .9\psi_{j-1})z^j + \cdots = 1 + .5z.$$

Matching the coefficients of z on the left and right sides we get $\psi_1 - .9 = .5$ and $\psi_j - .9\psi_{j-1} = 0$ for $j > 1$. Thus, $\psi_j = 1.4(.9)^{j-1}$ for $j \geq 1$ and (3.27) can be written as

$$x_t = w_t + 1.4 \sum_{j=1}^{\infty} .9^{j-1} w_{t-j}.$$

The values of ψ_j may be calculated in R as follows:

```
ARMAToMA(ar = .9, ma = .5, 10) # first 10 psi-weights
[1] 1.40 1.26 1.13 1.02 0.92 0.83 0.74 0.67 0.60 0.54
```

The invertible representation using **Property 3.1** is obtained by matching coefficients in $\theta(z)\pi(z) = \phi(z)$,

$$(1 + .5z)(1 + \pi_1 z + \pi_2 z^2 + \pi_3 z^3 + \cdots) = 1 - .9z.$$

In this case, the π -weights are given by $\pi_j = (-1)^j 1.4 (.5)^{j-1}$, for $j \geq 1$, and hence, because $w_t = \sum_{j=0}^{\infty} \pi_j x_{t-j}$, we can also write (3.27) as

$$x_t = 1.4 \sum_{j=1}^{\infty} (-.5)^{j-1} x_{t-j} + w_t.$$

The values of π_j may be calculated in R as follows by reversing the roles of w_t and x_t ; i.e., write the model as $w_t = -.5w_{t-1} + x_t - .9x_{t-1}$:

```
ARMAToMA(ar = -.5, ma = -.9, 10) # first 10 pi-weights
[1] -1.400 .700 -.350 .175 -.087 .044 -.022 .011 -.006 .003
```

Example 3.9 Causal Conditions for an AR(2) Process

For an AR(1) model, $(1 - \phi B)x_t = w_t$, to be causal, the root of $\phi(z) = 1 - \phi z$ must lie outside of the unit circle. In this case, $\phi(z) = 0$ when $z = 1/\phi$, so it is easy to go from the causal requirement on the root, $|1/\phi| > 1$, to a requirement on the parameter, $|\phi| < 1$. It is not so easy to establish this relationship for higher order models.

For example, the AR(2) model, $(1 - \phi_1 B - \phi_2 B^2)x_t = w_t$, is causal when the two roots of $\phi(z) = 1 - \phi_1 z - \phi_2 z^2$ lie outside of the unit circle. Using the quadratic formula, this requirement can be written as

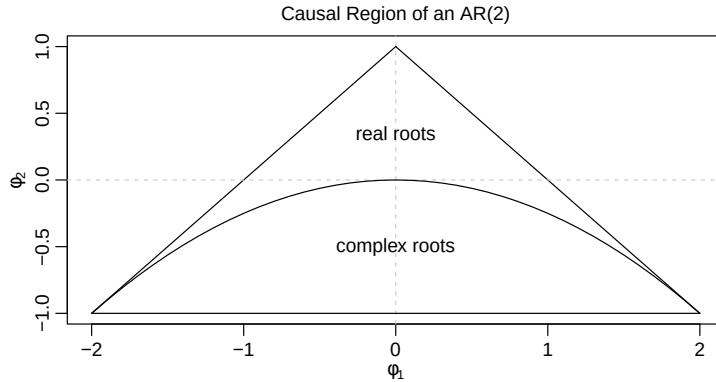


Fig. 3.3. Causal region for an AR(2) in terms of the parameters.

$$\left| \frac{\phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2}}{-2\phi_2} \right| > 1.$$

The roots of $\phi(z)$ may be real and distinct, real and equal, or a complex conjugate pair. If we denote those roots by z_1 and z_2 , we can write $\phi(z) = (1 - z_1^{-1}z)(1 - z_2^{-1}z)$; note that $\phi(z_1) = \phi(z_2) = 0$. The model can be written in operator form as $(1 - z_1^{-1}B)(1 - z_2^{-1}B)x_t = w_t$. From this representation, it follows that $\phi_1 = (z_1^{-1} + z_2^{-1})$ and $\phi_2 = -(z_1 z_2)^{-1}$. This relationship and the fact that $|z_1| > 1$ and $|z_2| > 1$ can be used to establish the following equivalent condition for causality:

$$\phi_1 + \phi_2 < 1, \quad \phi_2 - \phi_1 < 1, \quad \text{and} \quad |\phi_2| < 1. \quad (3.28)$$

This causality condition specifies a triangular region in the parameter space; see Figure 3.3 We leave the details of the equivalence to the reader (Problem 3.5).

3.2 Difference Equations

The study of the behavior of ARMA processes and their ACFs is greatly enhanced by a basic knowledge of difference equations, simply because they are difference equations. We will give a brief and heuristic account of the topic along with some examples of the usefulness of the theory. For details, the reader is referred to Mickens (1990).

Suppose we have a sequence of numbers u_0, u_1, u_2, \dots such that

$$u_n - \alpha u_{n-1} = 0, \quad \alpha \neq 0, \quad n = 1, 2, \dots. \quad (3.29)$$

For example, recall (3.9) in which we showed that the ACF of an AR(1) process is a sequence, $\rho(h)$, satisfying

$$\rho(h) - \phi\rho(h-1) = 0, \quad h = 1, 2, \dots.$$

Equation (3.29) represents a *homogeneous difference equation of order 1*. To solve the equation, we write:

$$\begin{aligned} u_1 &= \alpha u_0 \\ u_2 &= \alpha u_1 = \alpha^2 u_0 \\ &\vdots \\ u_n &= \alpha u_{n-1} = \alpha^n u_0. \end{aligned}$$

Given an initial condition $u_0 = c$, we may solve (3.29), namely, $u_n = \alpha^n c$.

In operator notation, (3.29) can be written as $(1 - \alpha B)u_n = 0$. The polynomial associated with (3.29) is $\alpha(z) = 1 - \alpha z$, and the root, say, z_0 , of this polynomial is $z_0 = 1/\alpha$; that is $\alpha(z_0) = 0$. We know a solution (in fact, *the* solution) to (3.29), with initial condition $u_0 = c$, is

$$u_n = \alpha^n c = (z_0^{-1})^n c. \quad (3.30)$$

That is, the solution to the difference equation (3.29) depends only on the initial condition and the inverse of the root to the associated polynomial $\alpha(z)$.

Now suppose that the sequence satisfies

$$u_n - \alpha_1 u_{n-1} - \alpha_2 u_{n-2} = 0, \quad \alpha_2 \neq 0, \quad n = 2, 3, \dots \quad (3.31)$$

This equation is a *homogeneous difference equation of order 2*. The corresponding polynomial is

$$\alpha(z) = 1 - \alpha_1 z - \alpha_2 z^2,$$

which has two roots, say, z_1 and z_2 ; that is, $\alpha(z_1) = \alpha(z_2) = 0$. We will consider two cases. First suppose $z_1 \neq z_2$. Then the general solution to (3.31) is

$$u_n = c_1 z_1^{-n} + c_2 z_2^{-n}, \quad (3.32)$$

where c_1 and c_2 depend on the initial conditions. The claim it is a solution can be verified by direct substitution of (3.32) into (3.31):

$$\begin{aligned} &\underbrace{(c_1 z_1^{-n} + c_2 z_2^{-n})}_{u_n} - \alpha_1 \underbrace{(c_1 z_1^{-(n-1)} + c_2 z_2^{-(n-1)})}_{u_{n-1}} - \alpha_2 \underbrace{(c_1 z_1^{-(n-2)} + c_2 z_2^{-(n-2)})}_{u_{n-2}} \\ &= c_1 z_1^{-n} \left(1 - \alpha_1 z_1 - \alpha_2 z_1^2\right) + c_2 z_2^{-n} \left(1 - \alpha_1 z_2 - \alpha_2 z_2^2\right) \\ &= c_1 z_1^{-n} \alpha(z_1) + c_2 z_2^{-n} \alpha(z_2) = 0. \end{aligned}$$

Given two initial conditions u_0 and u_1 , we may solve for c_1 and c_2 :

$$u_0 = c_1 + c_2 \quad \text{and} \quad u_1 = c_1 z_1^{-1} + c_2 z_2^{-1},$$

where z_1 and z_2 can be solved for in terms of α_1 and α_2 using the quadratic formula, for example.

When the roots are equal, $z_1 = z_2 (= z_0)$, a general solution to (3.31) is

$$u_n = z_0^{-n}(c_1 + c_2n). \quad (3.33)$$

This claim can also be verified by direct substitution of (3.33) into (3.31):

$$\begin{aligned} & \underbrace{z_0^{-n}(c_1 + c_2n)}_{u_n} - \alpha_1 \underbrace{(z_0^{-(n-1)}[c_1 + c_2(n-1)])}_{u_{n-1}} - \alpha_2 \underbrace{(z_0^{-(n-2)}[c_1 + c_2(n-2)])}_{u_{n-2}} \\ &= z_0^{-n}(c_1 + c_2n) \left(1 - \alpha_1 z_0 - \alpha_2 z_0^2\right) + c_2 z_0^{-n+1} (\alpha_1 + 2\alpha_2 z_0) \\ &= c_2 z_0^{-n+1} (\alpha_1 + 2\alpha_2 z_0). \end{aligned}$$

To show that $(\alpha_1 + 2\alpha_2 z_0) = 0$, write $1 - \alpha_1 z - \alpha_2 z^2 = (1 - z_0^{-1}z)^2$, and take derivatives with respect to z on both sides of the equation to obtain $(\alpha_1 + 2\alpha_2 z) = 2z_0^{-1}(1 - z_0^{-1}z)$. Thus, $(\alpha_1 + 2\alpha_2 z_0) = 2z_0^{-1}(1 - z_0^{-1}z_0) = 0$, as was to be shown. Finally, given two initial conditions, u_0 and u_1 , we can solve for c_1 and c_2 :

$$u_0 = c_1 \quad \text{and} \quad u_1 = (c_1 + c_2)z_0^{-1}.$$

It can also be shown that these solutions are unique.

To summarize these results, in the case of distinct roots, the solution to the homogeneous difference equation of degree two was

$$\begin{aligned} u_n &= z_1^{-n} \times (\text{a polynomial in } n \text{ of degree } m_1 - 1) \\ &\quad + z_2^{-n} \times (\text{a polynomial in } n \text{ of degree } m_2 - 1), \end{aligned} \quad (3.34)$$

where m_1 is the multiplicity of the root z_1 and m_2 is the multiplicity of the root z_2 . In this example, of course, $m_1 = m_2 = 1$, and we called the polynomials of degree zero c_1 and c_2 , respectively. In the case of the repeated root, the solution was

$$u_n = z_0^{-n} \times (\text{a polynomial in } n \text{ of degree } m_0 - 1), \quad (3.35)$$

where m_0 is the multiplicity of the root z_0 ; that is, $m_0 = 2$. In this case, we wrote the polynomial of degree one as $c_1 + c_2n$. In both cases, we solved for c_1 and c_2 given two initial conditions, u_0 and u_1 .

These results generalize to the homogeneous difference equation of order p :

$$u_n - \alpha_1 u_{n-1} - \cdots - \alpha_p u_{n-p} = 0, \quad \alpha_p \neq 0, \quad n = p, p+1, \dots. \quad (3.36)$$

The associated polynomial is $\alpha(z) = 1 - \alpha_1 z - \cdots - \alpha_p z^p$. Suppose $\alpha(z)$ has r distinct roots, z_1 with multiplicity m_1 , z_2 with multiplicity m_2 , \dots , and z_r with multiplicity m_r , such that $m_1 + m_2 + \cdots + m_r = p$. The general solution to the difference equation (3.36) is

$$u_n = z_1^{-n} P_1(n) + z_2^{-n} P_2(n) + \cdots + z_r^{-n} P_r(n), \quad (3.37)$$

where $P_j(n)$, for $j = 1, 2, \dots, r$, is a polynomial in n , of degree $m_j - 1$. Given p initial conditions u_0, \dots, u_{p-1} , we can solve for the $P_j(n)$ explicitly.

Example 3.10 The ACF of an AR(2) Process

Suppose $x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + w_t$ is a causal AR(2) process. Multiply each side of the model by x_{t-h} for $h > 0$, and take expectation:

$$E(x_t x_{t-h}) = \phi_1 E(x_{t-1} x_{t-h}) + \phi_2 E(x_{t-2} x_{t-h}) + E(w_t x_{t-h}).$$

The result is

$$\gamma(h) = \phi_1 \gamma(h-1) + \phi_2 \gamma(h-2), \quad h = 1, 2, \dots. \quad (3.38)$$

In (3.38), we used the fact that $E(x_t) = 0$ and for $h > 0$,

$$E(w_t x_{t-h}) = E\left(w_t \sum_{j=0}^{\infty} \psi_j w_{t-h-j}\right) = 0.$$

Divide (3.38) through by $\gamma(0)$ to obtain the difference equation for the ACF of the process:

$$\rho(h) - \phi_1 \rho(h-1) - \phi_2 \rho(h-2) = 0, \quad h = 1, 2, \dots. \quad (3.39)$$

The initial conditions are $\rho(0) = 1$ and $\rho(-1) = \phi_1/(1 - \phi_2)$, which is obtained by evaluating (3.39) for $h = 1$ and noting that $\rho(1) = \rho(-1)$.

Using the results for the homogeneous difference equation of order two, let z_1 and z_2 be the roots of the associated polynomial, $\phi(z) = 1 - \phi_1 z - \phi_2 z^2$. Because the model is causal, we know the roots are outside the unit circle: $|z_1| > 1$ and $|z_2| > 1$. Now, consider the solution for three cases:

(i) When z_1 and z_2 are real and distinct, then

$$\rho(h) = c_1 z_1^{-h} + c_2 z_2^{-h},$$

so $\rho(h) \rightarrow 0$ exponentially fast as $h \rightarrow \infty$.

(ii) When $z_1 = z_2 (= z_0)$ are real and equal, then

$$\rho(h) = z_0^{-h}(c_1 + c_2 h),$$

so $\rho(h) \rightarrow 0$ exponentially fast as $h \rightarrow \infty$.

(iii) When $z_1 = \bar{z}_2$ are a complex conjugate pair, then $c_2 = \bar{c}_1$ (because $\rho(h)$ is real), and

$$\rho(h) = c_1 z_1^{-h} + \bar{c}_1 \bar{z}_1^{-h}.$$

Write c_1 and z_1 in polar coordinates, for example, $z_1 = |z_1| e^{i\theta}$, where θ is the angle whose tangent is the ratio of the imaginary part and the real part of z_1 (sometimes called $\arg(z_1)$; the range of θ is $[-\pi, \pi]$). Then, using the fact that $e^{i\alpha} + e^{-i\alpha} = 2 \cos(\alpha)$, the solution has the form

$$\rho(h) = a |z_1|^{-h} \cos(h\theta + b),$$

where a and b are determined by the initial conditions. Again, $\rho(h)$ dampens to zero exponentially fast as $h \rightarrow \infty$, but it does so in a sinusoidal fashion. The implication of this result is shown in the next example.

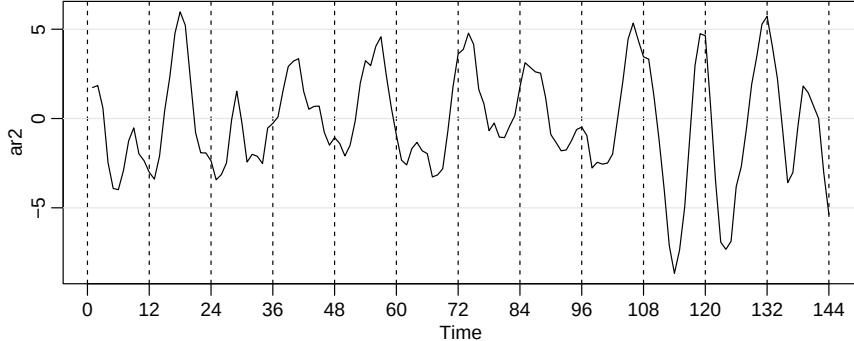


Fig. 3.4. Simulated AR(2) model, $n = 144$ with $\phi_1 = 1.5$ and $\phi_2 = -.75$.

Example 3.11 An AR(2) with Complex Roots

Figure 3.4 shows $n = 144$ observations from the AR(2) model

$$x_t = 1.5x_{t-1} - .75x_{t-2} + w_t,$$

with $\sigma_w^2 = 1$, and with complex roots chosen so the process exhibits pseudo-cyclic behavior at the rate of one cycle every 12 time points. The autoregressive polynomial for this model is $\phi(z) = 1 - 1.5z + .75z^2$. The roots of $\phi(z)$ are $1 \pm i/\sqrt{3}$, and $\theta = \tan^{-1}(1/\sqrt{3}) = 2\pi/12$ radians per unit time. To convert the angle to cycles per unit time, divide by 2π to get 1/12 cycles per unit time. The ACF for this model is shown in left-hand-side of Figure 3.5.

To calculate the roots of the polynomial and solve for arg in R:

```

z = c(1, -1.5, .75)      # coefficients of the polynomial
(a = polyroot(z)[1])    # print one root = 1 + i/sqrt(3)
[1] 1+0.57735i
arg = Arg(a)/(2*pi)     # arg in cycles/pt
1/arg                   # the pseudo period
[1] 12

```

To reproduce Figure 3.4:

```

set.seed(8675309)
ar2 = arima.sim(list(order=c(2,0,0), ar=c(1.5,-.75)), n = 144)
plot(ar2, axes=FALSE, xlab="Time")
axis(2); axis(1, at=seq(0,144,by=12)); box()
abline(v=seq(0,144,by=12), lty=2)

```

To calculate and display the ACF for this model:

```

ACF = ARMAacf(ar=c(1.5,-.75), ma=0, 50)
plot(ACF, type="h", xlab="lag")
abline(h=0)

```

Example 3.12 The ψ -weights for an ARMA Model

For a causal ARMA(p, q) model, $\phi(B)x_t = \theta(B)w_t$, where the zeros of $\phi(z)$ are outside the unit circle, recall that we may write

$$x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j},$$

where the ψ -weights are determined using [Property 3.1](#).

For the pure MA(q) model, $\psi_0 = 1$, $\psi_j = \theta_j$, for $j = 1, \dots, q$, and $\psi_j = 0$, otherwise. For the general case of ARMA(p, q) models, the task of solving for the ψ -weights is much more complicated, as was demonstrated in [Example 3.8](#). The use of the theory of homogeneous difference equations can help here. To solve for the ψ -weights in general, we must match the coefficients in $\phi(z)\psi(z) = \theta(z)$:

$$(1 - \phi_1 z - \phi_2 z^2 - \dots)(\psi_0 + \psi_1 z + \psi_2 z^2 + \dots) = (1 + \theta_1 z + \theta_2 z^2 + \dots).$$

The first few values are

$$\begin{aligned} \psi_0 &= 1 \\ \psi_1 - \phi_1 \psi_0 &= \theta_1 \\ \psi_2 - \phi_1 \psi_1 - \phi_2 \psi_0 &= \theta_2 \\ \psi_3 - \phi_1 \psi_2 - \phi_2 \psi_1 - \phi_3 \psi_0 &= \theta_3 \\ &\vdots \end{aligned}$$

where we would take $\phi_j = 0$ for $j > p$, and $\theta_j = 0$ for $j > q$. The ψ -weights satisfy the homogeneous difference equation given by

$$\psi_j - \sum_{k=1}^p \phi_k \psi_{j-k} = 0, \quad j \geq \max(p, q+1), \quad (3.40)$$

with initial conditions

$$\psi_j - \sum_{k=1}^j \phi_k \psi_{j-k} = \theta_j, \quad 0 \leq j < \max(p, q+1). \quad (3.41)$$

The general solution depends on the roots of the AR polynomial $\phi(z) = 1 - \phi_1 z - \dots - \phi_p z^p$, as seen from [\(3.40\)](#). The specific solution will, of course, depend on the initial conditions.

Consider the ARMA process given in [\(3.27\)](#), $x_t = .9x_{t-1} + .5w_{t-1} + w_t$. Because $\max(p, q+1) = 2$, using [\(3.41\)](#), we have $\psi_0 = 1$ and $\psi_1 = .9 + .5 = 1.4$. By [\(3.40\)](#), for $j = 2, 3, \dots$, the ψ -weights satisfy $\psi_j - .9\psi_{j-1} = 0$. The general solution is $\psi_j = c \cdot .9^j$. To find the specific solution, use the initial condition $\psi_1 = 1.4$, so $1.4 = .9c$ or $c = 1.4/.9$. Finally, $\psi_j = 1.4(.9)^{j-1}$, for $j \geq 1$, as we saw in [Example 3.8](#).

To view, for example, the first 50 ψ -weights in R, use:

```
ARMAtoMA(ar=.9, ma=.5, 50)      # for a list
plot(ARMAtoMA(ar=.9, ma=.5, 50)) # for a graph
```

3.3 Autocorrelation and Partial Autocorrelation

We begin by exhibiting the ACF of an MA(q) process, $x_t = \theta(B)w_t$, where $\theta(B) = 1 + \theta_1B + \dots + \theta_qB^q$. Because x_t is a finite linear combination of white noise terms, the process is stationary with mean

$$\mathbb{E}(x_t) = \sum_{j=0}^q \theta_j \mathbb{E}(w_{t-j}) = 0,$$

where we have written $\theta_0 = 1$, and with autocovariance function

$$\begin{aligned} \gamma(h) &= \text{cov}(x_{t+h}, x_t) = \text{cov}\left(\sum_{j=0}^q \theta_j w_{t+h-j}, \sum_{k=0}^q \theta_k w_{t-k}\right) \\ &= \begin{cases} \sigma_w^2 \sum_{j=0}^{q-h} \theta_j \theta_{j+h}, & 0 \leq h \leq q \\ 0 & h > q. \end{cases} \end{aligned} \quad (3.42)$$

Recall that $\gamma(h) = \gamma(-h)$, so we will only display the values for $h \geq 0$. Note that $\gamma(q)$ cannot be zero because $\theta_q \neq 0$. The cutting off of $\gamma(h)$ after q lags is the signature of the MA(q) model. Dividing (3.42) by $\gamma(0)$ yields the *ACF of an MA(q)*:

$$\rho(h) = \begin{cases} \frac{\sum_{j=0}^{q-h} \theta_j \theta_{j+h}}{1 + \theta_1^2 + \dots + \theta_q^2} & 1 \leq h \leq q \\ 0 & h > q. \end{cases} \quad (3.43)$$

For a causal ARMA(p, q) model, $\phi(B)x_t = \theta(B)w_t$, where the zeros of $\phi(z)$ are outside the unit circle, write

$$x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j}. \quad (3.44)$$

It follows immediately that $\mathbb{E}(x_t) = 0$ and the autocovariance function of x_t is

$$\gamma(h) = \text{cov}(x_{t+h}, x_t) = \sigma_w^2 \sum_{j=0}^{\infty} \psi_j \psi_{j+h}, \quad h \geq 0. \quad (3.45)$$

We could then use (3.40) and (3.41) to solve for the ψ -weights. In turn, we could solve for $\gamma(h)$, and the ACF $\rho(h) = \gamma(h)/\gamma(0)$. As in Example 3.10, it is also possible to obtain a homogeneous difference equation directly in terms of $\gamma(h)$. First, we write

$$\begin{aligned} \gamma(h) &= \text{cov}(x_{t+h}, x_t) = \text{cov}\left(\sum_{j=1}^p \phi_j x_{t+h-j} + \sum_{j=0}^q \theta_j w_{t+h-j}, x_t\right) \\ &= \sum_{j=1}^p \phi_j \gamma(h-j) + \sigma_w^2 \sum_{j=h}^q \theta_j \psi_{j-h}, \quad h \geq 0, \end{aligned} \quad (3.46)$$

where we have used the fact that, for $h \geq 0$,

$$\text{cov}(w_{t+h-j}, x_t) = \text{cov}\left(w_{t+h-j}, \sum_{k=0}^{\infty} \psi_k w_{t-k}\right) = \psi_{j-h} \sigma_w^2.$$

From (3.46), we can write a *general homogeneous equation for the ACF of a causal ARMA process*:

$$\gamma(h) - \phi_1 \gamma(h-1) - \cdots - \phi_p \gamma(h-p) = 0, \quad h \geq \max(p, q+1), \quad (3.47)$$

with initial conditions

$$\gamma(h) - \sum_{j=1}^p \phi_j \gamma(h-j) = \sigma_w^2 \sum_{j=h}^q \theta_j \psi_{j-h}, \quad 0 \leq h < \max(p, q+1). \quad (3.48)$$

Dividing (3.47) and (3.48) through by $\gamma(0)$ will allow us to solve for the ACF, $\rho(h) = \gamma(h)/\gamma(0)$.

Example 3.13 The ACF of an AR(p)

In Example 3.10 we considered the case where $p = 2$. For the general case, it follows immediately from (3.47) that

$$\rho(h) - \phi_1 \rho(h-1) - \cdots - \phi_p \rho(h-p) = 0, \quad h \geq p. \quad (3.49)$$

Let z_1, \dots, z_r denote the roots of $\phi(z)$, each with multiplicity m_1, \dots, m_r , respectively, where $m_1 + \cdots + m_r = p$. Then, from (3.37), the general solution is

$$\rho(h) = z_1^{-h} P_1(h) + z_2^{-h} P_2(h) + \cdots + z_r^{-h} P_r(h), \quad h \geq p, \quad (3.50)$$

where $P_j(h)$ is a polynomial in h of degree $m_j - 1$.

Recall that for a causal model, all of the roots are outside the unit circle, $|z_i| > 1$, for $i = 1, \dots, r$. If all the roots are real, then $\rho(h)$ dampens exponentially fast to zero as $h \rightarrow \infty$. If some of the roots are complex, then they will be in conjugate pairs and $\rho(h)$ will dampen, in a sinusoidal fashion, exponentially fast to zero as $h \rightarrow \infty$. In the case of complex roots, the time series will appear to be cyclic in nature. This, of course, is also true for ARMA models in which the AR part has complex roots.

Example 3.14 The ACF of an ARMA(1, 1)

Consider the ARMA(1, 1) process $x_t = \phi x_{t-1} + \theta w_{t-1} + w_t$, where $|\phi| < 1$. Based on (3.47), the autocovariance function satisfies

$$\gamma(h) - \phi \gamma(h-1) = 0, \quad h = 2, 3, \dots,$$

and it follows from (3.29)–(3.30) that the general solution is

$$\gamma(h) = c \phi^h, \quad h = 1, 2, \dots. \quad (3.51)$$

To obtain the initial conditions, we use (3.48):

$$\gamma(0) = \phi\gamma(1) + \sigma_w^2[1 + \theta\phi + \theta^2] \quad \text{and} \quad \gamma(1) = \phi\gamma(0) + \sigma_w^2\theta.$$

Solving for $\gamma(0)$ and $\gamma(1)$, we obtain:

$$\gamma(0) = \sigma_w^2 \frac{1 + 2\theta\phi + \theta^2}{1 - \phi^2} \quad \text{and} \quad \gamma(1) = \sigma_w^2 \frac{(1 + \theta\phi)(\phi + \theta)}{1 - \phi^2}.$$

To solve for c , note that from (3.51), $\gamma(1) = c\phi$ or $c = \gamma(1)/\phi$. Hence, the specific solution for $h \geq 1$ is

$$\gamma(h) = \frac{\gamma(1)}{\phi} \phi^h = \sigma_w^2 \frac{(1 + \theta\phi)(\phi + \theta)}{1 - \phi^2} \phi^{h-1}.$$

Finally, dividing through by $\gamma(0)$ yields the ACF

$$\rho(h) = \frac{(1 + \theta\phi)(\phi + \theta)}{1 + 2\theta\phi + \theta^2} \phi^{h-1}, \quad h \geq 1. \quad (3.52)$$

Notice that the general pattern of $\rho(h)$ versus h in (3.52) is not different from that of an AR(1) given in (3.8). Hence, it is unlikely that we will be able to tell the difference between an ARMA(1,1) and an AR(1) based solely on an ACF estimated from a sample. This consideration will lead us to the partial autocorrelation function.

THE PARTIAL AUTOCORRELATION FUNCTION (PACF)

We have seen in (3.43), for MA(q) models, the ACF will be zero for lags greater than q . Moreover, because $\theta_q \neq 0$, the ACF will not be zero at lag q . Thus, the ACF provides a considerable amount of information about the order of the dependence when the process is a moving average process. If the process, however, is ARMA or AR, the ACF alone tells us little about the orders of dependence. Hence, it is worthwhile pursuing a function that will behave like the ACF of MA models, but for AR models, namely, the *partial autocorrelation function (PACF)*.

Recall that if X , Y , and Z are random variables, then the partial correlation between X and Y given Z is obtained by regressing X on Z to obtain \hat{X} , regressing Y on Z to obtain \hat{Y} , and then calculating

$$\rho_{XY|Z} = \text{corr}\{X - \hat{X}, Y - \hat{Y}\}.$$

The idea is that $\rho_{XY|Z}$ measures the correlation between X and Y with the linear effect of Z removed (or partialled out). If the variables are multivariate normal, then this definition coincides with $\rho_{XY|Z} = \text{corr}(X, Y | Z)$.

To motivate the idea for time series, consider a causal AR(1) model, $x_t = \phi x_{t-1} + w_t$. Then,

$$\begin{aligned} \gamma_x(2) &= \text{cov}(x_t, x_{t-2}) = \text{cov}(\phi x_{t-1} + w_t, x_{t-2}) \\ &= \text{cov}(\phi^2 x_{t-2} + \phi w_{t-1} + w_t, x_{t-2}) = \phi^2 \gamma_x(0). \end{aligned}$$

This result follows from causality because x_{t-2} involves $\{w_{t-2}, w_{t-3}, \dots\}$, which are all uncorrelated with w_t and w_{t-1} . The correlation between x_t and x_{t-2} is not zero, as it would be for an MA(1), because x_t is dependent on x_{t-2} through x_{t-1} . Suppose we break this chain of dependence by removing (or partial out) the effect x_{t-1} . That is, we consider the correlation between $x_t - \phi x_{t-1}$ and $x_{t-2} - \phi x_{t-1}$, because it is the correlation between x_t and x_{t-2} with the linear dependence of each on x_{t-1} removed. In this way, we have broken the dependence chain between x_t and x_{t-2} . In fact,

$$\text{cov}(x_t - \phi x_{t-1}, x_{t-2} - \phi x_{t-1}) = \text{cov}(w_t, x_{t-2} - \phi x_{t-1}) = 0.$$

Hence, the tool we need is partial autocorrelation, which is the correlation between x_s and x_t with the linear effect of everything “in the middle” removed.

To formally define the PACF for mean-zero stationary time series, let \hat{x}_{t+h} , for $h \geq 2$, denote the regression^{3.3} of x_{t+h} on $\{x_{t+h-1}, x_{t+h-2}, \dots, x_{t+1}\}$, which we write as

$$\hat{x}_{t+h} = \beta_1 x_{t+h-1} + \beta_2 x_{t+h-2} + \dots + \beta_{h-1} x_{t+1}. \quad (3.53)$$

No intercept term is needed in (3.53) because the mean of x_t is zero (otherwise, replace x_t by $x_t - \mu_x$ in this discussion). In addition, let \hat{x}_t denote the regression of x_t on $\{x_{t+1}, x_{t+2}, \dots, x_{t+h-1}\}$, then

$$\hat{x}_t = \beta_1 x_{t+1} + \beta_2 x_{t+2} + \dots + \beta_{h-1} x_{t+h-1}. \quad (3.54)$$

Because of stationarity, the coefficients, $\beta_1, \dots, \beta_{h-1}$ are the same in (3.53) and (3.54); we will explain this result in the next section, but it will be evident from the examples.

Definition 3.9 *The partial autocorrelation function (PACF) of a stationary process, x_t , denoted ϕ_{hh} , for $h = 1, 2, \dots$, is*

$$\phi_{11} = \text{corr}(x_{t+1}, x_t) = \rho(1) \quad (3.55)$$

and

$$\phi_{hh} = \text{corr}(x_{t+h} - \hat{x}_{t+h}, x_t - \hat{x}_t), \quad h \geq 2. \quad (3.56)$$

The reason for using a double subscript will become evident in the next section. The PACF, ϕ_{hh} , is the correlation between x_{t+h} and x_t with the linear dependence of $\{x_{t+1}, \dots, x_{t+h-1}\}$ on each, removed. If the process x_t is Gaussian, then $\phi_{hh} = \text{corr}(x_{t+h}, x_t | x_{t+1}, \dots, x_{t+h-1})$; that is, ϕ_{hh} is the correlation coefficient between x_{t+h} and x_t in the bivariate distribution of (x_{t+h}, x_t) conditional on $\{x_{t+1}, \dots, x_{t+h-1}\}$.

^{3.3} The term regression here refers to regression in the population sense. That is, \hat{x}_{t+h} is the linear combination of $\{x_{t+h-1}, x_{t+h-2}, \dots, x_{t+1}\}$ that minimizes the mean squared error $E(x_{t+h} - \sum_{j=1}^{h-1} \alpha_j x_{t+j})^2$.

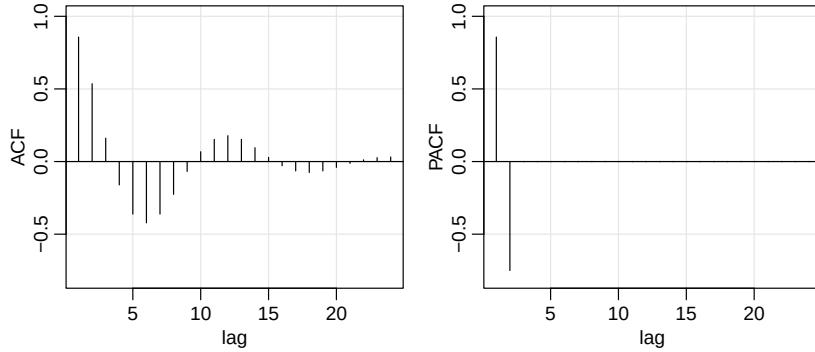


Fig. 3.5. The ACF and PACF of an AR(2) model with $\phi_1 = 1.5$ and $\phi_2 = -.75$.

Example 3.15 The PACF of an AR(1)

Consider the PACF of the AR(1) process given by $x_t = \phi x_{t-1} + w_t$, with $|\phi| < 1$. By definition, $\phi_{11} = \rho(1) = \phi$. To calculate ϕ_{22} , consider the regression of x_{t+2} on x_{t+1} , say, $\hat{x}_{t+2} = \beta x_{t+1}$. We choose β to minimize

$$E(x_{t+2} - \hat{x}_{t+2})^2 = E(x_{t+2} - \beta x_{t+1})^2 = \gamma(0) - 2\beta\gamma(1) + \beta^2\gamma(0).$$

Taking derivatives with respect to β and setting the result equal to zero, we have $\beta = \gamma(1)/\gamma(0) = \rho(1) = \phi$. Next, consider the regression of x_t on x_{t+1} , say $\hat{x}_t = \beta x_{t+1}$. We choose β to minimize

$$E(x_t - \hat{x}_t)^2 = E(x_t - \beta x_{t+1})^2 = \gamma(0) - 2\beta\gamma(1) + \beta^2\gamma(0).$$

This is the same equation as before, so $\beta = \phi$. Hence,

$$\begin{aligned}\phi_{22} &= \text{corr}(x_{t+2} - \hat{x}_{t+2}, x_t - \hat{x}_t) = \text{corr}(x_{t+2} - \phi x_{t+1}, x_t - \phi x_{t+1}) \\ &= \text{corr}(w_{t+2}, x_t - \phi x_{t+1}) = 0\end{aligned}$$

by causality. Thus, $\phi_{22} = 0$. In the next example, we will see that in this case, $\phi_{hh} = 0$ for all $h > 1$.

Example 3.16 The PACF of an AR(p)

The model implies $x_{t+h} = \sum_{j=1}^p \phi_j x_{t+h-j} + w_{t+h}$, where the roots of $\phi(z)$ are outside the unit circle. When $h > p$, the regression of x_{t+h} on $\{x_{t+1}, \dots, x_{t+h-1}\}$, is

$$\hat{x}_{t+h} = \sum_{j=1}^p \phi_j x_{t+h-j}.$$

We have not proved this obvious result yet, but we will prove it in the next section. Thus, when $h > p$,

Table 3.1. Behavior of the ACF and PACF for ARMA Models

	AR(p)	MA(q)	ARMA(p, q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

$$\phi_{hh} = \text{corr}(x_{t+h} - \hat{x}_{t+h}, x_t - \hat{x}_t) = \text{corr}(w_{t+h}, x_t - \hat{x}_t) = 0,$$

because, by causality, $x_t - \hat{x}_t$ depends only on $\{w_{t+h-1}, w_{t+h-2}, \dots\}$; recall equation (3.54). When $h \leq p$, ϕ_{pp} is not zero, and $\phi_{11}, \dots, \phi_{p-1,p-1}$ are not necessarily zero. We will see later that, in fact, $\phi_{pp} = \phi_p$. Figure 3.5 shows the ACF and the PACF of the AR(2) model presented in Example 3.11. To reproduce Figure 3.5 in R, use the following commands:

```
ACF = ARMAacf(ar=c(1.5,-.75), ma=0, 24)[-1]
PACF = ARMAacf(ar=c(1.5,-.75), ma=0, 24, pacf=TRUE)
par(mfrow=c(1,2))
plot(ACF, type="h", xlab="lag", ylim=c(-.8,1)); abline(h=0)
plot(PACF, type="h", xlab="lag", ylim=c(-.8,1)); abline(h=0)
```

Example 3.17 The PACF of an Invertible MA(q)

For an invertible MA(q), we can write $x_t = -\sum_{j=1}^{\infty} \pi_j x_{t-j} + w_t$. Moreover, no finite representation exists. From this result, it should be apparent that the PACF will never cut off, as in the case of an AR(p).

For an MA(1), $x_t = w_t + \theta w_{t-1}$, with $|\theta| < 1$, calculations similar to Example 3.15 will yield $\phi_{22} = -\theta^2/(1 + \theta^2 + \theta^4)$. For the MA(1) in general, we can show that

$$\phi_{hh} = -\frac{(-\theta)^h(1 - \theta^2)}{1 - \theta^{2(h+1)}}, \quad h \geq 1.$$

In the next section, we will discuss methods of calculating the PACF. The PACF for MA models behaves much like the ACF for AR models. Also, the PACF for AR models behaves much like the ACF for MA models. Because an invertible ARMA model has an infinite AR representation, the PACF will not cut off. We may summarize these results in Table 3.1.

Example 3.18 Preliminary Analysis of the Recruitment Series

We consider the problem of modeling the Recruitment series shown in Figure 1.5. There are 453 months of observed recruitment ranging over the years 1950–1987. The ACF and the PACF given in Figure 3.6 are consistent with the behavior of an AR(2). The ACF has cycles corresponding roughly to a 12-month period, and the PACF has large values for $h = 1, 2$ and then is essentially zero for higher order lags. Based on Table 3.1, these results suggest that a second-order ($p = 2$) autoregressive model might provide a good fit. Although we will discuss estimation

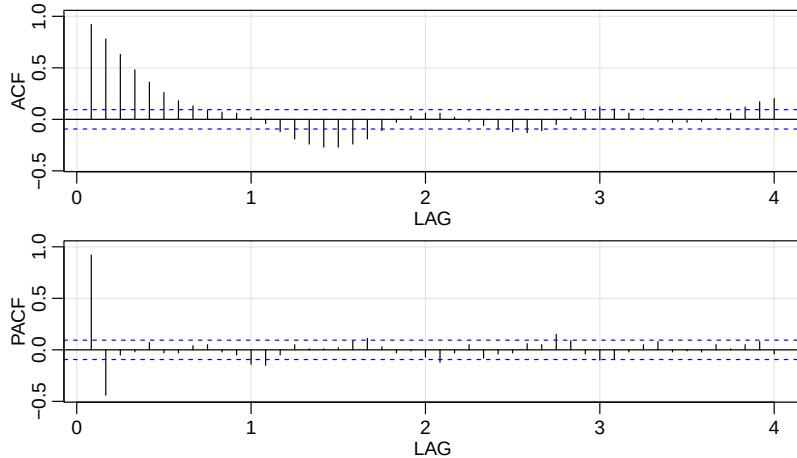


Fig. 3.6. ACF and PACF of the Recruitment series. Note that the lag axes are in terms of season (12 months in this case).

in detail in [Section 3.5](#), we ran a regression (see [Section 2.1](#)) using the data triplets $\{(x; z_1, z_2) : (x_3; x_2, x_1), (x_4; x_3, x_2), \dots, (x_{453}; x_{452}, x_{451})\}$ to fit a model of the form

$$x_t = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + w_t$$

for $t = 3, 4, \dots, 453$. The estimates and standard errors (in parentheses) are $\hat{\phi}_0 = 6.74_{(1.11)}$, $\hat{\phi}_1 = 1.35_{(.04)}$, $\hat{\phi}_2 = -.46_{(.04)}$, and $\hat{\sigma}_w^2 = 89.72$.

The following R code can be used for this analysis. We use `acf2` from `astsa` to print and plot the ACF and PACF.

```
acf2(rec, 48)      # will produce values and a graphic
(regr = ar.ols(rec, order=2, demean=FALSE, intercept=TRUE))
regr$asy.se.coef # standard errors of the estimates
```

3.4 Forecasting

In forecasting, the goal is to predict future values of a time series, x_{n+m} , $m = 1, 2, \dots$, based on the data collected to the present, $x_{1:n} = \{x_1, x_2, \dots, x_n\}$. Throughout this section, we will assume x_t is stationary and the model parameters are known. The problem of forecasting when the model parameters are unknown will be discussed in the next section; also, see [Problem 3.26](#). The minimum mean square error predictor of x_{n+m} is

$$x_{n+m}^n = E(x_{n+m} | x_{1:n}) \quad (3.57)$$

because the conditional expectation minimizes the mean square error

$$E [x_{n+m} - g(x_{1:n})]^2, \quad (3.58)$$

where $g(x_{1:n})$ is a function of the observations $x_{1:n}$; see [Problem 3.14](#).

First, we will restrict attention to predictors that are linear functions of the data, that is, predictors of the form

$$x_{n+m}^n = \alpha_0 + \sum_{k=1}^n \alpha_k x_k, \quad (3.59)$$

where $\alpha_0, \alpha_1, \dots, \alpha_n$ are real numbers. We note that the α s depend on n and m , but for now we drop the dependence from the notation. For example, if $n = m = 1$, then x_2^1 is the one-step-ahead linear forecast of x_2 given x_1 . In terms of (3.59), $x_2^1 = \alpha_0 + \alpha_1 x_1$. But if $n = 2$, x_3^2 is the one-step-ahead linear forecast of x_3 given x_1 and x_2 . In terms of (3.59), $x_3^2 = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2$, and in general, the α s in x_2^1 and x_3^2 will be different.

Linear predictors of the form (3.59) that minimize the mean square prediction error (3.58) are called *best linear predictors* (BLPs). As we shall see, linear prediction depends only on the second-order moments of the process, which are easy to estimate from the data. Much of the material in this section is enhanced by the theoretical material presented in [Appendix B](#). For example, [Theorem B.3](#) states that if the process is Gaussian, minimum mean square error predictors and best linear predictors are the same. The following property, which is based on the Projection Theorem, [Theorem B.1](#), is a key result.

Property 3.3 Best Linear Prediction for Stationary Processes

Given data x_1, \dots, x_n , the best linear predictor, $x_{n+m}^n = \alpha_0 + \sum_{k=1}^n \alpha_k x_k$, of x_{n+m} , for $m \geq 1$, is found by solving

$$E[(x_{n+m} - x_{n+m}^n) x_k] = 0, \quad k = 0, 1, \dots, n, \quad (3.60)$$

where $x_0 = 1$, for $\alpha_0, \alpha_1, \dots, \alpha_n$.

The equations specified in (3.60) are called the *prediction equations*, and they are used to solve for the coefficients $\{\alpha_0, \alpha_1, \dots, \alpha_n\}$. The results of [Property 3.3](#) can also be obtained via least squares; i.e., to minimize $Q = E(x_{n+m} - \sum_{k=0}^n \alpha_k x_k)^2$ with respect to the α s, solve $\partial Q / \partial \alpha_j = 0$ for the α_j , $j = 0, 1, \dots, n$. This leads to (3.60).

If $E(x_t) = \mu$, the first equation ($k = 0$) of (3.60) implies

$$E(x_{n+m}^n) = E(x_{n+m}) = \mu.$$

Thus, taking expectation in (3.59), we have

$$\mu = \alpha_0 + \sum_{k=1}^n \alpha_k \mu \quad \text{or} \quad \alpha_0 = \mu \left(1 - \sum_{k=1}^n \alpha_k\right).$$

Hence, the form of the BLP is

$$x_{n+m}^n = \mu + \sum_{k=1}^n \alpha_k (x_k - \mu).$$

Thus, until we discuss estimation, there is no loss of generality in considering the case that $\mu = 0$, in which case, $\alpha_0 = 0$.

First, consider *one-step-ahead prediction*. That is, given $\{x_1, \dots, x_n\}$, we wish to forecast the value of the time series at the next time point, x_{n+1} . The BLP of x_{n+1} is of the form

$$x_{n+1}^n = \phi_{n1}x_n + \phi_{n2}x_{n-1} + \dots + \phi_{nn}x_1, \quad (3.61)$$

where we now display the dependence of the coefficients on n ; in this case, α_k in (3.59) is $\phi_{n,n+1-k}$ in (3.61), for $k = 1, \dots, n$. Using [Property 3.3](#), the coefficients $\{\phi_{n1}, \phi_{n2}, \dots, \phi_{nn}\}$ satisfy

$$\mathbb{E}\left[\left(x_{n+1} - \sum_{j=1}^n \phi_{nj}x_{n+1-j}\right)x_{n+1-k}\right] = 0, \quad k = 1, \dots, n,$$

or

$$\sum_{j=1}^n \phi_{nj}\gamma(k-j) = \gamma(k), \quad k = 1, \dots, n. \quad (3.62)$$

The prediction equations (3.62) can be written in matrix notation as

$$\Gamma_n \phi_n = \gamma_n, \quad (3.63)$$

where $\Gamma_n = \{\gamma(k-j)\}_{j,k=1}^n$ is an $n \times n$ matrix, $\phi_n = (\phi_{n1}, \dots, \phi_{nn})'$ is an $n \times 1$ vector, and $\gamma_n = (\gamma(1), \dots, \gamma(n))'$ is an $n \times 1$ vector.

The matrix Γ_n is nonnegative definite. If Γ_n is singular, there are many solutions to (3.63), but, by the Projection Theorem ([Theorem B.1](#)), x_{n+1}^n is unique. If Γ_n is nonsingular, the elements of ϕ_n are unique, and are given by

$$\phi_n = \Gamma_n^{-1} \gamma_n. \quad (3.64)$$

For ARMA models, the fact that $\sigma_w^2 > 0$ and $\gamma(h) \rightarrow 0$ as $h \rightarrow \infty$ is enough to ensure that Γ_n is positive definite ([Problem 3.12](#)). It is sometimes convenient to write the one-step-ahead forecast in vector notation

$$x_{n+1}^n = \phi_n' x, \quad (3.65)$$

where $x = (x_n, x_{n-1}, \dots, x_1)'$.

The *mean square one-step-ahead prediction error* is

$$P_{n+1}^n = \mathbb{E}(x_{n+1} - x_{n+1}^n)^2 = \gamma(0) - \gamma_n' \Gamma_n^{-1} \gamma_n. \quad (3.66)$$

To verify (3.66) using (3.64) and (3.65),

$$\begin{aligned} \mathbb{E}(x_{n+1} - x_{n+1}^n)^2 &= \mathbb{E}(x_{n+1} - \phi_n' x)^2 = \mathbb{E}(x_{n+1} - \gamma_n' \Gamma_n^{-1} x)^2 \\ &= \mathbb{E}(x_{n+1}^2 - 2\gamma_n' \Gamma_n^{-1} x x_{n+1} + \gamma_n' \Gamma_n^{-1} x x' \Gamma_n^{-1} \gamma_n) \\ &= \gamma(0) - 2\gamma_n' \Gamma_n^{-1} \gamma_n + \gamma_n' \Gamma_n^{-1} \Gamma_n \Gamma_n^{-1} \gamma_n \\ &= \gamma(0) - \gamma_n' \Gamma_n^{-1} \gamma_n. \end{aligned}$$

Example 3.19 Prediction for an AR(2)

Suppose we have a causal AR(2) process $x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + w_t$, and one observation x_1 . Then, using equation (3.64), the one-step-ahead prediction of x_2 based on x_1 is

$$x_2^1 = \phi_{11} x_1 = \frac{\gamma(1)}{\gamma(0)} x_1 = \rho(1) x_1.$$

Now, suppose we want the one-step-ahead prediction of x_3 based on two observations x_1 and x_2 ; i.e., $x_3^2 = \phi_{21} x_2 + \phi_{22} x_1$. We could use (3.62)

$$\begin{aligned}\phi_{21}\gamma(0) + \phi_{22}\gamma(1) &= \gamma(1) \\ \phi_{21}\gamma(1) + \phi_{22}\gamma(0) &= \gamma(2)\end{aligned}$$

to solve for ϕ_{21} and ϕ_{22} , or use the matrix form in (3.64) and solve

$$\begin{pmatrix} \phi_{21} \\ \phi_{22} \end{pmatrix} = \begin{pmatrix} \gamma(0) & \gamma(1) \\ \gamma(1) & \gamma(0) \end{pmatrix}^{-1} \begin{pmatrix} \gamma(1) \\ \gamma(2) \end{pmatrix},$$

but, it should be apparent from the model that $x_3^2 = \phi_1 x_2 + \phi_2 x_1$. Because $\phi_1 x_2 + \phi_2 x_1$ satisfies the prediction equations (3.60),

$$E\{[x_3 - (\phi_1 x_2 + \phi_2 x_1)]x_1\} = E(w_3 x_1) = 0,$$

$$E\{[x_3 - (\phi_1 x_2 + \phi_2 x_1)]x_2\} = E(w_3 x_2) = 0,$$

it follows that, indeed, $x_3^2 = \phi_1 x_2 + \phi_2 x_1$, and by the uniqueness of the coefficients in this case, that $\phi_{21} = \phi_1$ and $\phi_{22} = \phi_2$. Continuing in this way, it is easy to verify that, for $n \geq 2$,

$$x_{n+1}^n = \phi_1 x_n + \phi_2 x_{n-1}.$$

That is, $\phi_{n1} = \phi_1$, $\phi_{n2} = \phi_2$, and $\phi_{nj} = 0$, for $j = 3, 4, \dots, n$.

From Example 3.19, it should be clear (Problem 3.45) that, if the time series is a causal AR(p) process, then, for $n \geq p$,

$$x_{n+1}^n = \phi_1 x_n + \phi_2 x_{n-1} + \cdots + \phi_p x_{n-p+1}. \quad (3.67)$$

For ARMA models in general, the prediction equations will not be as simple as the pure AR case. In addition, for n large, the use of (3.64) is prohibitive because it requires the inversion of a large matrix. There are, however, iterative solutions that do not require any matrix inversion. In particular, we mention the recursive solution due to Levinson (1947) and Durbin (1960).

Property 3.4 The Durbin–Levinson Algorithm

Equations (3.64) and (3.66) can be solved iteratively as follows:

$$\phi_{00} = 0, \quad P_1^0 = \gamma(0). \quad (3.68)$$

For $n \geq 1$,

$$\phi_{nn} = \frac{\rho(n) - \sum_{k=1}^{n-1} \phi_{n-1,k} \rho(n-k)}{1 - \sum_{k=1}^{n-1} \phi_{n-1,k} \rho(k)}, \quad P_{n+1}^n = P_n^{n-1}(1 - \phi_{nn}^2), \quad (3.69)$$

where, for $n \geq 2$,

$$\phi_{nk} = \phi_{n-1,k} - \phi_{nn}\phi_{n-1,n-k}, \quad k = 1, 2, \dots, n-1. \quad (3.70)$$

The proof of [Property 3.4](#) is left as an exercise; see [Problem 3.13](#).

Example 3.20 Using the Durbin–Levinson Algorithm

To use the algorithm, start with $\phi_{00} = 0$, $P_1^0 = \gamma(0)$. Then, for $n = 1$,

$$\phi_{11} = \rho(1), \quad P_2^1 = \gamma(0)[1 - \phi_{11}^2].$$

For $n = 2$,

$$\begin{aligned} \phi_{22} &= \frac{\rho(2) - \phi_{11} \rho(1)}{1 - \phi_{11} \rho(1)}, \quad \phi_{21} = \phi_{11} - \phi_{22}\phi_{11}, \\ P_3^2 &= P_2^1[1 - \phi_{22}^2] = \gamma(0)[1 - \phi_{11}^2][1 - \phi_{22}^2]. \end{aligned}$$

For $n = 3$,

$$\begin{aligned} \phi_{33} &= \frac{\rho(3) - \phi_{21} \rho(2) - \phi_{22} \rho(1)}{1 - \phi_{21} \rho(1) - \phi_{22} \rho(2)}, \\ \phi_{32} &= \phi_{22} - \phi_{33}\phi_{21}, \quad \phi_{31} = \phi_{21} - \phi_{33}\phi_{22}, \\ P_4^3 &= P_3^2[1 - \phi_{33}^2] = \gamma(0)[1 - \phi_{11}^2][1 - \phi_{22}^2][1 - \phi_{33}^2], \end{aligned}$$

and so on. Note that, in general, the standard error of the one-step-ahead forecast is the square root of

$$P_{n+1}^n = \gamma(0) \prod_{j=1}^n [1 - \phi_{jj}^2]. \quad (3.71)$$

An important consequence of the Durbin–Levinson algorithm is (see [Problem 3.13](#)) as follows.

Property 3.5 Iterative Solution for the PACF

The PACF of a stationary process x_t , can be obtained iteratively via (3.69) as ϕ_{nn} , for $n = 1, 2, \dots$.

Using [Property 3.5](#) and putting $n = p$ in (3.61) and (3.67), it follows that for an AR(p) model,

$$\begin{aligned} x_{p+1}^p &= \phi_{p1} x_p + \phi_{p2} x_{p-1} + \cdots + \phi_{pp} x_1 \\ &= \phi_1 x_p + \phi_2 x_{p-1} + \cdots + \phi_p x_1. \end{aligned} \quad (3.72)$$

Result (3.72) shows that for an AR(p) model, the partial autocorrelation coefficient at lag p , ϕ_{pp} , is also the last coefficient in the model, ϕ_p , as was claimed in [Example 3.16](#).

Example 3.21 The PACF of an AR(2)

We will use the results of [Example 3.20](#) and [Property 3.5](#) to calculate the first three values, ϕ_{11} , ϕ_{22} , ϕ_{33} , of the PACF. Recall from [Example 3.10](#) that $\rho(h) - \phi_1\rho(h-1) - \phi_2\rho(h-2) = 0$ for $h \geq 1$. When $h = 1, 2, 3$, we have $\rho(1) = \phi_1/(1-\phi_2)$, $\rho(2) = \phi_1\rho(1) + \phi_2$, $\rho(3) - \phi_1\rho(2) - \phi_2\rho(1) = 0$. Thus,

$$\begin{aligned}\phi_{11} &= \rho(1) = \frac{\phi_1}{1-\phi_2} \\ \phi_{22} &= \frac{\rho(2) - \rho(1)^2}{1 - \rho(1)^2} = \frac{\left[\phi_1\left(\frac{\phi_1}{1-\phi_2}\right) + \phi_2\right] - \left(\frac{\phi_1}{1-\phi_2}\right)^2}{1 - \left(\frac{\phi_1}{1-\phi_2}\right)^2} = \phi_2 \\ \phi_{21} &= \rho(1)[1 - \phi_2] = \phi_1 \\ \phi_{33} &= \frac{\rho(3) - \phi_1\rho(2) - \phi_2\rho(1)}{1 - \phi_1\rho(1) - \phi_2\rho(2)} = 0.\end{aligned}$$

Notice that, as shown in [\(3.72\)](#), $\phi_{22} = \phi_2$ for an AR(2) model.

So far, we have concentrated on one-step-ahead prediction, but [Property 3.3](#) allows us to calculate the BLP of x_{n+m} for any $m \geq 1$. Given data, $\{x_1, \dots, x_n\}$, the m -step-ahead predictor is

$$x_{n+m}^n = \phi_{n1}^{(m)} x_n + \phi_{n2}^{(m)} x_{n-1} + \dots + \phi_{nn}^{(m)} x_1, \quad (3.73)$$

where $\{\phi_{n1}^{(m)}, \phi_{n2}^{(m)}, \dots, \phi_{nn}^{(m)}\}$ satisfy the prediction equations,

$$\sum_{j=1}^n \phi_{nj}^{(m)} E(x_{n+1-j} x_{n+1-k}) = E(x_{n+m} x_{n+1-k}), \quad k = 1, \dots, n,$$

or

$$\sum_{j=1}^n \phi_{nj}^{(m)} \gamma(k-j) = \gamma(m+k-1), \quad k = 1, \dots, n. \quad (3.74)$$

The prediction equations can again be written in matrix notation as

$$\Gamma_n \phi_n^{(m)} = \gamma_n^{(m)}, \quad (3.75)$$

where $\gamma_n^{(m)} = (\gamma(m), \dots, \gamma(m+n-1))'$, and $\phi_n^{(m)} = (\phi_{n1}^{(m)}, \dots, \phi_{nn}^{(m)})'$ are $n \times 1$ vectors. The *mean square m -step-ahead prediction error* is

$$P_{n+m}^n = E(x_{n+m} - x_{n+m}^n)^2 = \gamma(0) - \gamma_n^{(m)'} \Gamma_n^{-1} \gamma_n^{(m)}. \quad (3.76)$$

Another useful algorithm for calculating forecasts was given by Brockwell and Davis (1991, Chapter 5). This algorithm follows directly from applying the projection theorem ([Theorem B.1](#)) to the *innovations*, $x_t - x_t^{t-1}$, for $t = 1, \dots, n$, using the fact that the innovations $x_t - x_t^{t-1}$ and $x_s - x_s^{s-1}$ are uncorrelated for $s \neq t$ (see [Problem 3.46](#)). We present the case in which x_t is a mean-zero stationary time series.

Property 3.6 The Innovations Algorithm

The one-step-ahead predictors, x_{t+1}^t , and their mean-squared errors, P_{t+1}^t , can be calculated iteratively as

$$x_1^0 = 0, \quad P_1^0 = \gamma(0)$$

$$x_{t+1}^t = \sum_{j=1}^t \theta_{tj}(x_{t+1-j} - x_{t+1-j}^{t-j}), \quad t = 1, 2, \dots \quad (3.77)$$

$$P_{t+1}^t = \gamma(0) - \sum_{j=0}^{t-1} \theta_{t,t-j}^2 P_{j+1}^j \quad t = 1, 2, \dots, \quad (3.78)$$

where, for $j = 0, 1, \dots, t-1$,

$$\theta_{t,t-j} = \left(\gamma(t-j) - \sum_{k=0}^{j-1} \theta_{j,j-k} \theta_{t-k} P_{k+1}^k \right) / P_{j+1}^j. \quad (3.79)$$

Given data x_1, \dots, x_n , the innovations algorithm can be calculated successively for $t = 1$, then $t = 2$ and so on, in which case the calculation of x_{n+1}^n and P_{n+1}^n is made at the final step $t = n$. The m -step-ahead predictor and its mean-square error based on the innovations algorithm (Problem 3.46) are given by

$$x_{n+m}^n = \sum_{j=m}^{n+m-1} \theta_{n+m-1,j}(x_{n+m-j} - x_{n+m-j}^{n+m-j-1}), \quad (3.80)$$

$$P_{n+m}^n = \gamma(0) - \sum_{j=m}^{n+m-1} \theta_{n+m-1,j}^2 P_{n+m-j}^{n+m-j-1}, \quad (3.81)$$

where the $\theta_{n+m-1,j}$ are obtained by continued iteration of (3.79).

Example 3.22 Prediction for an MA(1)

The innovations algorithm lends itself well to prediction for moving average processes. Consider an MA(1) model, $x_t = w_t + \theta w_{t-1}$. Recall that $\gamma(0) = (1 + \theta^2)\sigma_w^2$, $\gamma(1) = \theta\sigma_w^2$, and $\gamma(h) = 0$ for $h > 1$. Then, using Property 3.6, we have

$$\begin{aligned} \theta_{n1} &= \theta\sigma_w^2/P_n^{n-1} \\ \theta_{nj} &= 0, \quad j = 2, \dots, n \\ P_1^0 &= (1 + \theta^2)\sigma_w^2 \\ P_{n+1}^n &= (1 + \theta^2 - \theta\theta_{n1})\sigma_w^2. \end{aligned}$$

Finally, from (3.77), the one-step-ahead predictor is

$$x_{n+1}^n = \theta \left(x_n - x_n^{n-1} \right) \sigma_w^2 / P_n^{n-1}.$$

FORECASTING ARMA PROCESSES

The general prediction equations (3.60) provide little insight into forecasting for ARMA models in general. There are a number of different ways to express these forecasts, and each aids in understanding the special structure of ARMA prediction. Throughout, we assume x_t is a causal and invertible ARMA(p, q) process, $\phi(B)x_t = \theta(B)w_t$, where $w_t \sim \text{iid } N(0, \sigma_w^2)$. In the non-zero mean case, $E(x_t) = \mu_x$, simply replace x_t with $x_t - \mu_x$ in the model. First, we consider two types of forecasts. We write x_{n+m}^n to mean the minimum mean square error predictor of x_{n+m} based on the data $\{x_n, \dots, x_1\}$, that is,

$$x_{n+m}^n = E(x_{n+m} \mid x_n, \dots, x_1).$$

For ARMA models, it is easier to calculate the predictor of x_{n+m} , assuming we have the complete history of the process $\{x_n, x_{n-1}, \dots, x_1, x_0, x_{-1}, \dots\}$. We will denote the predictor of x_{n+m} based on the infinite past as

$$\tilde{x}_{n+m} = E(x_{n+m} \mid x_n, x_{n-1}, \dots, x_1, x_0, x_{-1}, \dots).$$

In general, x_{n+m}^n and \tilde{x}_{n+m} are not the same, but the idea here is that, for large samples, \tilde{x}_{n+m} will provide a good approximation to x_{n+m}^n .

Now, write x_{n+m} in its causal and invertible forms:

$$x_{n+m} = \sum_{j=0}^{\infty} \psi_j w_{n+m-j}, \quad \psi_0 = 1 \quad (3.82)$$

$$w_{n+m} = \sum_{j=0}^{\infty} \pi_j x_{n+m-j}, \quad \pi_0 = 1. \quad (3.83)$$

Then, taking conditional expectations in (3.82), we have

$$\tilde{x}_{n+m} = \sum_{j=0}^{\infty} \psi_j \tilde{w}_{n+m-j} = \sum_{j=m}^{\infty} \psi_j w_{n+m-j}, \quad (3.84)$$

because, by causality and invertibility,

$$\tilde{w}_t = E(w_t \mid x_n, x_{n-1}, \dots, x_0, x_{-1}, \dots) = \begin{cases} 0 & t > n \\ w_t & t \leq n. \end{cases}$$

Similarly, taking conditional expectations in (3.83), we have

$$0 = \tilde{x}_{n+m} + \sum_{j=1}^{\infty} \pi_j \tilde{x}_{n+m-j},$$

or

$$\tilde{x}_{n+m} = - \sum_{j=1}^{m-1} \pi_j \tilde{x}_{n+m-j} - \sum_{j=m}^{\infty} \pi_j x_{n+m-j}, \quad (3.85)$$

using the fact $E(x_t \mid x_n, x_{n-1}, \dots, x_0, x_{-1}, \dots) = x_t$, for $t \leq n$. Prediction is accomplished recursively using (3.85), starting with the one-step-ahead predictor, $m = 1$, and then continuing for $m = 2, 3, \dots$. Using (3.84), we can write

$$x_{n+m} - \tilde{x}_{n+m} = \sum_{j=0}^{m-1} \psi_j w_{n+m-j},$$

so the *mean-square prediction error* can be written as

$$P_{n+m}^n = E(x_{n+m} - \tilde{x}_{n+m})^2 = \sigma_w^2 \sum_{j=0}^{m-1} \psi_j^2. \quad (3.86)$$

Also, we note, for a fixed sample size, n , the prediction errors are correlated. That is, for $k \geq 1$,

$$E\{(x_{n+m} - \tilde{x}_{n+m})(x_{n+m+k} - \tilde{x}_{n+m+k})\} = \sigma_w^2 \sum_{j=0}^{m-1} \psi_j \psi_{j+k}. \quad (3.87)$$

Example 3.23 Long-Range Forecasts

Consider forecasting an ARMA process with mean μ_x . Replacing x_{n+m} with $x_{n+m} - \mu_x$ in (3.82), and taking conditional expectation as in (3.84), we deduce that the m -step-ahead forecast can be written as

$$\tilde{x}_{n+m} = \mu_x + \sum_{j=m}^{\infty} \psi_j w_{n+m-j}. \quad (3.88)$$

Noting that the ψ -weights dampen to zero exponentially fast, it is clear that

$$\tilde{x}_{n+m} \rightarrow \mu_x \quad (3.89)$$

exponentially fast (in the mean square sense) as $m \rightarrow \infty$. Moreover, by (3.86), the mean square prediction error

$$P_{n+m}^n \rightarrow \sigma_w^2 \sum_{j=0}^{\infty} \psi_j^2 = \gamma_x(0) = \sigma_x^2, \quad (3.90)$$

exponentially fast as $m \rightarrow \infty$.

It should be clear from (3.89) and (3.90) that ARMA forecasts quickly settle to the mean with a constant prediction error as the forecast horizon, m , grows. This effect can be seen in Figure 3.7 where the Recruitment series is forecast for 24 months; see Example 3.25.

When n is small, the general prediction equations (3.60) can be used easily. When n is large, we would use (3.85) by truncating, because we do not observe

$x_0, x_{-1}, x_{-2}, \dots$, and only the data x_1, x_2, \dots, x_n are available. In this case, we can truncate (3.85) by setting $\sum_{j=n+m}^{\infty} \pi_j x_{n+m-j} = 0$. The *truncated predictor* is then written as

$$\tilde{x}_{n+m}^n = - \sum_{j=1}^{m-1} \pi_j \tilde{x}_{n+m-j}^n - \sum_{j=m}^{n+m-1} \pi_j x_{n+m-j}, \quad (3.91)$$

which is also calculated recursively, $m = 1, 2, \dots$. The mean square prediction error, in this case, is approximated using (3.86).

For AR(p) models, and when $n > p$, equation (3.67) yields the exact predictor, x_{n+m}^n , of x_{n+m} , and there is no need for approximations. That is, for $n > p$, $\tilde{x}_{n+m}^n = \tilde{x}_{n+m} = x_{n+m}^n$. Also, in this case, the one-step-ahead prediction error is $E(x_{n+1} - x_{n+1}^n)^2 = \sigma_w^2$. For pure MA(q) or ARMA(p, q) models, truncated prediction has a fairly nice form.

Property 3.7 Truncated Prediction for ARMA

For ARMA(p, q) models, the truncated predictors for $m = 1, 2, \dots$, are

$$\tilde{x}_{n+m}^n = \phi_1 \tilde{x}_{n+m-1}^n + \dots + \phi_p \tilde{x}_{n+m-p}^n + \theta_1 \tilde{w}_{n+m-1}^n + \dots + \theta_q \tilde{w}_{n+m-q}^n, \quad (3.92)$$

where $\tilde{x}_t^n = x_t$ for $1 \leq t \leq n$ and $\tilde{x}_t^n = 0$ for $t \leq 0$. The truncated prediction errors are given by: $\tilde{w}_t^n = 0$ for $t \leq 0$ or $t > n$, and

$$\tilde{w}_t^n = \phi(B) \tilde{x}_t^n - \theta_1 \tilde{w}_{t-1}^n - \dots - \theta_q \tilde{w}_{t-q}^n$$

for $1 \leq t \leq n$.

Example 3.24 Forecasting an ARMA(1, 1) Series

Given data x_1, \dots, x_n , for forecasting purposes, write the model as

$$x_{n+1} = \phi x_n + w_{n+1} + \theta w_n.$$

Then, based on (3.92), the one-step-ahead truncated forecast is

$$\tilde{x}_{n+1}^n = \phi x_n + 0 + \theta \tilde{w}_n^n.$$

For $m \geq 2$, we have

$$\tilde{x}_{n+m}^n = \phi \tilde{x}_{n+m-1}^n,$$

which can be calculated recursively, $m = 2, 3, \dots$

To calculate \tilde{w}_n^n , which is needed to initialize the successive forecasts, the model can be written as $w_t = x_t - \phi x_{t-1} - \theta w_{t-1}$ for $t = 1, \dots, n$. For truncated forecasting using (3.92), put $\tilde{w}_0^n = 0$, $x_0 = 0$, and then iterate the errors forward in time

$$\tilde{w}_t^n = x_t - \phi x_{t-1} - \theta \tilde{w}_{t-1}^n, \quad t = 1, \dots, n.$$

The approximate forecast variance is computed from (3.86) using the ψ -weights determined as in Example 3.12. In particular, the ψ -weights satisfy $\psi_j = (\phi + \theta)\phi^{j-1}$, for $j \geq 1$. This result gives

$$P_{n+m}^n = \sigma_w^2 \left[1 + (\phi + \theta)^2 \sum_{j=1}^{m-1} \phi^{2(j-1)} \right] = \sigma_w^2 \left[1 + \frac{(\phi + \theta)^2 (1 - \phi^{2(m-1)})}{(1 - \phi^2)} \right].$$

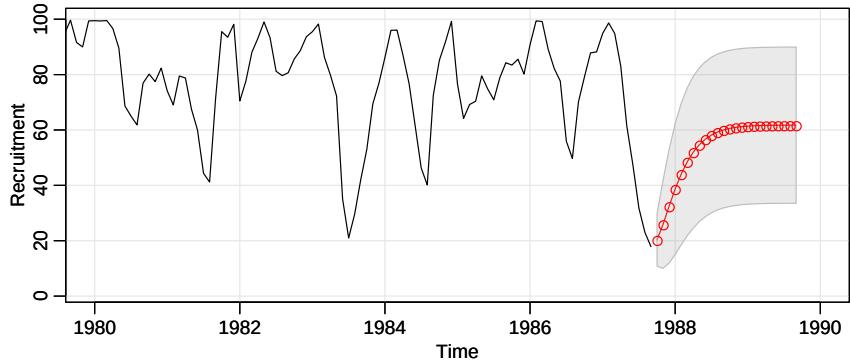


Fig. 3.7. Twenty-four month forecasts for the Recruitment series. The actual data shown are from about January 1980 to September 1987, and then the forecasts plus and minus one standard error are displayed.

To assess the precision of the forecasts, *prediction intervals* are typically calculated along with the forecasts. In general, $(1 - \alpha)$ prediction intervals are of the form

$$x_{n+m}^n \pm c_{\frac{\alpha}{2}} \sqrt{P_{n+m}^n}, \quad (3.93)$$

where $c_{\alpha/2}$ is chosen to get the desired degree of confidence. For example, if the process is Gaussian, then choosing $c_{\alpha/2} = 2$ will yield an approximate 95% prediction interval for x_{n+m} . If we are interested in establishing prediction intervals over more than one time period, then $c_{\alpha/2}$ should be adjusted appropriately, for example, by using Bonferroni's inequality [see (4.63) in Chapter 4 or Johnson and Wichern, 1992, Chapter 5].

Example 3.25 Forecasting the Recruitment Series

Using the parameter estimates as the actual parameter values, Figure 3.7 shows the result of forecasting the Recruitment series given in Example 3.18 over a 24-month horizon, $m = 1, 2, \dots, 24$. The actual forecasts are calculated as

$$x_{n+m}^n = 6.74 + 1.35x_{n+m-1}^n - .46x_{n+m-2}^n$$

for $n = 453$ and $m = 1, 2, \dots, 12$. Recall that $x_t^s = x_t$ when $t \leq s$. The forecasts errors P_{n+m}^n are calculated using (3.86). Recall that $\hat{\sigma}_w^2 = 89.72$, and using (3.40) from Example 3.12, we have $\psi_j = 1.35\psi_{j-1} - .46\psi_{j-2}$ for $j \geq 2$, where $\psi_0 = 1$ and $\psi_1 = 1.35$. Thus, for $n = 453$,

$$\begin{aligned} P_{n+1}^n &= 89.72, \\ P_{n+2}^n &= 89.72(1 + 1.35^2), \\ P_{n+3}^n &= 89.72(1 + 1.35^2 + [1.35^2 - .46]^2), \end{aligned}$$

and so on.

Note how the forecast levels off quickly and the prediction intervals are wide, even though in this case the forecast limits are only based on one standard error; that is, $x_{n+m}^n \pm \sqrt{P_{n+m}^n}$.

To reproduce the analysis and Figure 3.7, use the following commands:

```
regr = ar.ols(rec, order=2, demean=FALSE, intercept=TRUE)
fore = predict(regr, n.ahead=24)
ts.plot(rec, fore$pred, col=1:2, xlim=c(1980, 1990), ylab="Recruitment")
U = fore$pred+fore$se; L = fore$pred-fore$se
xx = c(time(U), rev(time(U))); yy = c(L, rev(U))
polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(fore$pred, type="p", col=2)
```

We complete this section with a brief discussion of *backcasting*. In backcasting, we want to predict x_{1-m} , for $m = 1, 2, \dots$, based on the data $\{x_1, \dots, x_n\}$. Write the backcast as

$$x_{1-m}^n = \sum_{j=1}^n \alpha_j x_j. \quad (3.94)$$

Analogous to (3.74), the prediction equations (assuming $\mu_x = 0$) are

$$\sum_{j=1}^n \alpha_j E(x_j x_k) = E(x_{1-m} x_k), \quad k = 1, \dots, n, \quad (3.95)$$

or

$$\sum_{j=1}^n \alpha_j \gamma(k-j) = \gamma(m+k-1), \quad k = 1, \dots, n. \quad (3.96)$$

These equations are precisely the prediction equations for forward prediction. That is, $\alpha_j \equiv \phi_{nj}^{(m)}$, for $j = 1, \dots, n$, where the $\phi_{nj}^{(m)}$ are given by (3.75). Finally, the backcasts are given by

$$x_{1-m}^n = \phi_{n1}^{(m)} x_1 + \dots + \phi_{nn}^{(m)} x_n, \quad m = 1, 2, \dots \quad (3.97)$$

Example 3.26 Backcasting an ARMA(1, 1)

Consider an ARMA(1, 1) process, $x_t = \phi x_{t-1} + \theta w_{t-1} + w_t$; we will call this the *forward model*. We have just seen that best linear prediction backward in time is the same as best linear prediction forward in time for stationary models. Assuming the models are Gaussian, we also have that minimum mean square error prediction backward in time is the same as forward in time for ARMA models.^{3.4} Thus, the process can equivalently be generated by the *backward model*,

$$x_t = \phi x_{t+1} + \theta v_{t+1} + v_t,$$

^{3.4} In the stationary Gaussian case, (a) the distribution of $\{x_{n+1}, x_n, \dots, x_1\}$ is the same as (b) the distribution of $\{x_0, x_1, \dots, x_n\}$. In forecasting we use (a) to obtain $E(x_{n+1} | x_n, \dots, x_1)$; in backcasting we use (b) to obtain $E(x_0 | x_1, \dots, x_n)$. Because (a) and (b) are the same, the two problems are equivalent.

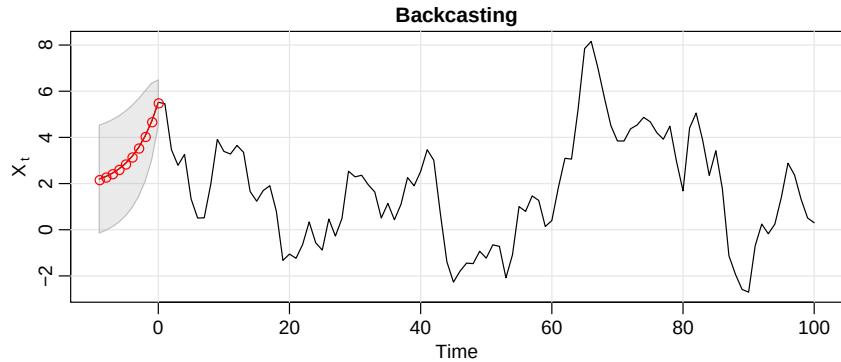


Fig. 3.8. Display for *Example 3.26*; backcasts from a simulated ARMA(1,1).

where $\{v_t\}$ is a Gaussian white noise process with variance σ_w^2 . We may write $x_t = \sum_{j=0}^{\infty} \psi_j v_{t+j}$, where $\psi_0 = 1$; this means that x_t is uncorrelated with $\{v_{t-1}, v_{t-2}, \dots\}$, in analogy to the forward model.

Given data $\{x_1, \dots, x_n\}$, truncate $v_n^n = E(v_n | x_1, \dots, x_n)$ to zero and then iterate backward. That is, put $\tilde{v}_n^n = 0$, as an initial approximation, and then generate the errors backward

$$\tilde{v}_t^n = x_t - \phi x_{t+1} - \theta \tilde{v}_{t+1}^n, \quad t = (n-1), (n-2), \dots, 1.$$

Then,

$$\tilde{x}_0^n = \phi x_1 + \theta \tilde{v}_1^n + \tilde{v}_0^n = \phi x_1 + \theta \tilde{v}_1^n,$$

because $\tilde{v}_t^n = 0$ for $t \leq 0$. Continuing, the general truncated backcasts are given by

$$\tilde{x}_{1-m}^n = \phi \tilde{x}_{2-m}^n, \quad m = 2, 3, \dots.$$

To backcast data in R, simply reverse the data, fit the model and predict. In the following, we backcasted a simulated ARMA(1,1) process; see Figure 3.8.

```
set.seed(90210)
x      = arima.sim(list(order = c(1,0,1), ar = .9, ma=.5), n = 100)
xr     = rev(x)                                # xr is the reversed data
pxr   = predict(arima(xr, order=c(1,0,1)), 10)  # predict the reversed data
pxrp  = rev(pxr$pred)                          # reorder the predictors (for plotting)
pxrse = rev(pxr$se)                            # reorder the SEs
nx    = ts(c(pxr$pred, x), start=-9)          # attach the backcasts to the data
plot(nx, ylab=expression(X[t]), main='Backcasting')
U = nx[1:10] + pxrse; L = nx[1:10] - pxrse
xx = c(-9:0, 0:-9); yy = c(L, rev(U))
polygon(xx, yy, border = 8, col = gray(0.6, alpha = 0.2))
lines(-9:0, nx[1:10], col=2, type='o')
```

3.5 Estimation

Throughout this section, we assume we have n observations, x_1, \dots, x_n , from a causal and invertible Gaussian ARMA(p, q) process in which, initially, the order parameters, p and q , are known. Our goal is to estimate the parameters, $\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q$, and σ_w^2 . We will discuss the problem of determining p and q later in this section.

We begin with *method of moments* estimators. The idea behind these estimators is that of equating population moments to sample moments and then solving for the parameters in terms of the sample moments. We immediately see that, if $E(x_t) = \mu$, then the method of moments estimator of μ is the sample average, \bar{x} . Thus, while discussing method of moments, we will assume $\mu = 0$. Although the method of moments can produce good estimators, they can sometimes lead to suboptimal estimators. We first consider the case in which the method leads to optimal (efficient) estimators, that is, AR(p) models,

$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t,$$

where the first $p + 1$ equations of (3.47) and (3.48) lead to the following:

Definition 3.10 *The Yule–Walker equations are given by*

$$\gamma(h) = \phi_1 \gamma(h-1) + \dots + \phi_p \gamma(h-p), \quad h = 1, 2, \dots, p, \quad (3.98)$$

$$\sigma_w^2 = \gamma(0) - \phi_1 \gamma(1) - \dots - \phi_p \gamma(p). \quad (3.99)$$

In matrix notation, the Yule–Walker equations are

$$\Gamma_p \phi = \gamma_p, \quad \sigma_w^2 = \gamma(0) - \phi' \gamma_p, \quad (3.100)$$

where $\Gamma_p = \{\gamma(k-j)\}_{j,k=1}^p$ is a $p \times p$ matrix, $\phi = (\phi_1, \dots, \phi_p)'$ is a $p \times 1$ vector, and $\gamma_p = (\gamma(1), \dots, \gamma(p))'$ is a $p \times 1$ vector. Using the method of moments, we replace $\gamma(h)$ in (3.100) by $\hat{\gamma}(h)$ [see equation (1.36)] and solve

$$\hat{\phi} = \hat{F}_p^{-1} \hat{\gamma}_p, \quad \hat{\sigma}_w^2 = \hat{\gamma}(0) - \hat{\gamma}' \hat{F}_p^{-1} \hat{\gamma}_p. \quad (3.101)$$

These estimators are typically called the *Yule–Walker estimators*. For calculation purposes, it is sometimes more convenient to work with the sample ACF. By factoring $\hat{\gamma}(0)$ in (3.101), we can write the Yule–Walker estimates as

$$\hat{\phi} = \hat{R}_p^{-1} \hat{\rho}_p, \quad \hat{\sigma}_w^2 = \hat{\gamma}(0) [1 - \hat{\rho}_p' \hat{R}_p^{-1} \hat{\rho}_p], \quad (3.102)$$

where $\hat{R}_p = \{\hat{\rho}(k-j)\}_{j,k=1}^p$ is a $p \times p$ matrix and $\hat{\rho}_p = (\hat{\rho}(1), \dots, \hat{\rho}(p))'$ is a $p \times 1$ vector.

For AR(p) models, if the sample size is large, the Yule–Walker estimators are approximately normally distributed, and $\hat{\sigma}_w^2$ is close to the true value of σ_w^2 . We state these results in [Property 3.8](#); for details, see [Section B.3](#).

Property 3.8 Large Sample Results for Yule–Walker Estimators

The asymptotic ($n \rightarrow \infty$) behavior of the Yule–Walker estimators in the case of causal AR(p) processes is as follows:

$$\sqrt{n} (\hat{\phi} - \phi) \xrightarrow{d} N\left(0, \sigma_w^2 \Gamma_p^{-1}\right), \quad \hat{\sigma}_w^2 \xrightarrow{P} \sigma_w^2. \quad (3.103)$$

The Durbin–Levinson algorithm, (3.68)–(3.70), can be used to calculate $\hat{\phi}$ without inverting $\hat{\Gamma}_p$ or \hat{R}_p , by replacing $\gamma(h)$ by $\hat{\gamma}(h)$ in the algorithm. In running the algorithm, we will iteratively calculate the $h \times 1$ vector, $\hat{\phi}_h = (\hat{\phi}_{h1}, \dots, \hat{\phi}_{hh})'$, for $h = 1, 2, \dots$. Thus, in addition to obtaining the desired forecasts, the Durbin–Levinson algorithm yields $\hat{\phi}_{hh}$, the sample PACF. Using (3.103), we can show the following property.

Property 3.9 Large Sample Distribution of the PACF

For a causal AR(p) process, asymptotically ($n \rightarrow \infty$),

$$\sqrt{n} \hat{\phi}_{hh} \xrightarrow{d} N(0, 1), \quad \text{for } h > p. \quad (3.104)$$

Example 3.27 Yule–Walker Estimation for an AR(2) Process

The data shown in Figure 3.4 were $n = 144$ simulated observations from the AR(2) model

$$x_t = 1.5x_{t-1} - .75x_{t-2} + w_t,$$

where $w_t \sim \text{iid } N(0, 1)$. For these data, $\hat{\gamma}(0) = 8.903$, $\hat{\rho}(1) = .849$, and $\hat{\rho}(2) = .519$. Thus,

$$\hat{\phi} = \begin{pmatrix} \hat{\phi}_1 \\ \hat{\phi}_2 \end{pmatrix} = \begin{bmatrix} 1 & .849 \\ .849 & 1 \end{bmatrix}^{-1} \begin{pmatrix} .849 \\ .519 \end{pmatrix} = \begin{pmatrix} 1.463 \\ -.723 \end{pmatrix}$$

and

$$\hat{\sigma}_w^2 = 8.903 \left[1 - (.849, .519) \begin{pmatrix} 1.463 \\ -.723 \end{pmatrix} \right] = 1.187.$$

By Property 3.8, the asymptotic variance–covariance matrix of $\hat{\phi}$ is

$$\frac{1}{144} \frac{1.187}{8.903} \begin{bmatrix} 1 & .849 \\ .849 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} .058^2 & -.003 \\ -.003 & .058^2 \end{bmatrix},$$

and it can be used to get confidence regions for, or make inferences about $\hat{\phi}$ and its components. For example, an approximate 95% confidence interval for ϕ_2 is $-.723 \pm 2(.058)$, or $(-.838, -.608)$, which contains the true value of $\phi_2 = -.75$.

For these data, the first three sample partial autocorrelations are $\hat{\phi}_{11} = \hat{\rho}(1) = .849$, $\hat{\phi}_{22} = \hat{\phi}_2 = -.721$, and $\hat{\phi}_{33} = -.085$. According to Property 3.9, the asymptotic standard error of $\hat{\phi}_{33}$ is $1/\sqrt{144} = .083$, and the observed value, $-.085$, is about only one standard deviation from $\phi_{33} = 0$.

Example 3.28 Yule–Walker Estimation of the Recruitment Series

In Example 3.18 we fit an AR(2) model to the recruitment series using ordinary least squares (OLS). For AR models, the estimators obtained via OLS and Yule–Walker are nearly identical; we will see this when we discuss conditional sum of squares estimation in (3.111)–(3.116).

Below are the results of fitting the same model using Yule–Walker estimation in R, which are nearly identical to the values in Example 3.18.

```
rec.yw = ar.yw(rec, order=2)
rec.yw$x.mean    # = 62.26 (mean estimate)
rec.yw$ar         # = 1.33, -.44 (coefficient estimates)
sqrt(diag(rec.yw$asy.var.coef)) # = .04, .04 (standard errors)
rec.yw$var.pred  # = 94.80 (error variance estimate)
```

To obtain the 24 month ahead predictions and their standard errors, and then plot the results (not shown) as in Example 3.25, use the R commands:

```
rec.pr = predict(rec.yw, n.ahead=24)
ts.plot(rec, rec.pr$pred, col=1:2)
lines(rec.pr$pred + rec.pr$se, col=4, lty=2)
lines(rec.pr$pred - rec.pr$se, col=4, lty=2)
```

In the case of AR(p) models, the Yule–Walker estimators given in (3.102) are optimal in the sense that the asymptotic distribution, (3.103), is the best asymptotic normal distribution. This is because, given initial conditions, AR(p) models are linear models, and the Yule–Walker estimators are essentially least squares estimators. If we use method of moments for MA or ARMA models, we will not get optimal estimators because such processes are nonlinear in the parameters.

Example 3.29 Method of Moments Estimation for an MA(1)

Consider the time series

$$x_t = w_t + \theta w_{t-1},$$

where $|\theta| < 1$. The model can then be written as

$$x_t = \sum_{j=1}^{\infty} (-\theta)^j x_{t-j} + w_t,$$

which is nonlinear in θ . The first two population autocovariances are $\gamma(0) = \sigma_w^2(1 + \theta^2)$ and $\gamma(1) = \sigma_w^2\theta$, so the estimate of θ is found by solving:

$$\hat{\rho}(1) = \frac{\hat{\gamma}(1)}{\hat{\gamma}(0)} = \frac{\hat{\theta}}{1 + \hat{\theta}^2}.$$

Two solutions exist, so we would pick the invertible one. If $|\hat{\rho}(1)| \leq \frac{1}{2}$, the solutions are real, otherwise, a real solution does not exist. Even though $|\rho(1)| < \frac{1}{2}$ for an invertible MA(1), it may happen that $|\hat{\rho}(1)| \geq \frac{1}{2}$ because it is an estimator. For example, the following simulation in R produces a value of $\hat{\rho}(1) = .507$ when the true value is $\rho(1) = .9/(1 + .9^2) = .497$.

```
set.seed(2)
ma1 = arima.sim(list(order = c(0,0,1), ma = 0.9), n = 50)
acf(ma1, plot=FALSE)[1] # = .507 (lag 1 sample ACF)
```

When $|\hat{\rho}(1)| < \frac{1}{2}$, the invertible estimate is

$$\hat{\theta} = \frac{1 - \sqrt{1 - 4\hat{\rho}(1)^2}}{2\hat{\rho}(1)}. \quad (3.105)$$

It can be shown that^{3.5}

$$\hat{\theta} \sim \text{AN}\left(\theta, \frac{1 + \theta^2 + 4\theta^4 + \theta^6 + \theta^8}{n(1 - \theta^2)^2}\right);$$

AN is read *asymptotically normal* and is defined in [Definition A.5](#). The maximum likelihood estimator (which we discuss next) of θ , in this case, has an asymptotic variance of $(1 - \theta^2)/n$. When $\theta = .5$, for example, the ratio of the asymptotic variance of the method of moments estimator to the maximum likelihood estimator of θ is about 3.5. That is, for large samples, the variance of the method of moments estimator is about 3.5 times larger than the variance of the MLE of θ when $\theta = .5$.

MAXIMUM LIKELIHOOD AND LEAST SQUARES ESTIMATION

To fix ideas, we first focus on the causal AR(1) case. Let

$$x_t = \mu + \phi(x_{t-1} - \mu) + w_t \quad (3.106)$$

where $|\phi| < 1$ and $w_t \sim \text{iid } N(0, \sigma_w^2)$. Given data x_1, x_2, \dots, x_n , we seek the likelihood

$$L(\mu, \phi, \sigma_w^2) = f\left(x_1, x_2, \dots, x_n \mid \mu, \phi, \sigma_w^2\right).$$

In the case of an AR(1), we may write the likelihood as

$$L(\mu, \phi, \sigma_w^2) = f(x_1)f(x_2 \mid x_1) \cdots f(x_n \mid x_{n-1}),$$

where we have dropped the parameters in the densities, $f(\cdot)$, to ease the notation. Because $x_t \mid x_{t-1} \sim N(\mu + \phi(x_{t-1} - \mu), \sigma_w^2)$, we have

$$f(x_t \mid x_{t-1}) = f_w[(x_t - \mu) - \phi(x_{t-1} - \mu)],$$

where $f_w(\cdot)$ is the density of w_t , that is, the normal density with mean zero and variance σ_w^2 . We may then write the likelihood as

$$L(\mu, \phi, \sigma_w^2) = f(x_1) \prod_{t=2}^n f_w[(x_t - \mu) - \phi(x_{t-1} - \mu)].$$

^{3.5} The result follows from [Theorem A.7](#) and the delta method. See the proof of [Theorem A.7](#) for details on the delta method.

To find $f(x_1)$, we can use the causal representation

$$x_1 = \mu + \sum_{j=0}^{\infty} \phi^j w_{1-j}$$

to see that x_1 is normal, with mean μ and variance $\sigma_w^2/(1-\phi^2)$. Finally, for an AR(1), the likelihood is

$$L(\mu, \phi, \sigma_w^2) = (2\pi\sigma_w^2)^{-n/2}(1-\phi^2)^{1/2} \exp\left[-\frac{S(\mu, \phi)}{2\sigma_w^2}\right], \quad (3.107)$$

where

$$S(\mu, \phi) = (1-\phi^2)(x_1 - \mu)^2 + \sum_{t=2}^n [(x_t - \mu) - \phi(x_{t-1} - \mu)]^2. \quad (3.108)$$

Typically, $S(\mu, \phi)$ is called the *unconditional sum of squares*. We could have also considered the estimation of μ and ϕ using *unconditional least squares*, that is, estimation by minimizing $S(\mu, \phi)$.

Taking the partial derivative of the log of (3.107) with respect to σ_w^2 and setting the result equal to zero, we get the typical normal result that for any given values of μ and ϕ in the parameter space, $\sigma_w^2 = n^{-1}S(\mu, \phi)$ maximizes the likelihood. Thus, the maximum likelihood estimate of σ_w^2 is

$$\hat{\sigma}_w^2 = n^{-1}S(\hat{\mu}, \hat{\phi}), \quad (3.109)$$

where $\hat{\mu}$ and $\hat{\phi}$ are the MLEs of μ and ϕ , respectively. If we replace n in (3.109) by $n-2$, we would obtain the unconditional least squares estimate of σ_w^2 .

If, in (3.107), we take logs, replace σ_w^2 by $\hat{\sigma}_w^2$, and ignore constants, $\hat{\mu}$ and $\hat{\phi}$ are the values that minimize the criterion function

$$l(\mu, \phi) = \log [n^{-1}S(\mu, \phi)] - n^{-1}\log(1-\phi^2); \quad (3.110)$$

that is, $l(\mu, \phi) \propto -2\log L(\mu, \phi, \hat{\sigma}_w^2)$.^{3.6} Because (3.108) and (3.110) are complicated functions of the parameters, the minimization of $l(\mu, \phi)$ or $S(\mu, \phi)$ is accomplished numerically. In the case of AR models, we have the advantage that, conditional on initial values, they are linear models. That is, we can drop the term in the likelihood that causes the nonlinearity. Conditioning on x_1 , the *conditional likelihood* becomes

$$\begin{aligned} L(\mu, \phi, \sigma_w^2 \mid x_1) &= \prod_{t=2}^n f_w[(x_t - \mu) - \phi(x_{t-1} - \mu)] \\ &= (2\pi\sigma_w^2)^{-(n-1)/2} \exp\left[-\frac{S_c(\mu, \phi)}{2\sigma_w^2}\right], \end{aligned} \quad (3.111)$$

where the *conditional sum of squares* is

^{3.6} The criterion function is sometimes called the profile or concentrated likelihood.

$$S_c(\mu, \phi) = \sum_{t=2}^n [(x_t - \mu) - \phi(x_{t-1} - \mu)]^2. \quad (3.112)$$

The conditional MLE of σ_w^2 is

$$\hat{\sigma}_w^2 = S_c(\hat{\mu}, \hat{\phi})/(n-1), \quad (3.113)$$

and $\hat{\mu}$ and $\hat{\phi}$ are the values that minimize the conditional sum of squares, $S_c(\mu, \phi)$. Letting $\alpha = \mu(1 - \phi)$, the conditional sum of squares can be written as

$$S_c(\mu, \phi) = \sum_{t=2}^n [x_t - (\alpha + \phi x_{t-1})]^2. \quad (3.114)$$

The problem is now the linear regression problem stated in [Section 2.1](#). Following the results from least squares estimation, we have $\hat{\alpha} = \bar{x}_{(2)} - \hat{\phi}\bar{x}_{(1)}$, where $\bar{x}_{(1)} = (n-1)^{-1} \sum_{t=1}^{n-1} x_t$, and $\bar{x}_{(2)} = (n-1)^{-1} \sum_{t=2}^n x_t$, and the conditional estimates are then

$$\hat{\mu} = \frac{\bar{x}_{(2)} - \hat{\phi}\bar{x}_{(1)}}{1 - \hat{\phi}} \quad (3.115)$$

$$\hat{\phi} = \frac{\sum_{t=2}^n (x_t - \bar{x}_{(2)})(x_{t-1} - \bar{x}_{(1)})}{\sum_{t=2}^n (x_{t-1} - \bar{x}_{(1)})^2}. \quad (3.116)$$

From [\(3.115\)](#) and [\(3.116\)](#), we see that $\hat{\mu} \approx \bar{x}$ and $\hat{\phi} \approx \hat{\rho}(1)$. That is, the Yule–Walker estimators and the conditional least squares estimators are approximately the same. The only difference is the inclusion or exclusion of terms involving the endpoints, x_1 and x_n . We can also adjust the estimate of σ_w^2 in [\(3.113\)](#) to be equivalent to the least squares estimator, that is, divide $S_c(\hat{\mu}, \hat{\phi})$ by $(n-3)$ instead of $(n-1)$ in [\(3.113\)](#).

For general AR(p) models, maximum likelihood estimation, unconditional least squares, and conditional least squares follow analogously to the AR(1) example. For general ARMA models, it is difficult to write the likelihood as an explicit function of the parameters. Instead, it is advantageous to write the likelihood in terms of the *innovations*, or one-step-ahead prediction errors, $x_t - x_t^{t-1}$. This will also be useful in [Chapter 6](#) when we study state-space models.

For a normal ARMA(p, q) model, let $\beta = (\mu, \phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q)'$ be the $(p+q+1)$ -dimensional vector of the model parameters. The likelihood can be written as

$$L(\beta, \sigma_w^2) = \prod_{t=1}^n f(x_t \mid x_{t-1}, \dots, x_1).$$

The conditional distribution of x_t given x_{t-1}, \dots, x_1 is Gaussian with mean x_t^{t-1} and variance P_t^{t-1} . Recall from [\(3.71\)](#) that $P_t^{t-1} = \gamma(0) \prod_{j=1}^{t-1} (1 - \phi_{jj}^2)$. For ARMA models, $\gamma(0) = \sigma_w^2 \sum_{j=0}^{\infty} \psi_j^2$, in which case we may write

$$P_t^{t-1} = \sigma_w^2 \left\{ \left[\sum_{j=0}^{\infty} \psi_j^2 \right] \left[\prod_{j=1}^{t-1} (1 - \phi_{jj}^2) \right] \right\} \stackrel{\text{def}}{=} \sigma_w^2 r_t,$$

where r_t is the term in the braces. Note that the r_t terms are functions only of the regression parameters and that they may be computed recursively as $r_{t+1} = (1 - \phi_{tt}^2)r_t$ with initial condition $r_1 = \sum_{j=0}^{\infty} \psi_j^2$. The likelihood of the data can now be written as

$$L(\beta, \sigma_w^2) = (2\pi\sigma_w^2)^{-n/2} [r_1(\beta)r_2(\beta)\cdots r_n(\beta)]^{-1/2} \exp\left[-\frac{S(\beta)}{2\sigma_w^2}\right], \quad (3.117)$$

where

$$S(\beta) = \sum_{t=1}^n \left[\frac{(x_t - x_t^{t-1}(\beta))^2}{r_t(\beta)} \right]. \quad (3.118)$$

Both x_t^{t-1} and r_t are functions of β alone, and we make that fact explicit in (3.117)-(3.118). Given values for β and σ_w^2 , the likelihood may be evaluated using the techniques of [Section 3.4](#). Maximum likelihood estimation would now proceed by maximizing (3.117) with respect to β and σ_w^2 . As in the AR(1) example, we have

$$\hat{\sigma}_w^2 = n^{-1}S(\hat{\beta}), \quad (3.119)$$

where $\hat{\beta}$ is the value of β that minimizes the concentrated likelihood

$$l(\beta) = \log [n^{-1}S(\beta)] + n^{-1} \sum_{t=1}^n \log r_t(\beta). \quad (3.120)$$

For the AR(1) model (3.106) discussed previously, recall that $x_1^0 = \mu$ and $x_t^{t-1} = \mu + \phi(x_{t-1} - \mu)$, for $t = 2, \dots, n$. Also, using the fact that $\phi_{11} = \phi$ and $\phi_{hh} = 0$ for $h > 1$, we have $r_1 = \sum_{j=0}^{\infty} \phi^{2j} = (1 - \phi^2)^{-1}$, $r_2 = (1 - \phi^2)^{-1}(1 - \phi^2) = 1$, and in general, $r_t = 1$ for $t = 2, \dots, n$. Hence, the likelihood presented in (3.107) is identical to the innovations form of the likelihood given by (3.117). Moreover, the generic $S(\beta)$ in (3.118) is $S(\mu, \phi)$ given in (3.108) and the generic $l(\beta)$ in (3.120) is $l(\mu, \phi)$ in (3.110).

Unconditional least squares would be performed by minimizing (3.118) with respect to β . Conditional least squares estimation would involve minimizing (3.118) with respect to β but where, to ease the computational burden, the predictions and their errors are obtained by conditioning on initial values of the data. In general, numerical optimization routines are used to obtain the actual estimates and their standard errors.

Example 3.30 The Newton–Raphson and Scoring Algorithms

Two common numerical optimization routines for accomplishing maximum likelihood estimation are Newton–Raphson and scoring. We will give a brief account of the mathematical ideas here. The actual implementation of these algorithms is much more complicated than our discussion might imply. For details, the reader is referred to any of the *Numerical Recipes* books, for example, Press et al. (1993).

Let $l(\beta)$ be a criterion function of k parameters $\beta = (\beta_1, \dots, \beta_k)$ that we wish to minimize with respect to β . For example, consider the likelihood function given by (3.110) or by (3.120). Suppose $l(\hat{\beta})$ is the extremum that we are interested in

finding, and $\hat{\beta}$ is found by solving $\partial l(\beta)/\partial \beta_j = 0$, for $j = 1, \dots, k$. Let $l^{(1)}(\beta)$ denote the $k \times 1$ vector of partials

$$l^{(1)}(\beta) = \left(\frac{\partial l(\beta)}{\partial \beta_1}, \dots, \frac{\partial l(\beta)}{\partial \beta_k} \right)'$$

Note, $l^{(1)}(\hat{\beta}) = 0$, the $k \times 1$ zero vector. Let $l^{(2)}(\beta)$ denote the $k \times k$ matrix of second-order partials

$$l^{(2)}(\beta) = \left\{ -\frac{\partial^2 l(\beta)}{\partial \beta_i \partial \beta_j} \right\}_{i,j=1}^k,$$

and assume $l^{(2)}(\beta)$ is nonsingular. Let $\beta_{(0)}$ be a “sufficiently good” initial estimator of β . Then, using a Taylor expansion, we have the following approximation:

$$0 = l^{(1)}(\hat{\beta}) \approx l^{(1)}(\beta_{(0)}) - l^{(2)}(\beta_{(0)}) [\hat{\beta} - \beta_{(0)}].$$

Setting the right-hand side equal to zero and solving for $\hat{\beta}$ [call the solution $\beta_{(1)}$], we get

$$\beta_{(1)} = \beta_{(0)} + \left[l^{(2)}(\beta_{(0)}) \right]^{-1} l^{(1)}(\beta_{(0)}).$$

The Newton–Raphson algorithm proceeds by iterating this result, replacing $\beta_{(0)}$ by $\beta_{(1)}$ to get $\beta_{(2)}$, and so on, until convergence. Under a set of appropriate conditions, the sequence of estimators, $\beta_{(1)}, \beta_{(2)}, \dots$, will converge to $\hat{\beta}$, the MLE of β .

For maximum likelihood estimation, the criterion function used is $l(\beta)$ given by (3.120); $l^{(1)}(\beta)$ is called the score vector, and $l^{(2)}(\beta)$ is called the *Hessian*. In the method of scoring, we replace $l^{(2)}(\beta)$ by $E[l^{(2)}(\beta)]$, the *information* matrix. Under appropriate conditions, the inverse of the information matrix is the asymptotic variance–covariance matrix of the estimator $\hat{\beta}$. This is sometimes approximated by the inverse of the Hessian at $\hat{\beta}$. If the derivatives are difficult to obtain, it is possible to use quasi-maximum likelihood estimation where numerical techniques are used to approximate the derivatives.

Example 3.31 MLE for the Recruitment Series

So far, we have fit an AR(2) model to the Recruitment series using ordinary least squares ([Example 3.18](#)) and using Yule–Walker ([Example 3.28](#)). The following is an R session used to fit an AR(2) model via maximum likelihood estimation to the Recruitment series; these results can be compared to the results in [Example 3.18](#) and [Example 3.28](#).

```
rec.mle = ar.mle(rec, order=2)
rec.mle$x.mean   # 62.26
rec.mle$ar        # 1.35, -.46
sqrt(diag(rec.mle$asy.var.coef)) # .04, .04
rec.mle$var.pred  # 89.34
```

We now discuss least squares for ARMA(p, q) models via *Gauss–Newton*. For general and complete details of the Gauss–Newton procedure, the reader is referred

to Fuller (1996). As before, write $\beta = (\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q)'$, and for the ease of discussion, we will put $\mu = 0$. We write the model in terms of the errors

$$w_t(\beta) = x_t - \sum_{j=1}^p \phi_j x_{t-j} - \sum_{k=1}^q \theta_k w_{t-k}(\beta), \quad (3.121)$$

emphasizing the dependence of the errors on the parameters.

For conditional least squares, we approximate the residual sum of squares by conditioning on x_1, \dots, x_p (if $p > 0$) and $w_p = w_{p-1} = w_{p-2} = \dots = w_{1-q} = 0$ (if $q > 0$), in which case, given β , we may evaluate (3.121) for $t = p+1, p+2, \dots, n$. Using this conditioning argument, the conditional error sum of squares is

$$S_c(\beta) = \sum_{t=p+1}^n w_t^2(\beta). \quad (3.122)$$

Minimizing $S_c(\beta)$ with respect to β yields the conditional least squares estimates. If $q = 0$, the problem is linear regression and no iterative technique is needed to minimize $S_c(\phi_1, \dots, \phi_p)$. If $q > 0$, the problem becomes nonlinear regression and we will have to rely on numerical optimization.

When n is large, conditioning on a few initial values will have little influence on the final parameter estimates. In the case of small to moderate sample sizes, one may wish to rely on unconditional least squares. The unconditional least squares problem is to choose β to minimize the unconditional sum of squares, which we have generically denoted by $S(\beta)$ in this section. The unconditional sum of squares can be written in various ways, and one useful form in the case of ARMA(p, q) models is derived in Box et al. (1994, Appendix A7.3). They showed (see Problem 3.19) the unconditional sum of squares can be written as

$$S(\beta) = \sum_{t=-\infty}^n \tilde{w}_t^2(\beta), \quad (3.123)$$

where $\tilde{w}_t(\beta) = E(w_t | x_1, \dots, x_n)$. When $t \leq 0$, the $\tilde{w}_t(\beta)$ are obtained by backcasting. As a practical matter, we approximate $S(\beta)$ by starting the sum at $t = -M+1$, where M is chosen large enough to guarantee $\sum_{t=-\infty}^{-M} \tilde{w}_t^2(\beta) \approx 0$. In the case of unconditional least squares estimation, a numerical optimization technique is needed even when $q = 0$.

To employ Gauss–Newton, let $\beta_{(0)} = (\phi_1^{(0)}, \dots, \phi_p^{(0)}, \theta_1^{(0)}, \dots, \theta_q^{(0)})'$ be an initial estimate of β . For example, we could obtain $\beta_{(0)}$ by method of moments. The first-order Taylor expansion of $w_t(\beta)$ is

$$w_t(\beta) \approx w_t(\beta_{(0)}) - (\beta - \beta_{(0)})' z_t(\beta_{(0)}), \quad (3.124)$$

where

$$z'_t(\beta_{(0)}) = \left(-\frac{\partial w_t(\beta)}{\partial \beta_1}, \dots, -\frac{\partial w_t(\beta)}{\partial \beta_{p+q}} \right) \Bigg|_{\beta=\beta_{(0)}}, \quad t = 1, \dots, n.$$

The linear approximation of $S_c(\beta)$ is

$$Q(\beta) = \sum_{t=p+1}^n [w_t(\beta_{(0)}) - (\beta - \beta_{(0)})' z_t(\beta_{(0)})]^2 \quad (3.125)$$

and this is the quantity that we will minimize. For approximate unconditional least squares, we would start the sum in (3.125) at $t = -M + 1$, for a large value of M , and work with the backcasted values.

Using the results of ordinary least squares (Section 2.1), we know

$$\widehat{(\beta - \beta_{(0)})} = \left(n^{-1} \sum_{t=p+1}^n z_t(\beta_{(0)}) z_t'(\beta_{(0)}) \right)^{-1} \left(n^{-1} \sum_{t=p+1}^n z_t(\beta_{(0)}) w_t(\beta_{(0)}) \right) \quad (3.126)$$

minimizes $Q(\beta)$. From (3.126), we write the *one-step Gauss–Newton estimate* as

$$\beta_{(1)} = \beta_{(0)} + \Delta(\beta_{(0)}), \quad (3.127)$$

where $\Delta(\beta_{(0)})$ denotes the right-hand side of (3.126). Gauss–Newton estimation is accomplished by replacing $\beta_{(0)}$ by $\beta_{(1)}$ in (3.127). This process is repeated by calculating, at iteration $j = 2, 3, \dots$,

$$\beta_{(j)} = \beta_{(j-1)} + \Delta(\beta_{(j-1)})$$

until convergence.

Example 3.32 Gauss–Newton for an MA(1)

Consider an invertible MA(1) process, $x_t = w_t + \theta w_{t-1}$. Write the truncated errors as

$$w_t(\theta) = x_t - \theta w_{t-1}(\theta), \quad t = 1, \dots, n, \quad (3.128)$$

where we condition on $w_0(\theta) = 0$. Taking derivatives and negating,

$$-\frac{\partial w_t(\theta)}{\partial \theta} = w_{t-1}(\theta) + \theta \frac{\partial w_{t-1}(\theta)}{\partial \theta}, \quad t = 1, \dots, n, \quad (3.129)$$

where $\partial w_0(\theta)/\partial \theta = 0$. We can also write (3.129) as

$$z_t(\theta) = w_{t-1}(\theta) - \theta z_{t-1}(\theta), \quad t = 1, \dots, n, \quad (3.130)$$

where $z_t(\theta) = -\partial w_t(\theta)/\partial \theta$ and $z_0(\theta) = 0$.

Let $\theta_{(0)}$ be an initial estimate of θ , for example, the estimate given in Example 3.29. Then, the Gauss–Newton procedure for conditional least squares is given by

$$\theta_{(j+1)} = \theta_{(j)} + \frac{\sum_{t=1}^n z_t(\theta_{(j)}) w_t(\theta_{(j)})}{\sum_{t=1}^n z_t^2(\theta_{(j)})}, \quad j = 0, 1, 2, \dots, \quad (3.131)$$

where the values in (3.131) are calculated recursively using (3.128) and (3.130). The calculations are stopped when $|\theta_{(j+1)} - \theta_{(j)}|$, or $|Q(\theta_{(j+1)}) - Q(\theta_{(j)})|$, are smaller than some preset amount.

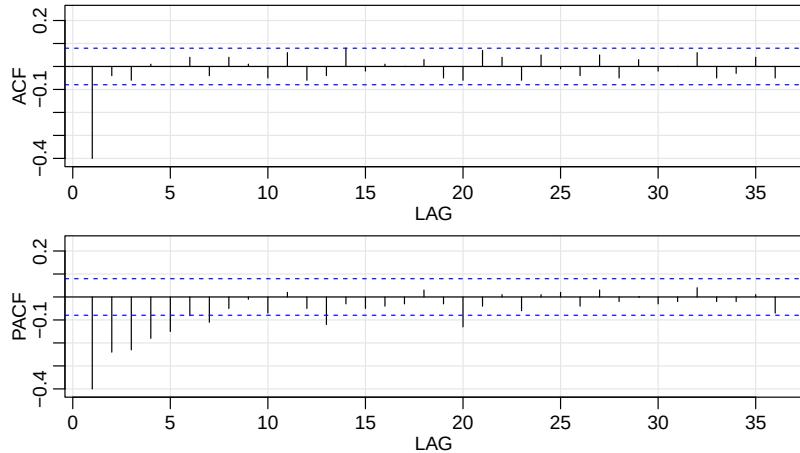


Fig. 3.9. ACF and PACF of transformed glacial varves.

Example 3.33 Fitting the Glacial Varve Series

Consider the series of glacial varve thicknesses from Massachusetts for $n = 634$ years, as analyzed in [Example 2.7](#) and in [Problem 2.8](#), where it was argued that a first-order moving average model might fit the logarithmically transformed and differenced varve series, say,

$$\nabla \log(x_t) = \log(x_t) - \log(x_{t-1}) = \log\left(\frac{x_t}{x_{t-1}}\right),$$

which can be interpreted as being approximately the percentage change in the thickness.

The sample ACF and PACF, shown in [Figure 3.9](#), confirm the tendency of $\nabla \log(x_t)$ to behave as a first-order moving average process as the ACF has only a significant peak at lag one and the PACF decreases exponentially. Using [Table 3.1](#), this sample behavior fits that of the MA(1) very well.

Since $\hat{\rho}(1) = -.397$, our initial estimate is $\theta_{(0)} = -.495$ using [\(3.105\)](#). The results of eleven iterations of the Gauss–Newton procedure, [\(3.131\)](#), starting with $\theta_{(0)}$ are given in [Table 3.2](#). The final estimate is $\hat{\theta} = \theta_{(11)} = -.773$; interim values and the corresponding value of the conditional sum of squares, $S_c(\theta)$ given in [\(3.122\)](#), are also displayed in the table. The final estimate of the error variance is $\hat{\sigma}_w^2 = 148.98/632 = .236$ with 632 degrees of freedom (one is lost in differencing). The value of the sum of the squared derivatives at convergence is $\sum_{t=1}^n z_t^2(\theta_{(11)}) = 368.741$, and consequently, the estimated standard error of $\hat{\theta}$ is $\sqrt{.236/368.741} = .025$,^{3.7} this leads to a t -value of $-.773/.025 = -30.92$ with 632 degrees of freedom.

[Figure 3.10](#) displays the conditional sum of squares, $S_c(\theta)$ as a function of θ , as well as indicating the values of each step of the Gauss–Newton algorithm. Note

^{3.7} To estimate the standard error, we are using the standard regression results from [\(2.6\)](#) as an approximation

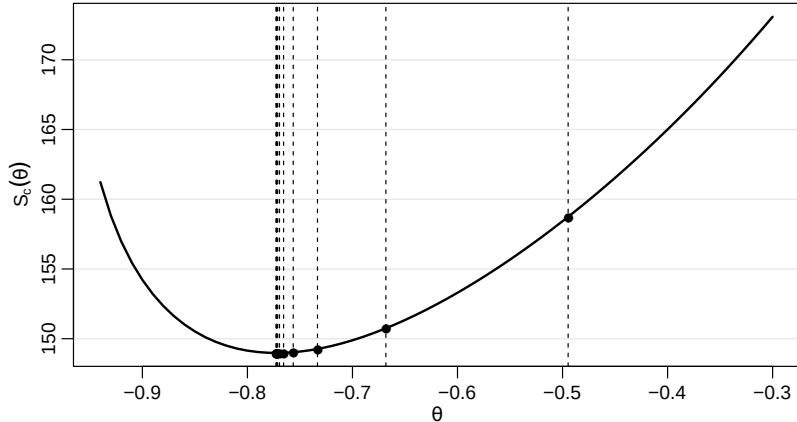


Fig. 3.10. Conditional sum of squares versus values of the moving average parameter for the glacial varve example, [Example 3.33](#). Vertical lines indicate the values of the parameter obtained via Gauss–Newton; see [Table 3.2](#) for the actual values.

that the Gauss–Newton procedure takes large steps toward the minimum initially, and then takes very small steps as it gets close to the minimizing value. When there is only one parameter, as in this case, it would be easy to evaluate $S_c(\theta)$ on a grid of points, and then choose the appropriate value of θ from the grid search. It would be difficult, however, to perform grid searches when there are many parameters.

The following code was used in this example.

```
x = diff(log(varve))
# Evaluate Sc on a Grid
c(0) -> w -> z
c() -> Sc -> Sz -> Szw
num = length(x)
th = seq(-.3,-.94,-.01)
for (p in 1:length(th)){
  for (i in 2:num){ w[i] = x[i]-th[p]*w[i-1] }
  Sc[p] = sum(w^2) }
plot(th, Sc, type="l", ylab=expression(S[c](theta)), xlab=expression(theta),
      lwd=2)
# Gauss-Newton Estimation
r = acf(x, lag=1, plot=FALSE)$acf[-1]
rstart = (1-sqrt(1-4*(r^2)))/(2*r)      # from (3.105)
c(0) -> w -> z
c() -> Sc -> Sz -> Szw -> para
niter = 12
para[1] = rstart
for (p in 1:niter){
  for (i in 2:num){ w[i] = x[i]-para[p]*w[i-1]
    z[i] = w[i-1]-para[p]*z[i-1] }
  Sc[p] = sum(w^2)
  Sz[p] = sum(z^2)
  Szw[p] = sum(z*w)
  para[p+1] = para[p] + Szw[p]/Sz[p] }
```

Table 3.2. Gauss–Newton Results for Example 3.33

j	$\theta_{(j)}$	$S_c(\theta_{(j)})$	$\sum_{t=1}^n z_t^2(\theta_{(j)})$
0	-0.495	158.739	171.240
1	-0.668	150.747	235.266
2	-0.733	149.264	300.562
3	-0.756	149.031	336.823
4	-0.766	148.990	354.173
5	-0.769	148.982	362.167
6	-0.771	148.980	365.801
7	-0.772	148.980	367.446
8	-0.772	148.980	368.188
9	-0.772	148.980	368.522
10	-0.773	148.980	368.673
11	-0.773	148.980	368.741

```
round(cbind(iteration=0:(niter-1), thethat=para[1:niter] , Sc , Sz ), 3)
abline(v = para[1:12], lty=2)
points(para[1:12], Sc[1:12], pch=16)
```

In the general case of causal and invertible ARMA(p, q) models, maximum likelihood estimation and conditional and unconditional least squares estimation (and Yule–Walker estimation in the case of AR models) all lead to optimal estimators. The proof of this general result can be found in a number of texts on theoretical time series analysis (for example, Brockwell and Davis, 1991, or Hannan, 1970, to mention a few). We will denote the ARMA coefficient parameters by $\beta = (\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q)'$.

Property 3.10 Large Sample Distribution of the Estimators

Under appropriate conditions, for causal and invertible ARMA processes, the maximum likelihood, the unconditional least squares, and the conditional least squares estimators, each initialized by the method of moments estimator, all provide optimal estimators of σ_w^2 and β , in the sense that $\hat{\sigma}_w^2$ is consistent, and the asymptotic distribution of $\hat{\beta}$ is the best asymptotic normal distribution. In particular, as $n \rightarrow \infty$,

$$\sqrt{n} (\hat{\beta} - \beta) \xrightarrow{d} N\left(0, \sigma_w^2 \Gamma_{p,q}^{-1}\right). \quad (3.132)$$

The asymptotic variance–covariance matrix of the estimator $\hat{\beta}$ is the inverse of the information matrix. In particular, the $(p+q) \times (p+q)$ matrix $\Gamma_{p,q}$, has the form

$$\Gamma_{p,q} = \begin{pmatrix} \Gamma_{\phi\phi} & \Gamma_{\phi\theta} \\ \Gamma_{\theta\phi} & \Gamma_{\theta\theta} \end{pmatrix}. \quad (3.133)$$

The $p \times p$ matrix $\Gamma_{\phi\phi}$ is given by (3.100), that is, the ij -th element of $\Gamma_{\phi\phi}$, for $i, j = 1, \dots, p$, is $\gamma_x(i-j)$ from an AR(p) process, $\phi(B)x_t = w_t$. Similarly, $\Gamma_{\theta\theta}$ is a $q \times q$ matrix with the ij -th element, for $i, j = 1, \dots, q$, equal to $\gamma_y(i-j)$ from an AR(q) process, $\theta(B)y_t = w_t$. The $p \times q$ matrix $\Gamma_{\phi\theta} = \{\gamma_{xy}(i-j)\}$, for $i = 1, \dots, p$; $j = 1, \dots, q$; that is, the ij -th element is the cross-covariance between the two AR processes given by $\phi(B)x_t = w_t$ and $\theta(B)y_t = w_t$. Finally, $\Gamma_{\theta\phi} = \Gamma'_{\phi\theta}$ is $q \times p$.

Further discussion of [Property 3.10](#), including a proof for the case of least squares estimators for AR(p) processes, can be found in [Section B.3](#).

Example 3.34 Some Specific Asymptotic Distributions

The following are some specific cases of [Property 3.10](#).

AR(1): $\gamma_x(0) = \sigma_w^2 / (1 - \phi^2)$, so $\sigma_w^2 \Gamma_{1,0}^{-1} = (1 - \phi^2)$. Thus,

$$\hat{\phi} \sim \text{AN} [\phi, n^{-1}(1 - \phi^2)]. \quad (3.134)$$

AR(2): The reader can verify that

$$\gamma_x(0) = \left(\frac{1 - \phi_2}{1 + \phi_2} \right) \frac{\sigma_w^2}{(1 - \phi_2)^2 - \phi_1^2}$$

and $\gamma_x(1) = \phi_1 \gamma_x(0) + \phi_2 \gamma_x(1)$. From these facts, we can compute $\Gamma_{2,0}^{-1}$. In particular, we have

$$\begin{pmatrix} \hat{\phi}_1 \\ \hat{\phi}_2 \end{pmatrix} \sim \text{AN} \left[\begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}, n^{-1} \begin{pmatrix} 1 - \phi_2^2 & -\phi_1(1 + \phi_2) \\ \text{sym} & 1 - \phi_2^2 \end{pmatrix} \right]. \quad (3.135)$$

MA(1): In this case, write $\theta(B)y_t = w_t$, or $y_t + \theta y_{t-1} = w_t$. Then, analogous to the AR(1) case, $\gamma_y(0) = \sigma_w^2 / (1 - \theta^2)$, so $\sigma_w^2 \Gamma_{0,1}^{-1} = (1 - \theta^2)$. Thus,

$$\hat{\theta} \sim \text{AN} [\theta, n^{-1}(1 - \theta^2)]. \quad (3.136)$$

MA(2): Write $y_t + \theta_1 y_{t-1} + \theta_2 y_{t-2} = w_t$, so , analogous to the AR(2) case, we have

$$\begin{pmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \end{pmatrix} \sim \text{AN} \left[\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}, n^{-1} \begin{pmatrix} 1 - \theta_2^2 & \theta_1(1 + \theta_2) \\ \text{sym} & 1 - \theta_2^2 \end{pmatrix} \right]. \quad (3.137)$$

ARMA(1,1): To calculate $\Gamma_{\phi\theta}$, we must find $\gamma_{xy}(0)$, where $x_t - \phi x_{t-1} = w_t$ and $y_t + \theta y_{t-1} = w_t$. We have

$$\begin{aligned} \gamma_{xy}(0) &= \text{cov}(x_t, y_t) = \text{cov}(\phi x_{t-1} + w_t, -\theta y_{t-1} + w_t) \\ &= -\phi\theta\gamma_{xy}(0) + \sigma_w^2. \end{aligned}$$

Solving, we find, $\gamma_{xy}(0) = \sigma_w^2 / (1 + \phi\theta)$. Thus,

$$\begin{pmatrix} \hat{\phi} \\ \hat{\theta} \end{pmatrix} \sim \text{AN} \left[\begin{pmatrix} \phi \\ \theta \end{pmatrix}, n^{-1} \begin{pmatrix} (1 - \phi^2)^{-1} & (1 + \phi\theta)^{-1} \\ \text{sym} & (1 - \theta^2)^{-1} \end{pmatrix}^{-1} \right]. \quad (3.138)$$

Example 3.35 Overfitting Caveat

The asymptotic behavior of the parameter estimators gives us an additional insight into the problem of fitting ARMA models to data. For example, suppose a time series follows an AR(1) process and we decide to fit an AR(2) to the data. Do any problems occur in doing this? More generally, why not simply fit large-order AR models to make sure that we capture the dynamics of the process? After all,

if the process is truly an AR(1), the other autoregressive parameters will not be significant. The answer is that if we *overfit*, we obtain less efficient, or less precise parameter estimates. For example, if we fit an AR(1) to an AR(1) process, for large n , $\text{var}(\hat{\phi}_1) \approx n^{-1}(1 - \phi_1^2)$. But, if we fit an AR(2) to the AR(1) process, for large n , $\text{var}(\hat{\phi}_1) \approx n^{-1}(1 - \phi_2^2) = n^{-1}$ because $\phi_2 = 0$. Thus, the variance of ϕ_1 has been inflated, making the estimator less precise.

We do want to mention, however, that overfitting can be used as a diagnostic tool. For example, if we fit an AR(2) model to the data and are satisfied with that model, then adding one more parameter and fitting an AR(3) should lead to approximately the same model as in the AR(2) fit. We will discuss model diagnostics in more detail in [Section 3.7](#).

The reader might wonder, for example, why the asymptotic distributions of $\hat{\phi}$ from an AR(1) and $\hat{\theta}$ from an MA(1) are of the same form; compare [\(3.134\)](#) to [\(3.136\)](#). It is possible to explain this unexpected result heuristically using the intuition of linear regression. That is, for the normal regression model presented in [Section 2.1](#) with no intercept term, $x_t = \beta z_t + w_t$, we know $\hat{\beta}$ is normally distributed with mean β , and from [\(2.6\)](#),

$$\text{var}\left\{\sqrt{n}(\hat{\beta} - \beta)\right\} = n\sigma_w^2 \left(\sum_{t=1}^n z_t^2\right)^{-1} = \sigma_w^2 \left(n^{-1} \sum_{t=1}^n z_t^2\right)^{-1}.$$

For the causal AR(1) model given by $x_t = \phi x_{t-1} + w_t$, the intuition of regression tells us to expect that, for n large,

$$\sqrt{n}(\hat{\phi} - \phi)$$

is approximately normal with mean zero and with variance given by

$$\sigma_w^2 \left(n^{-1} \sum_{t=2}^n x_{t-1}^2\right)^{-1}.$$

Now, $n^{-1} \sum_{t=2}^n x_{t-1}^2$ is the sample variance (recall that the mean of x_t is zero) of the x_t , so as n becomes large we would expect it to approach $\text{var}(x_t) = \gamma(0) = \sigma_w^2/(1 - \phi^2)$. Thus, the large sample variance of $\sqrt{n}(\hat{\phi} - \phi)$ is

$$\sigma_w^2 \gamma_x(0)^{-1} = \sigma_w^2 \left(\frac{\sigma_w^2}{1 - \phi^2}\right)^{-1} = (1 - \phi^2);$$

that is, [\(3.134\)](#) holds.

In the case of an MA(1), we may use the discussion of [Example 3.32](#) to write an approximate regression model for the MA(1). That is, consider the approximation [\(3.130\)](#) as the regression model

$$z_t(\hat{\theta}) = -\theta z_{t-1}(\hat{\theta}) + w_{t-1},$$

where now, $z_{t-1}(\hat{\theta})$ as defined in [Example 3.32](#), plays the role of the regressor. Continuing with the analogy, we would expect the asymptotic distribution of $\sqrt{n}(\hat{\theta} - \theta)$ to be normal, with mean zero, and approximate variance

$$\sigma_w^2 \left(n^{-1} \sum_{t=2}^n z_{t-1}^2(\hat{\theta}) \right)^{-1}.$$

As in the AR(1) case, $n^{-1} \sum_{t=2}^n z_{t-1}^2(\hat{\theta})$ is the sample variance of the $z_t(\hat{\theta})$ so, for large n , this should be $\text{var}\{z_t(\theta)\} = \gamma_z(0)$, say. But note, as seen from [\(3.130\)](#), $z_t(\theta)$ is approximately an AR(1) process with parameter $-\theta$. Thus,

$$\sigma_w^2 \gamma_z(0)^{-1} = \sigma_w^2 \left(\frac{\sigma_w^2}{1 - (-\theta)^2} \right)^{-1} = (1 - \theta^2),$$

which agrees with [\(3.136\)](#). Finally, the asymptotic distributions of the AR parameter estimates and the MA parameter estimates are of the same form because in the MA case, the “regressors” are the differential processes $z_t(\theta)$ that have AR structure, and it is this structure that determines the asymptotic variance of the estimators. For a rigorous account of this approach for the general case, see [Fuller \(1996, Theorem 5.5.4\)](#).

In [Example 3.33](#), the estimated standard error of $\hat{\theta}$ was .025. In that example, we used regression results to estimate the standard error as the square root of

$$n^{-1} \hat{\sigma}_w^2 \left(n^{-1} \sum_{t=1}^n z_t^2(\hat{\theta}) \right)^{-1} = \frac{\hat{\sigma}_w^2}{\sum_{t=1}^n z_t^2(\hat{\theta})},$$

where $n = 632$, $\hat{\sigma}_w^2 = .236$, $\sum_{t=1}^n z_t^2(\hat{\theta}) = 368.74$ and $\hat{\theta} = -.773$. Using [\(3.136\)](#), we could have also calculated this value using the asymptotic approximation, the square root of $(1 - (-.773)^2)/632$, which is also .025.

If n is small, or if the parameters are close to the boundaries, the asymptotic approximations can be quite poor. The *bootstrap* can be helpful in this case; for a broad treatment of the bootstrap, see [Efron and Tibshirani \(1994\)](#). We discuss the case of an AR(1) here and leave the general discussion for [Chapter 6](#). For now, we give a simple example of the bootstrap for an AR(1) process.

Example 3.36 Bootstrapping an AR(1)

We consider an AR(1) model with a regression coefficient near the boundary of causality and an error process that is symmetric but not normal. Specifically, consider the causal model

$$x_t = \mu + \phi(x_{t-1} - \mu) + w_t, \quad (3.139)$$

where $\mu = 50$, $\phi = .95$, and w_t are iid double exponential (Laplace) with location zero, and scale parameter $\beta = 2$. The density of w_t is given by

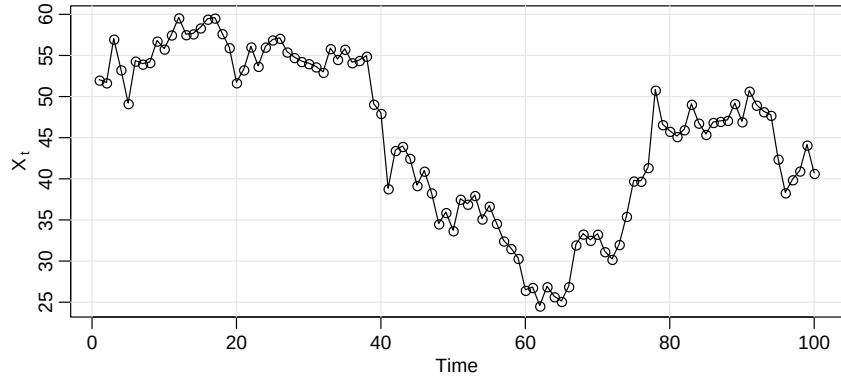


Fig. 3.11. One hundred observations generated from the model in Example 3.36.

$$f(w) = \frac{1}{2\beta} \exp \{-|w|/\beta\} \quad -\infty < w < \infty.$$

In this example, $E(w_t) = 0$ and $\text{var}(w_t) = 2\beta^2 = 8$. Figure 3.11 shows $n = 100$ simulated observations from this process. This particular realization is interesting; the data look like they were generated from a nonstationary process with three different mean levels. In fact, the data were generated from a well-behaved, albeit non-normal, stationary and causal model. To show the advantages of the bootstrap, we will act as if we do not know the actual error distribution. The data in Figure 3.11 were generated as follows.

```
set.seed(101010)
e = rexp(150, rate=.5); u = runif(150, -1, 1); de = e*sign(u)
dex = 50 + arima.sim(n=100, list(ar=.95), innov=de, n.start=50)
plot.ts(dex, type='o', ylab=expression(X[~t]))
```

Using these data, we obtained the Yule–Walker estimates $\hat{\mu} = 45.25$, $\hat{\phi} = .96$, and $\hat{\sigma}_w^2 = 7.88$, as follows.

```
fit = ar.yw(dex, order=1)
round(cbind(fit$x.mean, fit$ar, fit$var.pred), 2)
[1,] 45.25 0.96 7.88
```

To assess the finite sample distribution of $\hat{\phi}$ when $n = 100$, we simulated 1000 realizations of this AR(1) process and estimated the parameters via Yule–Walker. The finite sampling density of the Yule–Walker estimate of ϕ , based on the 1000 repeated simulations, is shown in Figure 3.12. Based on Property 3.10, we would say that $\hat{\phi}$ is approximately normal with mean ϕ (which we supposedly do not know) and variance $(1 - \phi^2)/100$, which we would approximate by $(1 - .96^2)/100 = .03^2$; this distribution is superimposed on Figure 3.12. Clearly the sampling distribution is not close to normality for this sample size. The R code to perform the simulation is as follows. We use the results at the end of the example

```
set.seed(111)
phi.yw = rep(NA, 1000)
for (i in 1:1000){
```

```
e = rexp(150, rate=.5); u = runif(150,-1,1); de = e*sign(u)
x = 50 + arima.sim(n=100,list(ar=.95), innov=de, n.start=50)
phi.yw[i] = ar.yw(x, order=1)$ar }
```

The preceding simulation required full knowledge of the model, the parameter values and the noise distribution. Of course, in a sampling situation, we would not have the information necessary to do the preceding simulation and consequently would not be able to generate a figure like Figure 3.12. The bootstrap, however, gives us a way to attack the problem.

To simplify the discussion and the notation, we condition on x_1 throughout the example. In this case, the one-step-ahead predictors have a simple form,

$$x_t^{t-1} = \mu + \phi(x_{t-1} - \mu), \quad t = 2, \dots, 100.$$

Consequently, the innovations, $\epsilon_t = x_t - x_t^{t-1}$, are given by

$$\epsilon_t = (x_t - \mu) - \phi(x_{t-1} - \mu), \quad t = 2, \dots, 100, \quad (3.140)$$

each with MSPE $P_t^{t-1} = E(\epsilon_t^2) = E(w_t^2) = \sigma_w^2$ for $t = 2, \dots, 100$. We can use (3.140) to write the model in terms of the innovations,

$$x_t = x_t^{t-1} + \epsilon_t = \mu + \phi(x_{t-1} - \mu) + \epsilon_t \quad t = 2, \dots, 100. \quad (3.141)$$

To perform the bootstrap simulation, we replace the parameters with their estimates in (3.141), that is, $\hat{\mu} = 45.25$ and $\hat{\phi} = .96$, and denote the resulting sample innovations as $\{\hat{\epsilon}_2, \dots, \hat{\epsilon}_{100}\}$. To obtain one bootstrap sample, first randomly sample, with replacement, $n = 99$ values from the set of sample innovations; call the sampled values $\{\epsilon_2^*, \dots, \epsilon_{100}^*\}$. Now, generate a bootstrapped data set sequentially by setting

$$x_t^* = 45.25 + .96(x_{t-1}^* - 45.25) + \epsilon_t^*, \quad t = 2, \dots, 100. \quad (3.142)$$

with x_1^* held fixed at x_1 . Next, estimate the parameters as if the data were x_t^* . Call these estimates $\hat{\mu}(1)$, $\hat{\phi}(1)$, and $\sigma_w^2(1)$. Repeat this process a large number, B , of times, generating a collection of bootstrapped parameter estimates, $\{\hat{\mu}(b), \hat{\phi}(b), \sigma_w^2(b); b = 1, \dots, B\}$. We can then approximate the finite sample distribution of an estimator from the bootstrapped parameter values. For example, we can approximate the distribution of $\hat{\phi} - \phi$ by the empirical distribution of $\hat{\phi}(b) - \hat{\phi}$, for $b = 1, \dots, B$.

Figure 3.12 shows the bootstrap histogram of 500 bootstrapped estimates of ϕ using the data shown in Figure 3.11. Note that the bootstrap distribution of $\hat{\phi}$ is close to the distribution of $\hat{\phi}$ shown in Figure 3.12. The following code was used to perform the bootstrap.

```
set.seed(666) # not that 666
fit = ar.yw(dex, order=1) # assumes the data were retained
m = fit$mean # estimate of mean
phi = fit$ar # estimate of phi
nboot = 500 # number of bootstrap replicates
resids = fit$resid[-1] # the 99 innovations
```

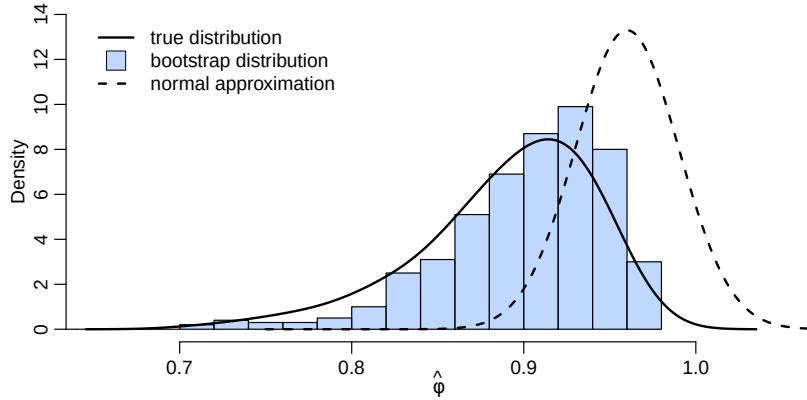


Fig. 3.12. Finite sample density of the Yule–Walker estimate of ϕ (solid line) in Example 3.36 and the corresponding asymptotic normal density (dashed line). Bootstrap histogram of $\hat{\phi}$ based on 500 bootstrapped samples.

```

x.star = dex                      # initialize x*
phi.star.yw = rep(NA, nboot)
# Bootstrap
for (i in 1:nboot) {
  resid.star = sample(resids, replace=TRUE)
  for (t in 1:99){ x.star[t+1] = m + phi*(x.star[t]-m) + resid.star[t] }
  phi.star.yw[i] = ar.yw(x.star, order=1)$ar
}
# Picture
culer = rgb(.5,.7,1,.5)
hist(phi.star.yw, 15, main="", prob=TRUE, xlim=c(.65,1.05), ylim=c(0,14),
      col=culer, xlab=expression(hat(phi)))
lines(density(phi.yw, bw=.02), lwd=2)  # from previous simulation
u = seq(.75, 1.1, by=.001)           # normal approximation
lines(u, dnorm(u, mean=.96, sd=.03), lty=2, lwd=2)
legend(.65, 14, legend=c('true distribution', 'bootstrap distribution',
                        'normal approximation'), bty='n', lty=c(1,0,2), lwd=c(2,0,2),
                        col=1, pch=c(NA,22,NA), pt.bg=c(NA,culer,NA), pt.cex=2.5)

```

3.6 Integrated Models for Nonstationary Data

In Chapter 1 and Chapter 2, we saw that if x_t is a random walk, $x_t = x_{t-1} + w_t$, then by differencing x_t , we find that $\nabla x_t = w_t$ is stationary. In many situations, time series can be thought of as being composed of two components, a nonstationary trend component and a zero-mean stationary component. For example, in Section 2.1 we considered the model

$$x_t = \mu_t + y_t, \quad (3.143)$$

where $\mu_t = \beta_0 + \beta_1 t$ and y_t is stationary. Differencing such a process will lead to a stationary process:

$$\nabla x_t = x_t - x_{t-1} = \beta_1 + y_t - y_{t-1} = \beta_1 + \nabla y_t.$$

Another model that leads to first differencing is the case in which μ_t in (3.143) is stochastic and slowly varying according to a random walk. That is,

$$\mu_t = \mu_{t-1} + v_t$$

where v_t is stationary. In this case,

$$\nabla x_t = v_t + \nabla y_t,$$

is stationary. If μ_t in (3.143) is a k -th order polynomial, $\mu_t = \sum_{j=0}^k \beta_j t^j$, then (Problem 3.27) the differenced series $\nabla^k x_t$ is stationary. Stochastic trend models can also lead to higher order differencing. For example, suppose

$$\mu_t = \mu_{t-1} + v_t \quad \text{and} \quad v_t = v_{t-1} + e_t,$$

where e_t is stationary. Then, $\nabla x_t = v_t + \nabla y_t$ is not stationary, but

$$\nabla^2 x_t = e_t + \nabla^2 y_t$$

is stationary.

The *integrated* ARMA, or ARIMA, model is a broadening of the class of ARMA models to include differencing.

Definition 3.11 A process x_t is said to be **ARIMA**(p, d, q) if

$$\nabla^d x_t = (1 - B)^d x_t$$

is ARMA(p, q). In general, we will write the model as

$$\phi(B)(1 - B)^d x_t = \theta(B)w_t. \quad (3.144)$$

If $E(\nabla^d x_t) = \mu$, we write the model as

$$\phi(B)(1 - B)^d x_t = \delta + \theta(B)w_t,$$

where $\delta = \mu(1 - \phi_1 - \cdots - \phi_p)$.

Because of the nonstationarity, care must be taken when deriving forecasts. For the sake of completeness, we discuss this issue briefly here, but we stress the fact that both the theoretical and computational aspects of the problem are best handled via state-space models. We discuss the theoretical details in Chapter 6. For information on the state-space based computational aspects in R, see the ARIMA help files (`?arima` and `?predict.Arima`); our scripts `sarima` and `sarima.for` are basically wrappers for these R scripts.

It should be clear that, since $y_t = \nabla^d x_t$ is ARMA, we can use Section 3.4 methods to obtain forecasts of y_t , which in turn lead to forecasts for x_t . For example, if $d = 1$, given forecasts y_{n+m}^n for $m = 1, 2, \dots$, we have $y_{n+m}^n = x_{n+m}^n - x_{n+m-1}^n$, so that

$$x_{n+m}^n = y_{n+m}^n + x_{n+m-1}^n$$

with initial condition $x_{n+1}^n = y_{n+1}^n + x_n$ (noting $x_n^n = x_n$).

It is a little more difficult to obtain the prediction errors P_{n+m}^n , but for large n , the approximation used in [Section 3.4](#), equation (3.86), works well. That is, the mean-squared prediction error can be approximated by

$$P_{n+m}^n = \sigma_w^2 \sum_{j=0}^{m-1} \psi_j^{*2}, \quad (3.145)$$

where ψ_j^* is the coefficient of z^j in $\psi^*(z) = \theta(z)/\phi(z)(1-z)^d$.

To better understand integrated models, we examine the properties of some simple cases; [Problem 3.29](#) covers the ARIMA(1, 1, 0) case.

Example 3.37 Random Walk with Drift

To fix ideas, we begin by considering the random walk with drift model first presented in [Example 1.11](#), that is,

$$x_t = \delta + x_{t-1} + w_t,$$

for $t = 1, 2, \dots$, and $x_0 = 0$. Technically, the model is not ARIMA, but we could include it trivially as an ARIMA(0, 1, 0) model. Given data x_1, \dots, x_n , the one-step-ahead forecast is given by

$$x_{n+1}^n = E(x_{n+1} \mid x_n, \dots, x_1) = E(\delta + x_n + w_{n+1} \mid x_n, \dots, x_1) = \delta + x_n.$$

The two-step-ahead forecast is given by $x_{n+2}^n = \delta + x_{n+1}^n = 2\delta + x_n$, and consequently, the m -step-ahead forecast, for $m = 1, 2, \dots$, is

$$x_{n+m}^n = m\delta + x_n, \quad (3.146)$$

To obtain the forecast errors, it is convenient to recall equation (1.4); i.e., $x_n = n\delta + \sum_{j=1}^n w_j$, in which case we may write

$$x_{n+m} = (n+m)\delta + \sum_{j=1}^{n+m} w_j = m\delta + x_n + \sum_{j=n+1}^{n+m} w_j.$$

From this it follows that the m -step-ahead prediction error is given by

$$P_{n+m}^n = E(x_{n+m} - x_{n+m}^n)^2 = E\left(\sum_{j=n+1}^{n+m} w_j\right)^2 = m\sigma_w^2. \quad (3.147)$$

Hence, unlike the stationary case (see [Example 3.23](#)), as the forecast horizon grows, the prediction errors, (3.147), increase without bound and the forecasts follow a straight line with slope δ emanating from x_n . We note that (3.145) is exact in this case because $\psi^*(z) = 1/(1-z) = \sum_{j=0}^{\infty} z^j$ for $|z| < 1$, so that $\psi_j^* = 1$ for all j .

The w_t are Gaussian, so estimation is straightforward because the differenced data, say $y_t = \nabla x_t$, are independent and identically distributed normal variates with mean δ and variance σ_w^2 . Consequently, optimal estimates of δ and σ_w^2 are the sample mean and variance of the y_t , respectively.

Example 3.38 IMA(1,1) and EWMA

The ARIMA(0,1,1), or IMA(1,1) model is of interest because many economic time series can be successfully modeled this way. In addition, the model leads to a frequently used, and abused, forecasting method called exponentially weighted moving averages (EWMA). We will write the model as

$$x_t = x_{t-1} + w_t - \lambda w_{t-1}, \quad (3.148)$$

with $|\lambda| < 1$, for $t = 1, 2, \dots$, and $x_0 = 0$, because this model formulation is easier to work with here, and it leads to the standard representation for EWMA. We could have included a drift term in (3.148), as was done in the previous example, but for the sake of simplicity, we leave it out of the discussion. If we write

$$y_t = w_t - \lambda w_{t-1},$$

we may write (3.148) as $x_t = x_{t-1} + y_t$. Because $|\lambda| < 1$, y_t has an invertible representation, $y_t = \sum_{j=1}^{\infty} \lambda^j y_{t-j} + w_t$, and substituting $y_t = x_t - x_{t-1}$, we may write

$$x_t = \sum_{j=1}^{\infty} (1 - \lambda) \lambda^{j-1} x_{t-j} + w_t. \quad (3.149)$$

as an approximation for large t (put $x_t = 0$ for $t \leq 0$). Verification of (3.149) is left to the reader (Problem 3.28). Using the approximation (3.149), we have that the approximate one-step-ahead predictor, using the notation of Section 3.4, is

$$\begin{aligned} \tilde{x}_{n+1} &= \sum_{j=1}^{\infty} (1 - \lambda) \lambda^{j-1} x_{n+1-j} \\ &= (1 - \lambda)x_n + \lambda \sum_{j=1}^{\infty} (1 - \lambda) \lambda^{j-1} x_{n-j} \\ &= (1 - \lambda)x_n + \lambda \tilde{x}_n. \end{aligned} \quad (3.150)$$

From (3.150), we see that the new forecast is a linear combination of the old forecast and the new observation. Based on (3.150) and the fact that we only observe x_1, \dots, x_n , and consequently y_1, \dots, y_n (because $y_t = x_t - x_{t-1}$; $x_0 = 0$), the truncated forecasts are

$$\tilde{x}_{n+1}^n = (1 - \lambda)x_n + \lambda \tilde{x}_n^{n-1}, \quad n \geq 1, \quad (3.151)$$

with $\tilde{x}_1^0 = x_1$ as an initial value. The mean-square prediction error can be approximated using (3.145) by noting that $\psi^*(z) = (1 - \lambda z)/(1 - z) = 1 + (1 - \lambda) \sum_{j=1}^{\infty} z^j$ for $|z| < 1$; consequently, for large n , (3.145) leads to

$$P_{n+m}^n \approx \sigma_w^2 [1 + (m-1)(1-\lambda)^2].$$

In EWMA, the parameter $1 - \lambda$ is often called the smoothing parameter and is restricted to be between zero and one. Larger values of λ lead to smoother forecasts.

This method of forecasting is popular because it is easy to use; we need only retain the previous forecast value and the current observation to forecast the next time period. Unfortunately, as previously suggested, the method is often abused because some forecasters do not verify that the observations follow an IMA(1, 1) process, and often arbitrarily pick values of λ . In the following, we show how to generate 100 observations from an IMA(1,1) model with $\lambda = -\theta = .8$ and then calculate and display the fitted EWMA superimposed on the data. This is accomplished using the Holt-Winters command in R (see the help file `?HoltWinters` for details; no output is shown):

```
set.seed(666)
x = arima.sim(list(order = c(0,1,1), ma = -0.8), n = 100)
(x.ima = HoltWinters(x, beta=FALSE, gamma=FALSE)) # alpha below is 1 - lambda
  Smoothing parameter: alpha:  0.1663072
plot(x.ima)
```

3.7 Building ARIMA Models

There are a few basic steps to fitting ARIMA models to time series data. These steps involve

- plotting the data,
- possibly transforming the data,
- identifying the dependence orders of the model,
- parameter estimation,
- diagnostics, and
- model choice.

First, as with any data analysis, we should construct a time plot of the data, and inspect the graph for any anomalies. If, for example, the variability in the data grows with time, it will be necessary to transform the data to stabilize the variance. In such cases, the Box–Cox class of power transformations, equation (2.34), could be employed. Also, the particular application might suggest an appropriate transformation. For example, we have seen numerous examples where the data behave as $x_t = (1 + p_t)x_{t-1}$, where p_t is a small percentage change from period $t - 1$ to t , which may be negative. If p_t is a relatively stable process, then $\nabla \log(x_t) \approx p_t$ will be relatively stable. Frequently, $\nabla \log(x_t)$ is called the *return* or *growth rate*. This general idea was used in [Example 3.33](#), and we will use it again in [Example 3.39](#).

After suitably transforming the data, the next step is to identify preliminary values of the autoregressive order, p , the order of differencing, d , and the moving average order, q . A time plot of the data will typically suggest whether any differencing is needed. If differencing is called for, then difference the data once, $d = 1$, and inspect the time plot of ∇x_t . If additional differencing is necessary, then try differencing again and inspect a time plot of $\nabla^2 x_t$. Be careful not to overdifference because this may introduce dependence where none exists. For example, $x_t = w_t$ is serially uncorrelated, but $\nabla x_t = w_t - w_{t-1}$ is MA(1). In addition to time plots, the sample

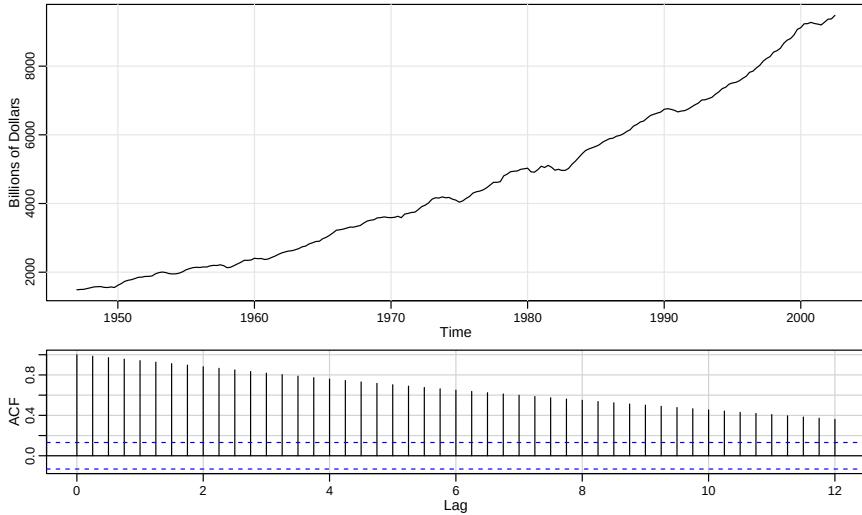


Fig. 3.13. Top: *Quarterly U.S. GNP from 1947(1) to 2002(3)*. Bottom: *Sample ACF of the GNP data. Lag is in terms of years.*

ACF can help in indicating whether differencing is needed. Because the polynomial $\phi(z)(1 - z)^d$ has a unit root, the sample ACF, $\hat{\rho}(h)$, will not decay to zero fast as h increases. Thus, a slow decay in $\hat{\rho}(h)$ is an indication that differencing may be needed.

When preliminary values of d have been settled, the next step is to look at the sample ACF and PACF of $\nabla^d x_t$ for whatever values of d have been chosen. Using [Table 3.1](#) as a guide, preliminary values of p and q are chosen. Note that it cannot be the case that both the ACF and PACF cut off. Because we are dealing with estimates, it will not always be clear whether the sample ACF or PACF is tailing off or cutting off. Also, two models that are seemingly different can actually be very similar. With this in mind, we should not worry about being so precise at this stage of the model fitting. At this point, a few preliminary values of p , d , and q should be at hand, and we can start estimating the parameters.

Example 3.39 Analysis of GNP Data

In this example, we consider the analysis of quarterly U.S. GNP from 1947(1) to 2002(3), $n = 223$ observations. The data are real U.S. gross national product in billions of chained 1996 dollars and have been seasonally adjusted. The data were obtained from the Federal Reserve Bank of St. Louis (<http://research.stlouisfed.org/>). [Figure 3.13](#) shows a plot of the data, say, y_t . Because strong trend tends to obscure other effects, it is difficult to see any other variability in data except for periodic large dips in the economy. When reports of GNP and similar economic indicators are given, it is often in growth rate (percent change) rather than in actual (or adjusted) values that is of interest. The growth rate, say, $x_t = \nabla \log(y_t)$, is plotted in [Figure 3.14](#), and it appears to be a stable process.

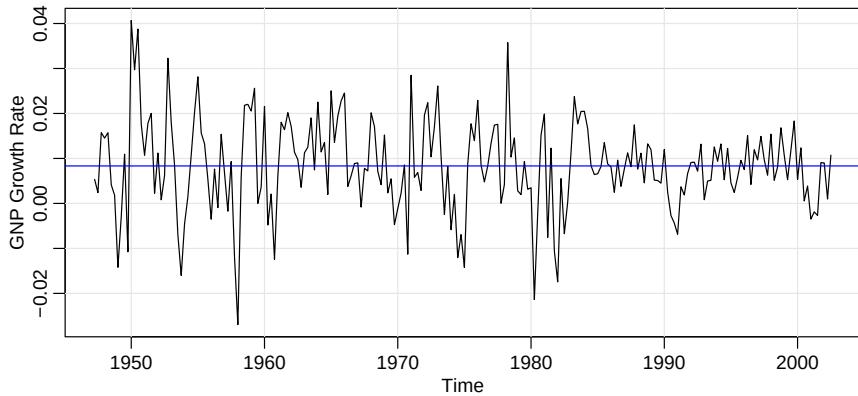


Fig. 3.14. U.S. GNP quarterly growth rate. The horizontal line displays the average growth of the process, which is close to 1%.

The sample ACF and PACF of the quarterly growth rate are plotted in [Figure 3.15](#). Inspecting the sample ACF and PACF, we might feel that the ACF is cutting off at lag 2 and the PACF is tailing off. This would suggest the GNP growth rate follows an MA(2) process, or log GNP follows an ARIMA(0, 1, 2) model. Rather than focus on one model, we will also suggest that it appears that the ACF is tailing off and the PACF is cutting off at lag 1. This suggests an AR(1) model for the growth rate, or ARIMA(1, 1, 0) for log GNP. As a preliminary analysis, we will fit both models.

Using MLE to fit the MA(2) model for the growth rate, x_t , the estimated model is

$$\hat{x}_t = .008_{(.001)} + .303_{(.065)}\hat{w}_{t-1} + .204_{(.064)}\hat{w}_{t-2} + \hat{w}_t, \quad (3.152)$$

where $\hat{\sigma}_w = .0094$ is based on 219 degrees of freedom. The values in parentheses are the corresponding estimated standard errors. All of the regression coefficients are significant, including the constant. *We make a special note of this because, as a default, some computer packages do not fit a constant in a differenced model.* That is, these packages assume, by default, that there is no drift. In this example, not including a constant leads to the wrong conclusions about the nature of the U.S. economy. Not including a constant assumes the average quarterly growth rate is zero, whereas the U.S. GNP average quarterly growth rate is about 1% (which can be seen easily in [Figure 3.14](#)). We leave it to the reader to investigate what happens when the constant is not included.

The estimated AR(1) model is

$$\hat{x}_t = .008_{(.001)}(1 - .347) + .347_{(.063)}\hat{x}_{t-1} + \hat{w}_t, \quad (3.153)$$

where $\hat{\sigma}_w = .0095$ on 220 degrees of freedom; note that the constant in (3.153) is $.008(1 - .347) = .005$.

We will discuss diagnostics next, but assuming both of these models fit well, how are we to reconcile the apparent differences of the estimated models (3.152)

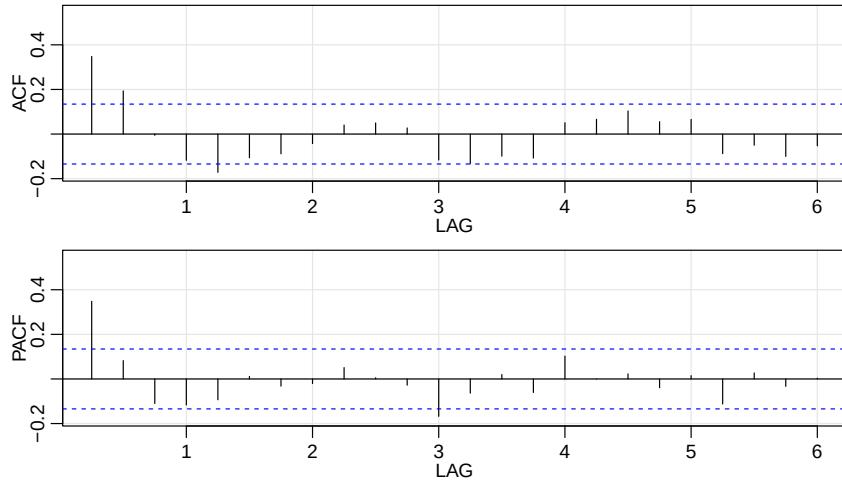


Fig. 3.15. Sample ACF and PACF of the GNP quarterly growth rate. Lag is in terms of years.

and (3.153)? In fact, the fitted models are nearly the same. To show this, consider an AR(1) model of the form in (3.153) without a constant term; that is,

$$x_t = .35x_{t-1} + w_t,$$

and write it in its causal form, $x_t = \sum_{j=0}^{\infty} \psi_j w_{t-j}$, where we recall $\psi_j = .35^j$. Thus, $\psi_0 = 1, \psi_1 = .350, \psi_2 = .123, \psi_3 = .043, \psi_4 = .015, \psi_5 = .005, \psi_6 = .002, \psi_7 = .001, \psi_8 = 0, \psi_9 = 0, \psi_{10} = 0$, and so forth. Thus,

$$x_t \approx .35w_{t-1} + .12w_{t-2} + w_t,$$

which is similar to the fitted MA(2) model in (3.153).

The analysis can be performed in R as follows.

```
plot(gnp)
acf2(gnp, 50)
gnpgr = diff(log(gnp)) # growth rate
plot(gnpgr)
acf2(gnpgr, 24)
sarima(gnpgr, 1, 0, 0) # AR(1)
sarima(gnpgr, 0, 0, 2) # MA(2)
ARMAtoMA(ar=.35, ma=0, 10) # prints psi-weights
```

The next step in model fitting is diagnostics. This investigation includes the analysis of the residuals as well as model comparisons. Again, the first step involves a time plot of the *innovations* (or residuals), $x_t - \hat{x}_t^{t-1}$, or of the *standardized innovations*

$$e_t = (x_t - \hat{x}_t^{t-1}) / \sqrt{\hat{P}_t^{t-1}}, \quad (3.154)$$

where \hat{x}_t^{t-1} is the one-step-ahead prediction of x_t based on the fitted model and \hat{P}_t^{t-1} is the estimated one-step-ahead error variance. If the model fits well, the standardized

residuals should behave as an iid sequence with mean zero and variance one. The time plot should be inspected for any obvious departures from this assumption. Unless the time series is Gaussian, it is not enough that the residuals are uncorrelated. For example, it is possible in the non-Gaussian case to have an uncorrelated process for which values contiguous in time are highly dependent. As an example, we mention the family of GARCH models that are discussed in [Chapter 5](#).

Investigation of marginal normality can be accomplished visually by looking at a histogram of the residuals. In addition to this, a normal probability plot or a Q-Q plot can help in identifying departures from normality. See Johnson and Wichern (1992, Chapter 4) for details of this test as well as additional tests for multivariate normality.

There are several tests of randomness, for example the runs test, that could be applied to the residuals. We could also inspect the sample autocorrelations of the residuals, say, $\hat{\rho}_e(h)$, for any patterns or large values. Recall that, for a white noise sequence, the sample autocorrelations are approximately independently and normally distributed with zero means and variances $1/n$. Hence, a good check on the correlation structure of the residuals is to plot $\hat{\rho}_e(h)$ versus h along with the error bounds of $\pm 2/\sqrt{n}$. The residuals from a model fit, however, will not quite have the properties of a white noise sequence and the variance of $\hat{\rho}_e(h)$ can be much less than $1/n$. Details can be found in Box and Pierce (1970) and McLeod (1978). This part of the diagnostics can be viewed as a visual inspection of $\hat{\rho}_e(h)$ with the main concern being the detection of obvious departures from the independence assumption.

In addition to plotting $\hat{\rho}_e(h)$, we can perform a general test that takes into consideration the magnitudes of $\hat{\rho}_e(h)$ as a group. For example, it may be the case that, individually, each $\hat{\rho}_e(h)$ is small in magnitude, say, each one is just slightly less than $2/\sqrt{n}$ in magnitude, but, collectively, the values are large. The *Ljung–Box–Pierce Q-statistic* given by

$$Q = n(n + 2) \sum_{h=1}^H \frac{\hat{\rho}_e^2(h)}{n - h} \quad (3.155)$$

can be used to perform such a test. The value H in (3.155) is chosen somewhat arbitrarily, typically, $H = 20$. Under the null hypothesis of model adequacy, asymptotically ($n \rightarrow \infty$), $Q \sim \chi_{H-p-q}^2$. Thus, we would reject the null hypothesis at level α if the value of Q exceeds the $(1 - \alpha)$ -quantile of the χ_{H-p-q}^2 distribution. Details can be found in Box and Pierce (1970), Ljung and Box (1978), and Davies et al. (1977). The basic idea is that if w_t is white noise, then by [Property 1.2](#), $n\hat{\rho}_w^2(h)$, for $h = 1, \dots, H$, are asymptotically independent χ_1^2 random variables. This means that $n \sum_{h=1}^H \hat{\rho}_w^2(h)$ is approximately a χ_H^2 random variable. Because the test involves the ACF of residuals from a model fit, there is a loss of $p + q$ degrees of freedom; the other values in (3.155) are used to adjust the statistic to better match the asymptotic chi-squared distribution.

Example 3.40 Diagnostics for GNP Growth Rate Example

We will focus on the MA(2) fit from [Example 3.39](#); the analysis of the AR(1) residuals is similar. [Figure 3.16](#) displays a plot of the standardized residuals, the ACF of the residuals, a boxplot of the standardized residuals, and the p-values

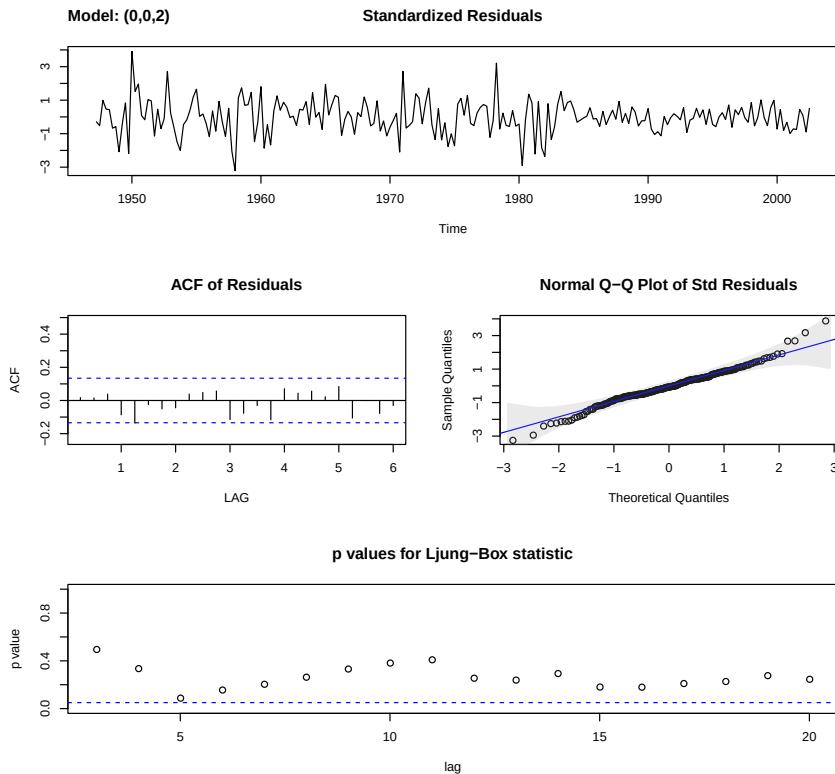


Fig. 3.16. Diagnostics of the residuals from MA(2) fit on GNP growth rate.

associated with the Q-statistic, (3.155), at lags $H = 3$ through $H = 20$ (with corresponding degrees of freedom $H - 2$).

Inspection of the time plot of the standardized residuals in Figure 3.16 shows no obvious patterns. Notice that there may be outliers, with a few values exceeding 3 standard deviations in magnitude. The ACF of the standardized residuals shows no apparent departure from the model assumptions, and the Q-statistic is never significant at the lags shown. The normal Q-Q plot of the residuals shows that the assumption of normality is reasonable, with the exception of the possible outliers.

The model appears to fit well. The diagnostics shown in Figure 3.16 are a by-product of the `sarima` command from the previous example.^{3.8}

^{3.8} The script `tsdiag` is available in R to run diagnostics for an ARIMA object, however, the script has errors and we do not recommend using it.

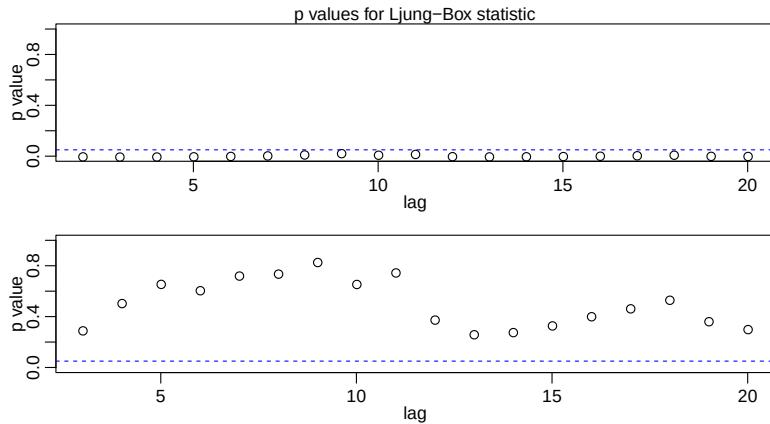


Fig. 3.17. Q-statistic p-values for the ARIMA(0, 1, 1) fit (top) and the ARIMA(1, 1, 1) fit (bottom) to the logged varve data.

Example 3.41 Diagnostics for the Glacial Varve Series

In Example 3.33, we fit an ARIMA(0, 1, 1) model to the logarithms of the glacial varve data and there appears to be a small amount of autocorrelation left in the residuals and the Q-tests are all significant; see Figure 3.17.

To adjust for this problem, we fit an ARIMA(1, 1, 1) to the logged varve data and obtained the estimates

$$\hat{\phi} = .23_{(.05)}, \hat{\theta} = -.89_{(.03)}, \hat{\sigma}_w^2 = .23.$$

Hence the AR term is significant. The Q-statistic p-values for this model are also displayed in Figure 3.17, and it appears this model fits the data well.

As previously stated, the diagnostics are byproducts of the individual `sarima` runs. We note that we did not fit a constant in either model because there is no apparent drift in the differenced, logged varve series. This fact can be verified by noting the constant is not significant when the command `no.constant=TRUE` is removed in the code:

```
sarima(log(varve), 0, 1, 1, no.constant=TRUE) # ARIMA(0, 1, 1)
sarima(log(varve), 1, 1, 1, no.constant=TRUE) # ARIMA(1, 1, 1)
```

In Example 3.39, we have two competing models, an AR(1) and an MA(2) on the GNP growth rate, that each appear to fit the data well. In addition, we might also consider that an AR(2) or an MA(3) might do better for forecasting. Perhaps combining both models, that is, fitting an ARMA(1, 2) to the GNP growth rate, would be the best. As previously mentioned, we have to be concerned with *overfitting* the model; it is not always the case that more is better. Overfitting leads to less-precise estimators, and adding more parameters may fit the data better but may also lead to bad forecasts. This result is illustrated in the following example.

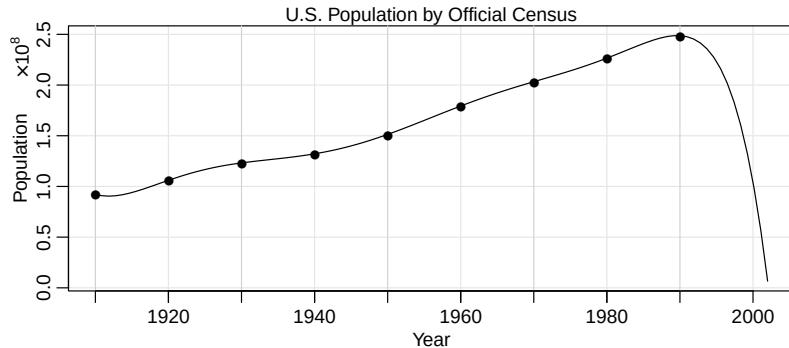


Fig. 3.18. A perfect fit and a terrible forecast.

Example 3.42 A Problem with Overfitting

Figure 3.18 shows the U.S. population by official census, every ten years from 1910 to 1990, as points. If we use these nine observations to predict the future population, we can use an eight-degree polynomial so the fit to the nine observations is perfect. The model in this case is

$$x_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \cdots + \beta_8 t^8 + w_t.$$

The fitted line, which is plotted in the figure, passes through the nine observations. The model predicts that the population of the United States will be close to zero in the year 2000, and will cross zero sometime in the year 2002!

The final step of model fitting is model choice or model selection. That is, we must decide which model we will retain for forecasting. The most popular techniques, AIC, AICc, and BIC, were described in Section 2.1 in the context of regression models.

Example 3.43 Model Choice for the U.S. GNP Series

Returning to the analysis of the U.S. GNP data presented in Example 3.39 and Example 3.40, recall that two models, an AR(1) and an MA(2), fit the GNP growth rate well. To choose the final model, we compare the AIC, the AICc, and the BIC for both models. These values are a byproduct of the `sarima` runs displayed at the end of Example 3.39, but for convenience, we display them again here (recall the growth rate data are in `gnpgr`):

```
sarima(gnpgr, 1, 0, 0) # AR(1)
$AIC: -8.294403 $AICc: -8.284898 $BIC: -9.263748
sarima(gnpgr, 0, 0, 2) # MA(2)
$AIC: -8.297693 $AICc: -8.287854 $BIC: -9.251711
```

The AIC and AICc both prefer the MA(2) fit, whereas the BIC prefers the simpler AR(1) model. It is often the case that the BIC will select a model of smaller order than the AIC or AICc. In either case, it is not unreasonable to retain the AR(1) because pure autoregressive models are easier to work with.

3.8 Regression with Autocorrelated Errors

In [Section 2.1](#), we covered the classical regression model with uncorrelated errors w_t . In this section, we discuss the modifications that might be considered when the errors are correlated. That is, consider the regression model

$$y_t = \sum_{j=1}^r \beta_j z_{tj} + x_t \quad (3.156)$$

where x_t is a process with some covariance function $\gamma_x(s, t)$. In ordinary least squares, the assumption is that x_t is white Gaussian noise, in which case $\gamma_x(s, t) = 0$ for $s \neq t$ and $\gamma_x(t, t) = \sigma^2$, independent of t . If this is not the case, then weighted least squares should be used.

Write the model in vector notation, $y = Z\beta + x$, where $y = (y_1, \dots, y_n)'$ and $x = (x_1, \dots, x_n)'$ are $n \times 1$ vectors, $\beta = (\beta_1, \dots, \beta_r)'$ is $r \times 1$, and $Z = [z_1 | z_2 | \dots | z_n]'$ is the $n \times r$ matrix composed of the input variables. Let $\Gamma = \{\gamma_x(s, t)\}$, then $\Gamma^{-1/2}y = \Gamma^{-1/2}Z\beta + \Gamma^{-1/2}x$, so that we can write the model as

$$y^* = Z^*\beta + \delta,$$

where $y^* = \Gamma^{-1/2}y$, $Z^* = \Gamma^{-1/2}Z$, and $\delta = \Gamma^{-1/2}x$. Consequently, the covariance matrix of δ is the identity and the model is in the classical linear model form. It follows that the weighted estimate of β is $\hat{\beta}_w = (Z^*Z^*)^{-1}Z^{*'}y^* = (Z'\Gamma^{-1}Z)^{-1}Z'\Gamma^{-1}y$, and the variance-covariance matrix of the estimator is $\text{var}(\hat{\beta}_w) = (Z'\Gamma^{-1}Z)^{-1}$. If x_t is white noise, then $\Gamma = \sigma^2 I$ and these results reduce to the usual least squares results.

In the time series case, it is often possible to assume a stationary covariance structure for the error process x_t that corresponds to a linear process and try to find an ARMA representation for x_t . For example, if we have a pure AR(p) error, then

$$\phi(B)x_t = w_t,$$

and $\phi(B) = 1 - \phi_1 B - \dots - \phi_p B^p$ is the linear transformation that, when applied to the error process, produces the white noise w_t . Multiplying the regression equation through by the transformation $\phi(B)$ yields,

$$\underbrace{\phi(B)y_t}_{y_t^*} = \sum_{j=1}^r \beta_j \underbrace{\phi(B)z_{tj}}_{z_{tj}^*} + \underbrace{\phi(B)x_t}_{w_t},$$

and we are back to the linear regression model where the observations have been transformed so that $y_t^* = \phi(B)y_t$ is the dependent variable, $z_{tj}^* = \phi(B)z_{tj}$ for $j = 1, \dots, r$, are the independent variables, but the β s are the same as in the original model. For example, if $p = 1$, then $y_t^* = y_t - \phi y_{t-1}$ and $z_{tj}^* = z_{tj} - \phi z_{t-1,j}$.

In the AR case, we may set up the least squares problem as minimizing the error sum of squares

$$S(\phi, \beta) = \sum_{t=1}^n w_t^2 = \sum_{t=1}^n \left[\phi(B)y_t - \sum_{j=1}^r \beta_j \phi(B)z_{tj} \right]^2$$

with respect to all the parameters, $\phi = \{\phi_1, \dots, \phi_p\}$ and $\beta = \{\beta_1, \dots, \beta_r\}$. Of course, the optimization is performed using numerical methods.

If the error process is ARMA(p, q), i.e., $\phi(B)x_t = \theta(B)w_t$, then in the above discussion, we transform by $\pi(B)x_t = w_t$, where $\pi(B) = \theta(B)^{-1}\phi(B)$. In this case the error sum of squares also depends on $\theta = \{\theta_1, \dots, \theta_q\}$:

$$S(\phi, \theta, \beta) = \sum_{t=1}^n w_t^2 = \sum_{t=1}^n \left[\pi(B)y_t - \sum_{j=1}^r \beta_j \pi(B)z_{tj} \right]^2$$

At this point, the main problem is that we do not typically know the behavior of the noise x_t prior to the analysis. An easy way to tackle this problem was first presented in Cochrane and Orcutt (1949), and with the advent of cheap computing is modernized below:

- (i) First, run an ordinary regression of y_t on z_{t1}, \dots, z_{tr} (acting as if the errors are uncorrelated). Retain the residuals, $\hat{x}_t = y_t - \sum_{j=1}^r \hat{\beta}_j z_{tj}$.
- (ii) Identify ARMA model(s) for the residuals \hat{x}_t .
- (iii) Run weighted least squares (or MLE) on the regression model with autocorrelated errors using the model specified in step (ii).
- (iv) Inspect the residuals \hat{w}_t for whiteness, and adjust the model if necessary.

Example 3.44 Mortality, Temperature and Pollution

We consider the analyses presented in Example 2.2, relating mean adjusted temperature T_t , and particulate levels P_t to cardiovascular mortality M_t . We consider the regression model

$$M_t = \beta_1 + \beta_2 t + \beta_3 T_t + \beta_4 T_t^2 + \beta_5 P_t + x_t, \quad (3.157)$$

where, for now, we assume that x_t is white noise. The sample ACF and PACF of the residuals from the ordinary least squares fit of (3.157) are shown in Figure 3.19, and the results suggest an AR(2) model for the residuals.

Our next step is to fit the correlated error model (3.157), but where x_t is AR(2),

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + w_t$$

and w_t is white noise. The model can be fit using the `sarima` function as follows (partial output shown).

```
trend = time(cmort); temp = temp - mean(temp); temp2 = temp^2
summary(fit <- lm(cmort~trend + temp + temp2 + part, na.action=NULL))
acf2(resid(fit), 52) # implies AR2
sarima(cmort, 2,0,0, xreg=cbind(trend,temp,temp2,part))
Coefficients:
      ar1      ar2  intercept      trend      temp      temp2      part
      0.3848  0.4326   80.2116 -1.5165 -0.0190  0.0154  0.1545
  s.e.  0.0436  0.0400    1.8072  0.4226  0.0495  0.0020  0.0272
sigma^2 estimated as 26.01: loglikelihood = -1549.04, aic = 3114.07
```

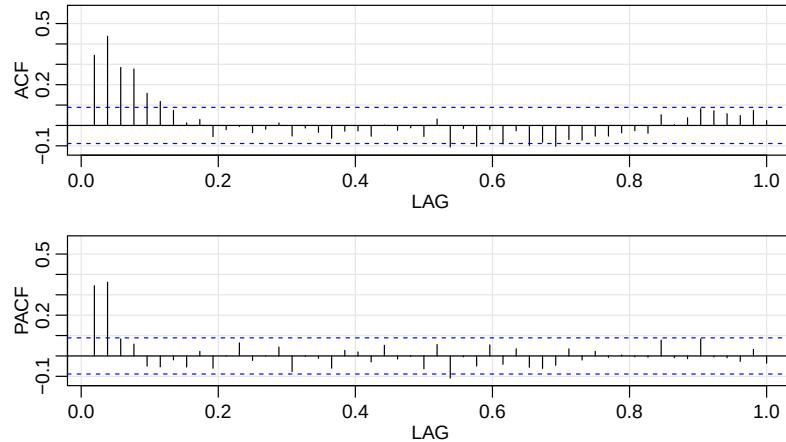


Fig. 3.19. Sample ACF and PACF of the mortality residuals indicating an AR(2) process.

The residual analysis output from `sarima` (not shown) shows no obvious departure of the residuals from whiteness.

Example 3.45 Regression with Lagged Variables (cont)

In Example 2.9 we fit the model

$$R_t = \beta_0 + \beta_1 S_{t-6} + \beta_2 D_{t-6} + \beta_3 D_{t-6} S_{t-6} + w_t,$$

where R_t is Recruitment, S_t is SOI, and D_t is a dummy variable that is 0 if $S_t < 0$ and 1 otherwise. However, residual analysis indicates that the residuals are not white noise. The sample (P)ACF of the residuals indicates that an AR(2) model might be appropriate, which is similar to the results of Example 3.44. We display partial results of the final model below.

```
dummy = ifelse(soi<0, 0, 1)
fish = ts.intersect(rec, soiL6=lag(soi,-6), dL6=lag(dummy,-6), dframe=TRUE)
summary(fit <- lm(rec ~soiL6*dL6, data=fish, na.action=NULL))
attach(fish)
plot(resid(fit))
acf2(resid(fit))      # indicates AR(2)
intract = soiL6*dL6   # interaction term
sarima(rec,2,0,0, xreg = cbind(soiL6, dL6, intract))
$ttable
      Estimate     SE  t.value p.value
ar1     1.3624 0.0440 30.9303 0.0000
ar2    -0.4703 0.0444 -10.5902 0.0000
intercept 64.8028 4.1121 15.7590 0.0000
soiL6     8.6671 2.2205  3.9033 0.0001
dL6     -2.5945 0.9535 -2.7209 0.0068
intract -10.3092 2.8311 -3.6415 0.0003
```

3.9 Multiplicative Seasonal ARIMA Models

In this section, we introduce several modifications made to the ARIMA model to account for seasonal and nonstationary behavior. Often, the dependence on the past tends to occur most strongly at multiples of some underlying seasonal lag s . For example, with monthly economic data, there is a strong yearly component occurring at lags that are multiples of $s = 12$, because of the strong connections of all activity to the calendar year. Data taken quarterly will exhibit the yearly repetitive period at $s = 4$ quarters. Natural phenomena such as temperature also have strong components corresponding to seasons. Hence, the natural variability of many physical, biological, and economic processes tends to match with seasonal fluctuations. Because of this, it is appropriate to introduce autoregressive and moving average polynomials that identify with the seasonal lags. The resulting *pure seasonal autoregressive moving average model*, say, $\text{ARMA}(P, Q)_s$, then takes the form

$$\Phi_P(B^s)x_t = \Theta_Q(B^s)w_t, \quad (3.158)$$

where the operators

$$\Phi_P(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \cdots - \Phi_P B^{Ps} \quad (3.159)$$

and

$$\Theta_Q(B^s) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \cdots + \Theta_Q B^{Qs} \quad (3.160)$$

are the **seasonal autoregressive operator** and the **seasonal moving average operator** of orders P and Q , respectively, with seasonal period s .

Analogous to the properties of nonseasonal ARMA models, the pure seasonal $\text{ARMA}(P, Q)_s$ is *causal* only when the roots of $\Phi_P(z^s)$ lie outside the unit circle, and it is *invertible* only when the roots of $\Theta_Q(z^s)$ lie outside the unit circle.

Example 3.46 A Seasonal AR Series

A first-order seasonal autoregressive series that might run over months could be written as

$$(1 - \Phi B^{12})x_t = w_t$$

or

$$x_t = \Phi x_{t-12} + w_t.$$

This model exhibits the series x_t in terms of past lags at the multiple of the yearly seasonal period $s = 12$ months. It is clear from the above form that estimation and forecasting for such a process involves only straightforward modifications of the unit lag case already treated. In particular, the causal condition requires $|\Phi| < 1$.

We simulated 3 years of data from the model with $\Phi = .9$, and exhibit the *theoretical* ACF and PACF of the model. See Figure 3.20.

```
set.seed(666)
phi = c(rep(0,11),.9)
sAR = arima.sim(list(order=c(12,0,0), ar=phi), n=37)
sAR = ts(sAR, freq=12)
layout(matrix(c(1,1,2, 1,1,3), nc=2))
```

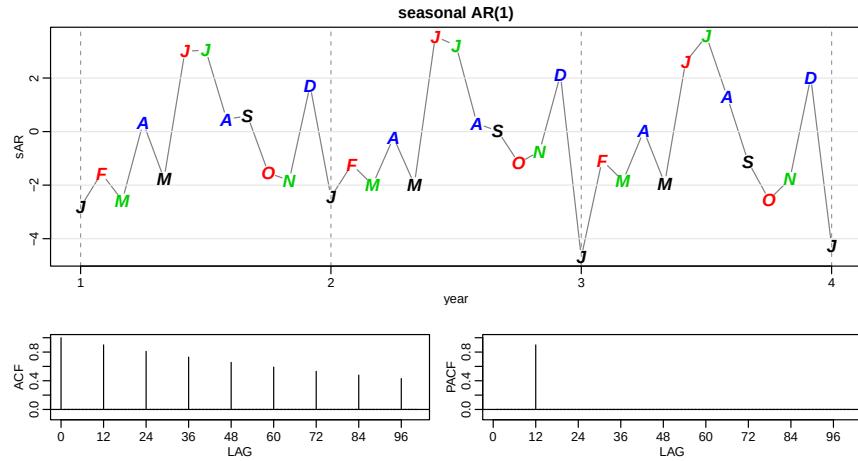


Fig. 3.20. Data generated from a seasonal ($s = 12$) AR(1), and the true ACF and PACF of the model $x_t = .9x_{t-12} + w_t$.

```
par(mar=c(3,3,2,1), mgp=c(1.6,.6,0))
plot(sAR, axes=FALSE, main='seasonal AR(1)', xlab="year", type='c')
Months = c("J","F","M","A","M","J","J","D","S","O","N","M")
points(sAR, pch=Months, cex=1.25, font=4, col=1:4)
axis(1, 1:4); abline(v=1:4, lty=2, col=gray(.7))
axis(2); box()
ACF = ARMAacf(ar=phi, ma=0, 100)
PACF = ARMAacf(ar=phi, ma=0, 100, pacf=TRUE)
plot(ACF,type="h", xlab="LAG", ylim=c(-.1,1)); abline(h=0)
plot(PACF, type="h", xlab="LAG", ylim=c(-.1,1)); abline(h=0)
```

For the first-order seasonal ($s = 12$) MA model, $x_t = w_t + \Theta w_{t-12}$, it is easy to verify that

$$\begin{aligned}\gamma(0) &= (1 + \Theta^2)\sigma^2 \\ \gamma(\pm 12) &= \Theta\sigma^2 \\ \gamma(h) &= 0, \quad \text{otherwise.}\end{aligned}$$

Thus, the only nonzero correlation, aside from lag zero, is

$$\rho(\pm 12) = \Theta/(1 + \Theta^2).$$

For the first-order seasonal ($s = 12$) AR model, using the techniques of the nonseasonal AR(1), we have

$$\begin{aligned}\gamma(0) &= \sigma^2/(1 - \Phi^2) \\ \gamma(\pm 12k) &= \sigma^2\Phi^k/(1 - \Phi^2) \quad k = 1, 2, \dots \\ \gamma(h) &= 0, \quad \text{otherwise.}\end{aligned}$$

In this case, the only non-zero correlations are

Table 3.3. Behavior of the ACF and PACF for Pure SARMA Models

	$\text{AR}(P)_s$	$\text{MA}(Q)_s$	$\text{ARMA}(P, Q)_s$
ACF*	Tails off at lags ks , $k = 1, 2, \dots$,	Cuts off after lag Qs	Tails off at lags ks
PACF*	Cuts off after lag Ps	Tails off at lags ks $k = 1, 2, \dots$,	Tails off at lags ks

*The values at nonseasonal lags $h \neq ks$, for $k = 1, 2, \dots$, are zero.

$$\rho(\pm 12k) = \Phi^k, \quad k = 0, 1, 2, \dots .$$

These results can be verified using the general result that $\gamma(h) = \Phi\gamma(h - 12)$, for $h \geq 1$. For example, when $h = 1$, $\gamma(1) = \Phi\gamma(11)$, but when $h = 11$, we have $\gamma(11) = \Phi\gamma(1)$, which implies that $\gamma(1) = \gamma(11) = 0$. In addition to these results, the PACF have the analogous extensions from nonseasonal to seasonal models. These results are demonstrated in [Figure 3.20](#).

As an initial diagnostic criterion, we can use the properties for the pure seasonal autoregressive and moving average series listed in [Table 3.3](#). These properties may be considered as generalizations of the properties for nonseasonal models that were presented in [Table 3.1](#).

In general, we can combine the seasonal and nonseasonal operators into a *multiplicative seasonal autoregressive moving average model*, denoted by $\text{ARMA}(p, q) \times (P, Q)_s$, and write

$$\Phi_P(B^s)\phi(B)x_t = \Theta_Q(B^s)\theta(B)w_t \quad (3.161)$$

as the overall model. Although the diagnostic properties in [Table 3.3](#) are not strictly true for the overall mixed model, the behavior of the ACF and PACF tends to show rough patterns of the indicated form. In fact, for mixed models, we tend to see a mixture of the facts listed in [Table 3.1](#) and [Table 3.3](#). In fitting such models, focusing on the seasonal autoregressive and moving average components first generally leads to more satisfactory results.

Example 3.47 A Mixed Seasonal Model

Consider an $\text{ARMA}(0, 1) \times (1, 0)_{12}$ model

$$x_t = \Phi x_{t-12} + w_t + \theta w_{t-1},$$

where $|\Phi| < 1$ and $|\theta| < 1$. Then, because x_{t-12} , w_t , and w_{t-1} are uncorrelated, and x_t is stationary, $\gamma(0) = \Phi^2\gamma(0) + \sigma_w^2 + \theta^2\sigma_w^2$, or

$$\gamma(0) = \frac{1 + \theta^2}{1 - \Phi^2} \sigma_w^2.$$

In addition, multiplying the model by x_{t-h} , $h > 0$, and taking expectations, we have $\gamma(1) = \Phi\gamma(11) + \theta\sigma_w^2$, and $\gamma(h) = \Phi\gamma(h - 12)$, for $h \geq 2$. Thus, the ACF for this model is

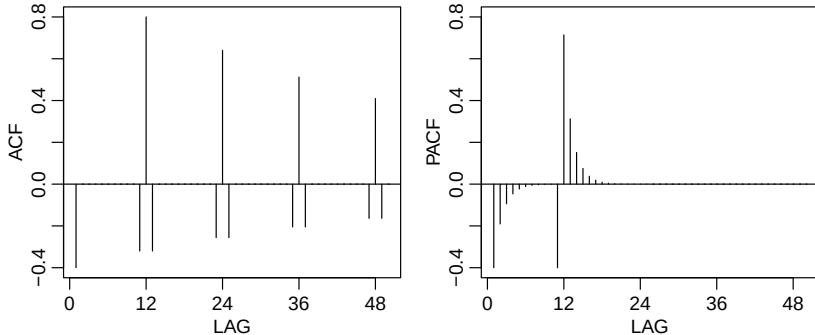


Fig. 3.21. ACF and PACF of the mixed seasonal ARMA model $x_t = .8x_{t-12} + w_t - .5w_{t-1}$.

$$\begin{aligned}\rho(12h) &= \Phi^h \quad h = 1, 2, \dots \\ \rho(12h-1) &= \rho(12h+1) = \frac{\theta}{1+\theta^2} \Phi^h \quad h = 0, 1, 2, \dots, \\ \rho(h) &= 0, \quad \text{otherwise.}\end{aligned}$$

The ACF and PACF for this model, with $\Phi = .8$ and $\theta = -.5$, are shown in Figure 3.21. These type of correlation relationships, although idealized here, are typically seen with seasonal data.

To reproduce Figure 3.21 in R, use the following commands:

```
phi = c(rep(0,11), .8)
ACF = ARMAacf(ar=phi, ma=-.5, 50)[-1]      # [-1] removes 0 lag
PACF = ARMAacf(ar=phi, ma=-.5, 50, pacf=TRUE)
par(mfrow=c(1,2))
plot(ACF, type="h", xlab="LAG", ylim=c(-.4,.8)); abline(h=0)
plot(PACF, type="h", xlab="LAG", ylim=c(-.4,.8)); abline(h=0)
```

Seasonal persistence occurs when the process is nearly periodic in the season. For example, with average monthly temperatures over the years, each January would be approximately the same, each February would be approximately the same, and so on. In this case, we might think of average monthly temperature x_t as being modeled as

$$x_t = S_t + w_t,$$

where S_t is a seasonal component that varies a little from one year to the next, according to a random walk,

$$S_t = S_{t-12} + v_t.$$

In this model, w_t and v_t are uncorrelated white noise processes. The tendency of data to follow this type of model will be exhibited in a sample ACF that is large and decays very slowly at lags $h = 12k$, for $k = 1, 2, \dots$. If we subtract the effect of successive years from each other, we find that

$$(1 - B^{12})x_t = x_t - x_{t-12} = v_t + w_t - w_{t-12}.$$

This model is a stationary MA(1)₁₂, and its ACF will have a peak only at lag 12. In general, seasonal differencing can be indicated when the ACF decays slowly at multiples of some season s , but is negligible between the periods. Then, a *seasonal difference of order D* is defined as

$$\nabla_s^D x_t = (1 - B^s)^D x_t, \quad (3.162)$$

where $D = 1, 2, \dots$, takes positive integer values. Typically, $D = 1$ is sufficient to obtain seasonal stationarity. Incorporating these ideas into a general model leads to the following definition.

Definition 3.12 *The multiplicative seasonal autoregressive integrated moving average model, or SARIMA model is given by*

$$\Phi_P(B^s)\phi(B)\nabla_s^D \nabla^d x_t = \delta + \Theta_Q(B^s)\theta(B)w_t, \quad (3.163)$$

where w_t is the usual Gaussian white noise process. The general model is denoted as **ARIMA**(p, d, q) \times (P, D, Q) _{s} . The ordinary autoregressive and moving average components are represented by polynomials $\phi(B)$ and $\theta(B)$ of orders p and q , respectively, and the seasonal autoregressive and moving average components by $\Phi_P(B^s)$ and $\Theta_Q(B^s)$ of orders P and Q and ordinary and seasonal difference components by $\nabla^d = (1 - B)^d$ and $\nabla_s^D = (1 - B^s)^D$.

Example 3.48 An SARIMA Model

Consider the following model, which often provides a reasonable representation for seasonal, nonstationary, economic time series. We exhibit the equations for the model, denoted by ARIMA(0, 1, 1) \times (0, 1, 1)₁₂ in the notation given above, where the seasonal fluctuations occur every 12 months. Then, with $\delta = 0$, the model (3.163) becomes

$$\nabla_{12}\nabla x_t = \Theta(B^{12})\theta(B)w_t$$

or

$$(1 - B^{12})(1 - B)x_t = (1 + \Theta B^{12})(1 + \theta B)w_t. \quad (3.164)$$

Expanding both sides of (3.164) leads to the representation

$$(1 - B - B^{12} + B^{13})x_t = (1 + \theta B + \Theta B^{12} + \Theta\theta B^{13})w_t,$$

or in difference equation form

$$x_t = x_{t-1} + x_{t-12} - x_{t-13} + w_t + \theta w_{t-1} + \Theta w_{t-12} + \Theta\theta w_{t-13}.$$

Note that the multiplicative nature of the model implies that the coefficient of w_{t-13} is the product of the coefficients of w_{t-1} and w_{t-12} rather than a free parameter. The multiplicative model assumption seems to work well with many seasonal time series data sets while reducing the number of parameters that must be estimated.

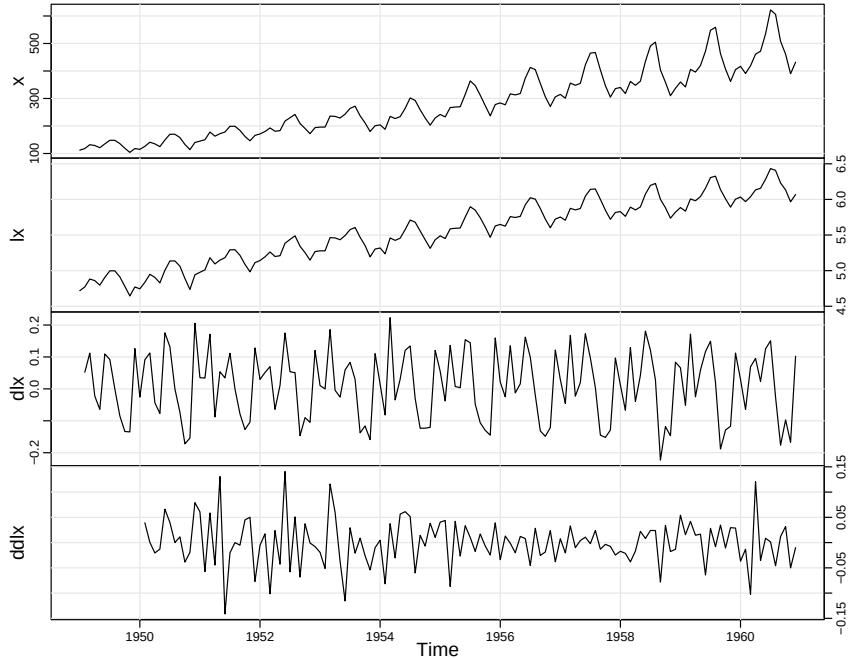


Fig. 3.22. R data set `AirPassengers`, which are the monthly totals of international airline passengers x , and the transformed data: $lx = \log x_t$, $dlx = \nabla \log x_t$, and $ddlx = \nabla_{12} \nabla \log x_t$.

Selecting the appropriate model for a given set of data from all of those represented by the general form (3.163) is a daunting task, and we usually think first in terms of finding difference operators that produce a roughly stationary series and then in terms of finding a set of simple autoregressive moving average or multiplicative seasonal ARMA to fit the resulting residual series. Differencing operations are applied first, and then the residuals are constructed from a series of reduced length. Next, the ACF and the PACF of these residuals are evaluated. Peaks that appear in these functions can often be eliminated by fitting an autoregressive or moving average component in accordance with the general properties of Table 3.1 and Table 3.3. In considering whether the model is satisfactory, the diagnostic techniques discussed in Section 3.7 still apply.

Example 3.49 Air Passengers

We consider the R data set `AirPassengers`, which are the monthly totals of international airline passengers, 1949 to 1960, taken from Box & Jenkins (1970). Various plots of the data and transformed data are shown in Figure 3.22 and were obtained as follows:

```
x = AirPassengers
lx = log(x); dlx = diff(lx); ddx = diff(dlx, 12)
```

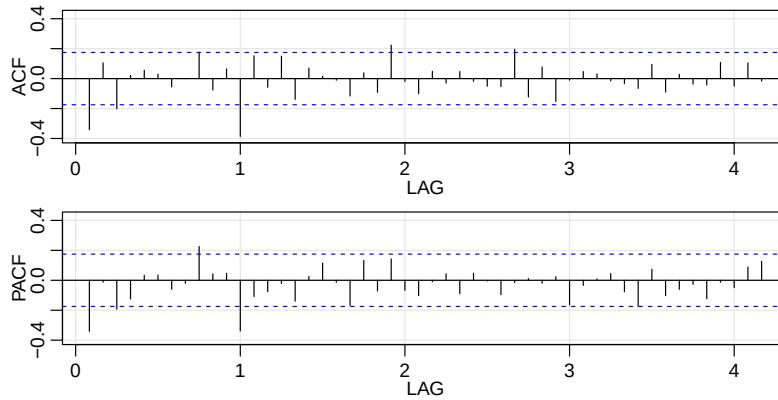


Fig. 3.23. Sample ACF and PACF of dd1x ($\nabla_{12} \nabla \log x_t$).

```
plot.ts(cbind(x, lx, dlx, dd1x), main="")
# below of interest for showing seasonal RW (not shown here):
par(mfrow=c(2,1))
monthplot(dlx); monthplot(dd1x)
```

Note that x is the original series, which shows trend plus increasing variance. The logged data are in lx , and the transformation stabilizes the variance. The logged data are then differenced to remove trend, and are stored in dlx . It is clear there is still persistence in the seasons (i.e., $dlx_t \approx dlx_{t-12}$), so that a twelfth-order difference is applied and stored in $dd1x$. The transformed data appears to be stationary and we are now ready to fit a model.

The sample ACF and PACF of dd1x ($\nabla_{12} \nabla \log x_t$) are shown in Figure 3.23. The R code is:

```
acf2(dd1x, 50)
```

Seasonal Component: It appears that at the seasons, the ACF is cutting off a lag $1s$ ($s = 12$), whereas the PACF is tailing off at lags $1s, 2s, 3s, 4s, \dots$. These results imply an SMA(1), $P = 0$, $Q = 1$, in the season ($s = 12$).

Non-Seasonal Component: Inspecting the sample ACF and PACF at the lower lags, it appears as though both are tailing off. This suggests an ARMA(1, 1) within the seasons, $p = q = 1$.

Thus, we first try an ARIMA(1, 1, 1) \times (0, 1, 1)₁₂ on the logged data:

```
sarima(lx, 1, 1, 1, 0, 1, 1, 12)
Coefficients:
      ar1      ma1      sma1
      0.1960 -0.5784 -0.5643
      s.e.  0.2475  0.2132  0.0747
sigma^2 estimated as 0.001341
$AIC -5.5726 $AICC -5.556713 $BIC -6.510729
```

However, the AR parameter is not significant, so we should try dropping one parameter from the within seasons part. In this case, we try both an ARIMA(0, 1, 1) \times (0, 1, 1)₁₂ and an ARIMA(1, 1, 0) \times (0, 1, 1)₁₂ model:

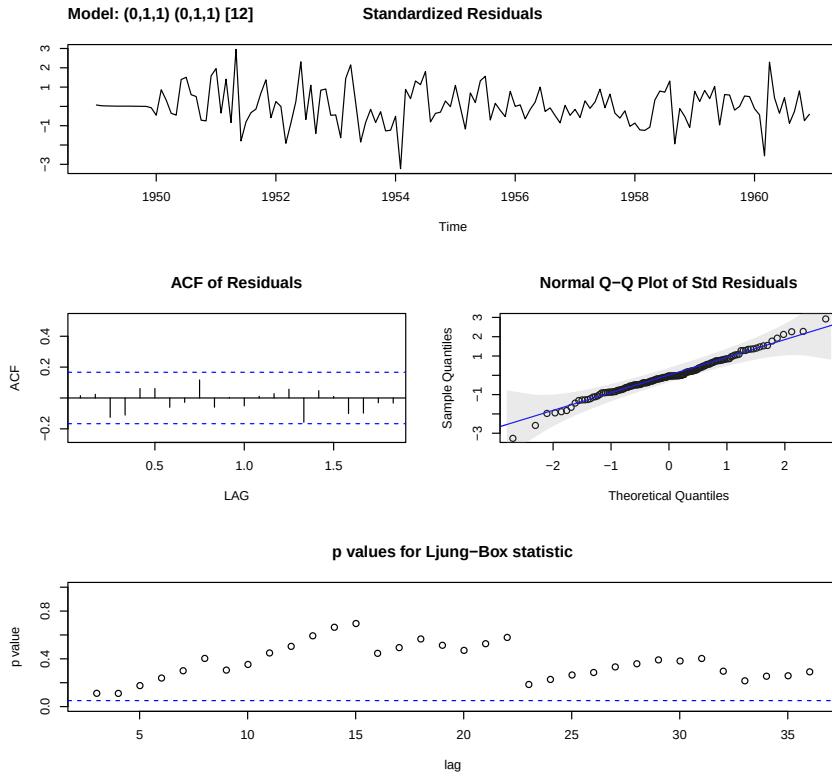


Fig. 3.24. Residual analysis for the $\text{ARIMA}(0, 1, 1) \times (0, 1, 1)_{12}$ fit to the logged air passengers data set.

```

sarima(lx, 0,1,1, 0,1,1,12)
Coefficients:
      ma1      sma1
     -0.4018   -0.5569
s.e.  0.0896   0.0731
sigma^2 estimated as 0.001348
$AIC -5.58133 $AICc -5.56625 $BIC -6.540082
sarima(lx, 1,1,0, 0,1,1,12)
Coefficients:
      ar1      sma1
     -0.3395   -0.5619
s.e.  0.0822   0.0748
sigma^2 estimated as 0.001367
$AIC -5.567081 $AICc -5.552002 $BIC -6.525834

```

All information criteria prefer the $\text{ARIMA}(0, 1, 1) \times (0, 1, 1)_{12}$ model, which is the model displayed in (3.164). The residual diagnostics are shown in Figure 3.24, and except for one or two outliers, the model seems to fit well.

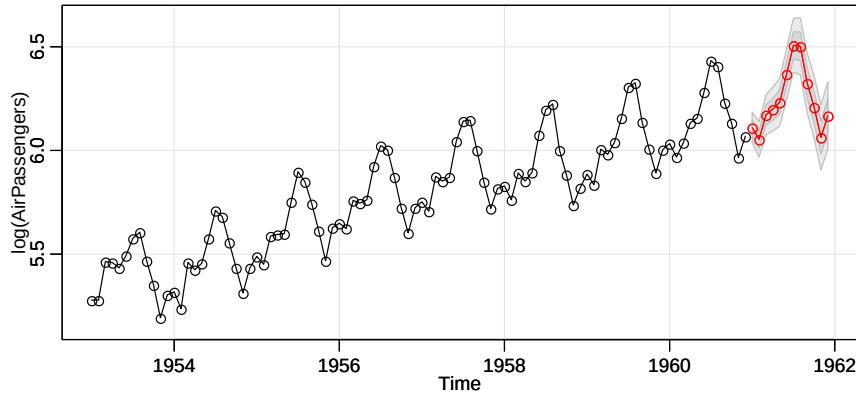


Fig. 3.25. Twelve month forecast using the ARIMA(0, 1, 1) \times (0, 1, 1)₁₂ model on the logged air passenger data set.

Finally, we forecast the logged data out twelve months, and the results are shown in Figure 3.25.

```
sarima.fore(1x, 12, 0, 1, 1, 0, 1, 1, 12)
```

Problems

Section 3.1

3.1 For an MA(1), $x_t = w_t + \theta w_{t-1}$, show that $|\rho_x(1)| \leq 1/2$ for any number θ . For which values of θ does $\rho_x(1)$ attain its maximum and minimum?

3.2 Let $\{w_t; t = 0, 1, \dots\}$ be a white noise process with variance σ_w^2 and let $|\phi| < 1$ be a constant. Consider the process $x_0 = w_0$, and

$$x_t = \phi x_{t-1} + w_t, \quad t = 1, 2, \dots.$$

We might use this method to simulate an AR(1) process from simulated white noise.

- (a) Show that $x_t = \sum_{j=0}^t \phi^j w_{t-j}$ for any $t = 0, 1, \dots$
- (b) Find the $E(x_t)$.
- (c) Show that, for $t = 0, 1, \dots$,

$$\text{var}(x_t) = \frac{\sigma_w^2}{1 - \phi^2} (1 - \phi^{2(t+1)})$$

- (d) Show that, for $h \geq 0$,

$$\text{cov}(x_{t+h}, x_t) = \phi^h \text{var}(x_t)$$

- (e) Is x_t stationary?

Chapter 6

State Space Models

A very general model that subsumes a whole class of special cases of interest in much the same way that linear regression does is the state-space model or the dynamic linear model, which was introduced in Kalman (1960) and Kalman and Bucy (1961). The model arose in the space tracking setting, where the state equation defines the motion equations for the position or state of a spacecraft with location x_t and the data y_t reflect information that can be observed from a tracking device such as velocity and azimuth. Although introduced as a method primarily for use in aerospace-related research, the model has been applied to modeling data from economics (Harrison and Stevens, 1976; Harvey and Pierse, 1984; Harvey and Todd, 1983; Kitagawa and Gersch 1984, Shumway and Stoffer, 1982), medicine (Jones, 1984) and the soil sciences (Shumway, 1988, §3.4.5). An excellent treatment of time series analysis based on the state space model is the text by Durbin and Koopman (2001). A modern treatment of nonlinear state space models can be found in Douc, Moulines and Stoffer (2014).

In this chapter, we focus primarily on linear Gaussian state space models. We present various forms of the model, introduce the concepts of prediction, filtering and smoothing state space models and include their derivations. We explain how to perform maximum likelihood estimation using various techniques, and include methods for handling missing data. In addition, we present several special topics such as hidden Markov models (HMM), switching autoregressions, smoothing splines, ARMAX models, bootstrapping, stochastic volatility, and state space models with switching. Finally, we discuss a Bayesian approach to fitting state space models using Markov chain Monte Carlo (MCMC) techniques. The essential material is supplied in Sections 6.1, 6.2, and 6.3. After that, the other sections may be read in any order with some occasional backtracking.

In general, the state space model is characterized by two principles. First, there is a hidden or latent process x_t called the state process. The state process is assumed to be a Markov process; this means that the future $\{x_s; s > t\}$, and past $\{x_s; s < t\}$, are independent conditional on the present, x_t . The second condition is that the observations, y_t are independent given the states x_t . This means that the dependence among the observations is generated by states. The principles are displayed in Figure 6.1.

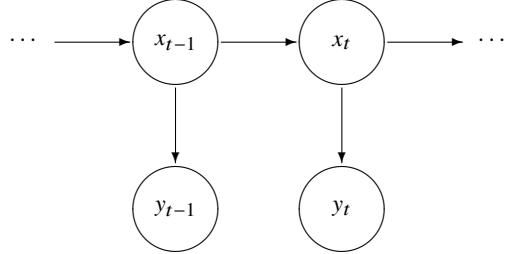


Fig. 6.1. Diagram of a state space model.

6.1 Linear Gaussian Model

The linear Gaussian state space model or dynamic linear model (DLM), in its basic form, employs an order one, p -dimensional vector autoregression as the *state equation*,

$$x_t = \Phi x_{t-1} + w_t . \quad (6.1)$$

The w_t are $p \times 1$ independent and identically distributed, zero-mean normal vectors with covariance matrix Q ; we write this as $w_t \sim \text{iid } N_p(0, Q)$. In the DLM, we assume the process starts with a normal vector x_0 , such that $x_0 \sim N_p(\mu_0, \Sigma_0)$.

We do not observe the state vector x_t directly, but only a linear transformed version of it with noise added, say

$$y_t = A_t x_t + v_t , \quad (6.2)$$

where A_t is a $q \times p$ *measurement or observation matrix*; (6.2) is called the *observation equation*. The observed data vector, y_t , is q -dimensional, which can be larger than or smaller than p , the state dimension. The additive observation noise is $v_t \sim \text{iid } N_q(0, R)$. In addition, we initially assume, for simplicity, x_0 , $\{w_t\}$ and $\{v_t\}$ are uncorrelated; this assumption is not necessary, but it helps in the explanation of first concepts. The case of correlated errors is discussed in Section 6.6.

As in the ARMAX model of Section 5.6, exogenous variables, or fixed inputs, may enter into the states or into the observations. In this case, we suppose we have an $r \times 1$ vector of inputs u_t , and write the model as

$$x_t = \Phi x_{t-1} + \Upsilon u_t + w_t \quad (6.3)$$

$$y_t = A_t x_t + \Gamma u_t + v_t \quad (6.4)$$

where Υ is $p \times r$ and Γ is $q \times r$; either of these matrices may be the zero matrix.

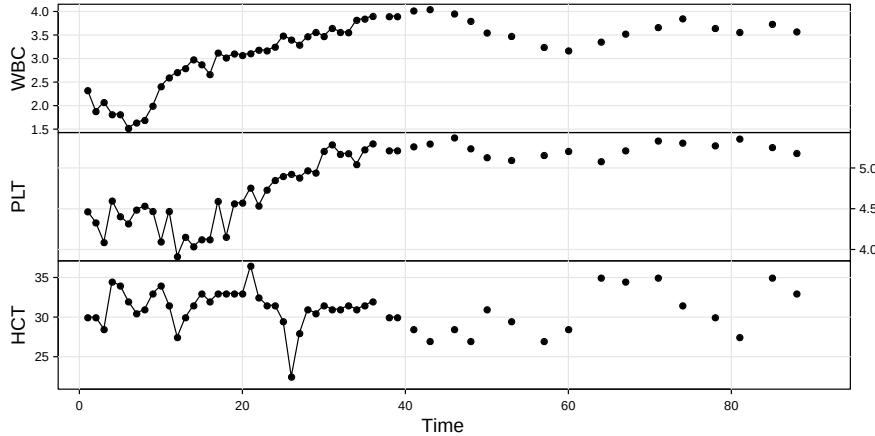


Fig. 6.2. Longitudinal series of monitored blood parameters, log (white blood count) [WBC], log(platelet) [PLT], and hematocrit [HCT], after a bone marrow transplant ($n = 91$ days).

Example 6.1 A Biomedical Example

Suppose we consider the problem of monitoring the level of several biomedical markers after a cancer patient undergoes a bone marrow transplant. The data in Figure 6.2, used by Jones (1984), are measurements made for 91 days on three variables, log(white blood count) [WBC], log(platelet) [PLT], and hematocrit [HCT], denoted $y_t = (y_{t1}, y_{t2}, y_{t3})'$. Approximately 40% of the values are missing, with missing values occurring primarily after the 35th day. The main objectives are to model the three variables using the state-space approach, and to estimate the missing values. According to Jones, “Platelet count at about 100 days post transplant has previously been shown to be a good indicator of subsequent long term survival.” For this particular situation, we model the three variables in terms of the state equation (6.1); that is,

$$\begin{pmatrix} x_{t1} \\ x_{t2} \\ x_{t3} \end{pmatrix} = \begin{pmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{pmatrix} \begin{pmatrix} x_{t-1,1} \\ x_{t-1,2} \\ x_{t-1,3} \end{pmatrix} + \begin{pmatrix} w_{t1} \\ w_{t2} \\ w_{t3} \end{pmatrix}. \quad (6.5)$$

The observation equations would be $y_t = A_t x_t + v_t$, where the 3×3 observation matrix, A_t , is either the identity matrix or the zero matrix depending on whether a blood sample was taken on that day. The covariance matrices R and Q are each 3×3 matrices. A plot similar to Figure 6.2 can be produced as follows.

```
plot(blood, type='o', pch=19, xlab='day', main='')
```

As we progress through the chapter, it will become apparent that, while the model seems simplistic, it is quite general. For example, if the state process is VAR(2), we may write the state equation as a $2p$ -dimensional process,

$$\begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix} = \begin{pmatrix} \Phi_1 & \Phi_2 \\ I & 0 \end{pmatrix} \begin{pmatrix} x_{t-1} \\ x_{t-2} \end{pmatrix} + \begin{pmatrix} w_t \\ 0 \end{pmatrix}, \quad (6.6)$$

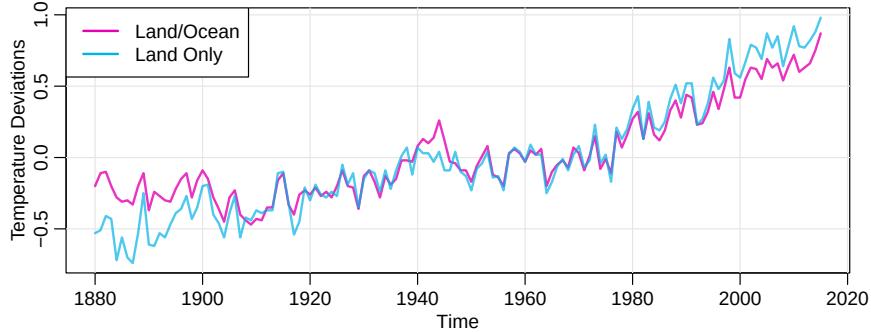


Fig. 6.3. Annual global temperature deviation series, measured in degrees centigrade, 1880–2015. The series differ by whether or not ocean data is included.

and the observation equation as the q -dimensional process,

$$\begin{matrix} y_t \\ q \times 1 \end{matrix} = \left[\begin{matrix} A_t & | & 0 \\ q \times 2p & & \end{matrix} \right] \begin{pmatrix} x_t \\ x_{t-1} \\ \vdots \\ x_{t-p+1} \end{pmatrix} + \begin{matrix} v_t \\ q \times 1 \end{matrix}. \quad (6.7)$$

The real advantages of the state space formulation, however, do not really come through in the simple example given above. The special forms that can be developed for various versions of the matrix A_t and for the transition scheme defined by the matrix Φ allow fitting more parsimonious structures with fewer parameters needed to describe a multivariate time series. We will see numerous examples throughout the chapter; [Section 6.5](#) on *structural models* is a good example of the model flexibility. The simple example shown below is instructive.

Example 6.2 Global Warming

Figure 6.3 shows two different estimators for the global temperature series from 1880 to 2015. One is `globtemp`, which was considered in the first chapter, and are the global mean land-ocean temperature index data. The second series, `globtempl`, are the surface air temperature index data using only meteorological station data. Conceptually, both series should be measuring the same underlying climatic signal, and we may consider the problem of extracting this underlying signal. The R code to generate the figure is

```
ts.plot(globtemp, globtempl, col=c(6,4), ylab='Temperature Deviations')
```

We suppose both series are observing the same signal with different noises; that is,

$$y_{t1} = x_t + v_{t1} \quad \text{and} \quad y_{t2} = x_t + v_{t2},$$

or more compactly as

$$\begin{pmatrix} y_{t1} \\ y_{t2} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} x_t + \begin{pmatrix} v_{t1} \\ v_{t2} \end{pmatrix}, \quad (6.8)$$

where

$$R = \text{var} \begin{pmatrix} v_{t1} \\ v_{t2} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}.$$

It is reasonable to suppose that the unknown common signal, x_t , can be modeled as a random walk with drift of the form

$$x_t = \delta + x_{t-1} + w_t, \quad (6.9)$$

with $Q = \text{var}(w_t)$. In terms of the model (6.3)–(6.4), this example has, $p = 1$, $q = 2$, $\Phi = 1$, and $\Upsilon = \delta$ with $u_t \equiv 1$.

The introduction of the state-space approach as a tool for modeling data in the social and biological sciences requires model identification and parameter estimation because there is rarely a well-defined differential equation describing the state transition. The questions of general interest for the dynamic linear model (6.3) and (6.4) relate to estimating the unknown parameters contained in Φ , Υ , Q , Γ , A_t , and R , that define the particular model, and estimating or forecasting values of the underlying unobserved process x_t . The advantages of the state-space formulation are in the ease with which we can treat various missing data configurations and in the incredible array of models that can be generated from (6.3) and (6.4). The analogy between the observation matrix A_t and the design matrix in the usual regression and analysis of variance setting is a useful one. We can generate fixed and random effect structures that are either constant or vary over time simply by making appropriate choices for the matrix A_t and the transition structure Φ .

Before continuing our investigation of the general model, it is instructive to consider a simple univariate state-space model wherein an AR(1) process is observed using a noisy instrument.

Example 6.3 An AR(1) Process with Observational Noise

Consider a univariate state-space model where the observations are noisy,

$$y_t = x_t + v_t, \quad (6.10)$$

and the signal (state) is an AR(1) process,

$$x_t = \phi x_{t-1} + w_t, \quad (6.11)$$

where $v_t \sim \text{iid } N(0, \sigma_v^2)$, $w_t \sim \text{iid } N(0, \sigma_w^2)$, and $x_0 \sim N(0, \frac{\sigma_w^2}{1-\phi^2})$; $\{v_t\}$, $\{w_t\}$, and x_0 are independent, and $t = 1, 2, \dots$

In Chapter 3, we investigated the properties of the state, x_t , because it is a stationary AR(1) process (recall Problem 3.2). For example, we know the autocovariance function of x_t is

$$\gamma_x(h) = \frac{\sigma_w^2}{1-\phi^2} \phi^h, \quad h = 0, 1, 2, \dots \quad (6.12)$$

But here, we must investigate how the addition of observation noise affects the dynamics. Although it is not a necessary assumption, we have assumed in this

example that x_t is stationary. In this case, the observations are also stationary because y_t is the sum of two independent stationary components x_t and v_t . We have

$$\gamma_y(0) = \text{var}(y_t) = \text{var}(x_t + v_t) = \frac{\sigma_w^2}{1 - \phi^2} + \sigma_v^2, \quad (6.13)$$

and, when $h \geq 1$,

$$\gamma_y(h) = \text{cov}(y_t, y_{t-h}) = \text{cov}(x_t + v_t, x_{t-h} + v_{t-h}) = \gamma_x(h). \quad (6.14)$$

Consequently, for $h \geq 1$, the ACF of the observations is

$$\rho_y(h) = \frac{\gamma_y(h)}{\gamma_y(0)} = \left(1 + \frac{\sigma_v^2}{\sigma_w^2}(1 - \phi^2)\right)^{-1} \phi^h. \quad (6.15)$$

It should be clear from the correlation structure given by (6.15) that the observations, y_t , are not AR(1) unless $\sigma_v^2 = 0$. In addition, the autocorrelation structure of y_t is identical to the autocorrelation structure of an ARMA(1,1) process, as presented in [Example 3.14](#). Thus, the observations can also be written in an ARMA(1,1) form,

$$y_t = \phi y_{t-1} + \theta u_{t-1} + u_t,$$

where u_t is Gaussian white noise with variance σ_u^2 , and with θ and σ_u^2 suitably chosen. We leave the specifics of this problem alone for now and defer the discussion to [Section 6.6](#); in particular, see [Example 6.11](#).

Although an equivalence exists between stationary ARMA models and stationary state-space models (see [Section 6.6](#)), it is sometimes easier to work with one form than another. As previously mentioned, in the case of missing data, complex multivariate systems, mixed effects, and certain types of nonstationarity, it is easier to work in the framework of state-space models.

6.2 Filtering, Smoothing, and Forecasting

From a practical view, a primary aim of any analysis involving the state space model, (6.3)–(6.4), would be to produce estimators for the underlying unobserved signal x_t , given the data $y_{1:s} = \{y_1, \dots, y_s\}$, to time s . As will be seen, state estimation is an essential component of parameter estimation. When $s < t$, the problem is called *forecasting* or *prediction*. When $s = t$, the problem is called *filtering*, and when $s > t$, the problem is called *smoothing*. In addition to these estimates, we would also want to measure their precision. The solution to these problems is accomplished via the *Kalman filter and smoother* and is the focus of this section.

Throughout this chapter, we will use the following definitions:

$$x_t^s = E(x_t | y_{1:s}) \quad (6.16)$$

and

$$P_{t_1, t_2}^s = E \{ (x_{t_1} - x_{t_1}^s)(x_{t_2} - x_{t_2}^s)' \}. \quad (6.17)$$

When $t_1 = t_2$ ($= t$ say) in (6.17), we will write P_t^s for convenience.

In obtaining the filtering and smoothing equations, we will rely heavily on the Gaussian assumption. Some knowledge of the material covered in [Appendix B](#) will be helpful in understanding the details of this section (although these details may be skipped on a casual reading of the material). Even in the non-Gaussian case, the estimators we obtain are the minimum mean-squared error estimators within the class of linear estimators. That is, we can think of E in (6.16) as the projection operator in the sense of [Section B.1](#) rather than expectation and $y_{1:s}$ as the space of linear combinations of $\{y_1, \dots, y_s\}$; in this case, P_t^s is the corresponding mean-squared error. Since the processes are Gaussian, (6.17) is also the conditional error covariance; that is,

$$P_{t_1, t_2}^s = E \{ (x_{t_1} - x_{t_1}^s)(x_{t_2} - x_{t_2}^s)' \mid y_{1:s} \}.$$

This fact can be seen, for example, by noting the covariance matrix between $(x_t - x_t^s)$ and $y_{1:s}$, for any t and s , is zero; we could say they are orthogonal in the sense of [Section B.1](#). This result implies that $(x_t - x_t^s)$ and $y_{1:s}$ are independent (because of the normality), and hence, the conditional distribution of $(x_t - x_t^s)$ given $y_{1:s}$ is the unconditional distribution of $(x_t - x_t^s)$. Derivations of the filtering and smoothing equations from a Bayesian perspective are given in Meinhold and Singpurwalla (1983); more traditional approaches based on the concept of projection and on multivariate normal distribution theory are given in Jazwinski (1970) and Anderson and Moore (1979).

First, we present the Kalman filter, which gives the filtering and forecasting equations. The name filter comes from the fact that x_t^t is a linear filter of the observations $y_{1:t}$; that is, $x_t^t = \sum_{s=1}^t B_s y_s$ for suitably chosen $p \times q$ matrices B_s . The advantage of the Kalman filter is that it specifies how to update the filter from x_{t-1}^{t-1} to x_t^t once a new observation y_t is obtained, without having to reprocess the entire data set $y_{1:t}$.

Property 6.1 The Kalman Filter

For the state-space model specified in (6.3) and (6.4), with initial conditions $x_0^0 = \mu_0$ and $P_0^0 = \Sigma_0$, for $t = 1, \dots, n$,

$$x_t^{t-1} = \Phi x_{t-1}^{t-1} + \Gamma u_t, \quad (6.18)$$

$$P_t^{t-1} = \Phi P_{t-1}^{t-1} \Phi' + Q, \quad (6.19)$$

with

$$x_t^t = x_t^{t-1} + K_t (y_t - A_t x_t^{t-1} - \Gamma u_t), \quad (6.20)$$

$$P_t^t = [I - K_t A_t] P_t^{t-1}, \quad (6.21)$$

where

$$K_t = P_t^{t-1} A_t' [A_t P_t^{t-1} A_t' + R]^{-1} \quad (6.22)$$

is called the Kalman gain. Prediction for $t > n$ is accomplished via (6.18) and (6.19) with initial conditions x_n^n and P_n^n . Important byproducts of the filter are the innovations (prediction errors)

$$\epsilon_t = y_t - E(y_t \mid y_{1:t-1}) = y_t - A_t x_t^{t-1} - \Gamma u_t, \quad (6.23)$$

and the corresponding variance-covariance matrices

$$\Sigma_t \stackrel{\text{def}}{=} \text{var}(\epsilon_t) = \text{var}[A_t(x_t - x_t^{t-1}) + v_t] = A_t P_t^{t-1} A_t' + R \quad (6.24)$$

for $t = 1, \dots, n$. We assume that $\Sigma_t > 0$ (is positive definite), which is guaranteed, for example, if $R > 0$. This assumption is not necessary and may be relaxed.

Proof: The derivations of (6.18) and (6.19) follow from straight forward calculations, because from (6.3) we have

$$x_t^{t-1} = E(x_t \mid y_{1:t-1}) = E(\Phi x_{t-1} + \Upsilon u_t + w_t \mid y_{1:t-1}) = \Phi x_{t-1}^{t-1} + \Upsilon u_t,$$

and thus

$$\begin{aligned} P_t^{t-1} &= E\{(x_t - x_t^{t-1})(x_t - x_t^{t-1})'\} \\ &= E\left\{\left[\Phi(x_{t-1} - x_{t-1}^{t-1}) + w_t\right]\left[\Phi(x_{t-1} - x_{t-1}^{t-1}) + w_t\right]'\right\} \\ &= \Phi P_{t-1}^{t-1} \Phi' + Q. \end{aligned}$$

To derive (6.20), we note that $\text{cov}(\epsilon_t, y_s) = 0$ for $s < t$, which in view of the fact the innovation sequence is a Gaussian process, implies that the innovations are independent of the past observations. Furthermore, the conditional covariance between x_t and ϵ_t given $y_{1:t-1}$ is

$$\begin{aligned} \text{cov}(x_t, \epsilon_t \mid y_{1:t-1}) &= \text{cov}(x_t, y_t - A_t x_t^{t-1} - \Gamma u_t \mid y_{1:t-1}) \\ &= \text{cov}(x_t - x_t^{t-1}, y_t - A_t x_t^{t-1} - \Gamma u_t \mid y_{1:t-1}) \\ &= \text{cov}[x_t - x_t^{t-1}, A_t(x_t - x_t^{t-1}) + v_t] \\ &= P_t^{t-1} A_t'. \end{aligned} \quad (6.25)$$

Using these results we have that the joint conditional distribution of x_t and ϵ_t given $y_{1:t-1}$ is normal

$$\begin{pmatrix} x_t \\ \epsilon_t \end{pmatrix} \mid y_{1:t-1} \sim N\left(\begin{bmatrix} x_t^{t-1} \\ 0 \end{bmatrix}, \begin{bmatrix} P_t^{t-1} & P_t^{t-1} A_t' \\ A_t P_t^{t-1} & \Sigma_t \end{bmatrix}\right). \quad (6.26)$$

Thus, using (B.9) of Appendix B, we can write

$$x_t^t = E(x_t \mid y_{1:t}) = E(x_t \mid y_{1:t-1}, \epsilon_t) = x_t^{t-1} + K_t \epsilon_t, \quad (6.27)$$

where

$$K_t = P_t^{t-1} A_t' \Sigma_t^{-1} = P_t^{t-1} A_t' (A_t P_t^{t-1} A_t' + R)^{-1}.$$

The evaluation of P_t^t is easily computed from (6.26) [see (B.10)] as

$$P_t^t = \text{cov}(x_t \mid y_{1:t-1}, \epsilon_t) = P_t^{t-1} - P_t^{t-1} A_t' \Sigma_t^{-1} A_t P_t^{t-1},$$

which simplifies to (6.21). \square

Nothing in the proof of Property 6.1 precludes the cases where some or all of the parameters vary with time, or where the observation dimension changes with time, which leads to the following corollary.

Corollary 6.1 Kalman Filter: The Time-Varying Case

If, in (6.3) and (6.4), any or all of the parameters are time dependent, $\Phi = \Phi_t$, $\Upsilon = \Upsilon_t$, $Q = Q_t$ in the state equation or $\Gamma = \Gamma_t$, $R = R_t$ in the observation equation, or the dimension of the observational equation is time dependent, $q = q_t$, **Property 6.1** holds with the appropriate substitutions.

Next, we explore the model, prediction, and filtering from a density point of view. To ease the notation, we will drop the inputs from the model. There are two key ingredients to the state space model. Letting $p_{\Theta}(\cdot)$ denote a generic density function with parameters represented by Θ , we have the state process is Markovian:

$$p_{\Theta}(x_t \mid x_{t-1}, x_{t-2}, \dots, x_0) = p_{\Theta}(x_t \mid x_{t-1}), \quad (6.28)$$

and the observations are conditionally independent given the states:

$$p_{\Theta}(y_{1:n} \mid x_{1:n}) = \prod_{t=1}^n p_{\Theta}(y_t \mid x_t), \quad (6.29)$$

Since we are focusing on the linear Gaussian model, if we let $g(x; \mu, \Sigma)$ denote a multivariate normal density with mean μ and covariance matrix Σ as given in (1.33), then

$$p_{\Theta}(x_t \mid x_{t-1}) = g(x_t; \Phi x_{t-1}, Q) \quad \text{and} \quad p_{\Theta}(y_t \mid x_t) = g(y_t; A_t x_t, R).$$

with initial condition $p_{\Theta}(x_0) = g(x_0; \mu_0, \Sigma_0)$.

In terms of densities, the Kalman filter can be seen as a simple updating scheme, where, to determine the forecast densities, we have,

$$\begin{aligned} p_{\Theta}(x_t \mid y_{1:t-1}) &= \int_{\mathbb{R}^p} p_{\Theta}(x_t, x_{t-1} \mid y_{1:t-1}) dx_{t-1} \\ &= \int_{\mathbb{R}^p} p_{\Theta}(x_t \mid x_{t-1}) p_{\Theta}(x_{t-1} \mid y_{1:t-1}) dx_{t-1} \\ &= \int_{\mathbb{R}^p} g(x_t; \Phi x_{t-1}, Q) g(x_{t-1}; x_{t-1}^{t-1}, P_{t-1}^{t-1}) dx_{t-1} \\ &= g(x_t; x_t^{t-1}, P_t^{t-1}), \end{aligned} \quad (6.30)$$

where the values of x_t^{t-1} and P_t^{t-1} are given in (6.18) and (6.19). These values are obtained upon evaluating the integral using the usual trick of completing the square; see **Example 6.4**. Since we were seeking an iterative procedure, we introduced x_{t-1} in (6.30) because we have (presumably) previously evaluated the filter density $p_{\Theta}(x_{t-1} \mid y_{1:t-1})$. Once we have the predictor, the filter density is obtained as

$$\begin{aligned} p_{\Theta}(x_t \mid y_{1:t}) &= p_{\Theta}(x_t \mid y_t, y_{1:t-1}) \propto p_{\Theta}(y_t \mid x_t) p_{\Theta}(x_t \mid y_{1:t-1}), \\ &= g(y_t; A_t x_t, R) g(x_t; x_t^{t-1}, P_t^{t-1}), \end{aligned} \quad (6.31)$$

from which we deduce is $g(x_t; x_t^t, P_t^t)$ where x_t^t and P_t^t are given in (6.20) and (6.21). The following example illustrates these ideas for a simple univariate case.

Example 6.4 Local Level Model

In this example, we suppose that we observe a univariate series y_t that consists of a trend component, μ_t , and a noise component, v_t , where

$$y_t = \mu_t + v_t \quad (6.32)$$

and $v_t \sim \text{iid } N(0, \sigma_v^2)$. In particular, we assume the trend is a random walk given by

$$\mu_t = \mu_{t-1} + w_t \quad (6.33)$$

where $w_t \sim \text{iid } N(0, \sigma_w^2)$ is independent of $\{v_t\}$. Recall [Example 6.2](#), where we suggested this type of trend model for the global temperature series.

The model is, of course, a state-space model with [\(6.32\)](#) being the observation equation, and [\(6.33\)](#) being the state equation. We will use the following notation introduced in Blight (1974). Let

$$\{x; \mu, \sigma^2\} = \exp \left\{ -\frac{1}{2\sigma^2}(x - \mu)^2 \right\}, \quad (6.34)$$

then simple manipulation shows

$$\{x; \mu, \sigma^2\} = \{\mu; x, \sigma^2\} \quad (6.35)$$

and by completing the square,

$$\begin{aligned} \{x; \mu_1, \sigma_1^2\} \{x; \mu_2, \sigma_2^2\} &= \left\{ x; \frac{\mu_1/\sigma_1^2 + \mu_2/\sigma_2^2}{1/\sigma_1^2 + 1/\sigma_2^2}, (1/\sigma_1^2 + 1/\sigma_2^2)^{-1} \right\} \\ &\times \{\mu_1; \mu_2, \sigma_1^2 + \sigma_2^2\}. \end{aligned} \quad (6.36)$$

Thus, using [\(6.30\)](#), [\(6.35\)](#) and [\(6.36\)](#) we have

$$\begin{aligned} p(\mu_t | y_{1:t-1}) &\propto \int \{\mu_t; \mu_{t-1}, \sigma_w^2\} \{\mu_{t-1}; \mu_{t-1}^{t-1}, P_{t-1}^{t-1}\} d\mu_{t-1} \\ &= \int \{\mu_{t-1}; \mu_t, \sigma_w^2\} \{\mu_{t-1}; \mu_{t-1}^{t-1}, P_{t-1}^{t-1}\} d\mu_{t-1} \\ &= \{\mu_t; \mu_{t-1}^{t-1}, P_{t-1}^{t-1} + \sigma_w^2\}. \end{aligned} \quad (6.37)$$

From [\(6.37\)](#) we conclude that

$$\mu_t | y_{1:t-1} \sim N(\mu_t^{t-1}, P_t^{t-1}) \quad (6.38)$$

where

$$\mu_t^{t-1} = \mu_{t-1}^{t-1} \quad \text{and} \quad P_t^{t-1} = P_{t-1}^{t-1} + \sigma_w^2 \quad (6.39)$$

which agrees with the first part of [Property 6.1](#). To derive the filter density using [\(6.31\)](#) and [\(6.35\)](#) we have

$$\begin{aligned} p(\mu_t \mid y_{1:t}) &\propto \{y_t; \mu_t, \sigma_v^2\} \{\mu_t; \mu_t^{t-1}, P_t^{t-1}\} \\ &= \{\mu_t; y_t, \sigma_v^2\} \{\mu_t; \mu_t^{t-1}, P_t^{t-1}\}. \end{aligned} \quad (6.40)$$

An application of (6.36) gives

$$\mu_t \mid y_{1:t} \sim N(\mu_t^t, P_t^t) \quad (6.41)$$

with

$$\mu_t^t = \frac{\sigma_v^2 \mu_t^{t-1}}{P_t^{t-1} + \sigma_v^2} + \frac{P_t^{t-1} y_t}{P_t^{t-1} + \sigma_v^2} = \mu_t^{t-1} + K_t(y_t - \mu_t^{t-1}), \quad (6.42)$$

where we have defined

$$K_t = \frac{P_t^{t-1}}{P_t^{t-1} + \sigma_v^2}, \quad (6.43)$$

and

$$P_t^t = \left(\frac{1}{\sigma_v^2} + \frac{1}{P_t^{t-1}} \right)^{-1} = \frac{\sigma_v^2 P_t^{t-1}}{P_t^{t-1} + \sigma_v^2} = (1 - K_t) P_t^{t-1}. \quad (6.44)$$

The filter for this specific case, of course, agrees with [Property 6.1](#).

Next, we consider the problem of obtaining estimators for x_t based on the entire data sample y_1, \dots, y_n , where $t \leq n$, namely, x_t^n . These estimators are called smoothers because a time plot of the sequence $\{x_t^n; t = 1, \dots, n\}$ is typically smoother than the forecasts $\{x_t^{t-1}; t = 1, \dots, n\}$ or the filters $\{x_t^t; t = 1, \dots, n\}$. As is obvious from the above remarks, smoothing implies that each estimated value is a function of the present, future, and past, whereas the filtered estimator depends on the present and past. The forecast depends only on the past, as usual.

Property 6.2 The Kalman Smoother

For the state-space model specified in (6.3) and (6.4), with initial conditions x_n^n and P_n^n obtained via [Property 6.1](#), for $t = n, n-1, \dots, 1$,

$$x_{t-1}^n = x_{t-1}^{t-1} + J_{t-1} \left(x_t^n - x_t^{t-1} \right), \quad (6.45)$$

$$P_{t-1}^n = P_{t-1}^{t-1} + J_{t-1} \left(P_t^n - P_t^{t-1} \right) J_{t-1}', \quad (6.46)$$

where

$$J_{t-1} = P_{t-1}^{t-1} \Phi' \left[P_t^{t-1} \right]^{-1}. \quad (6.47)$$

Proof: The smoother can be derived in many ways. Here we provide a proof that was given in Ansley and Kohn (1982). First, for $1 \leq t \leq n$, define

$$y_{1:t-1} = \{y_1, \dots, y_{t-1}\} \quad \text{and} \quad \eta_t = \{v_t, \dots, v_n, w_{t+1}, \dots, w_n\},$$

with $y_{1:0}$ being empty, and let

$$m_{t-1} = E\{x_{t-1} \mid y_{1:t-1}, x_t - x_t^{t-1}, \eta_t\}.$$

Then, because $y_{1:t-1}$, $\{x_t - x_t^{t-1}\}$, and η_t are mutually independent, and x_{t-1} and η_t are independent, using (B.9) we have

$$m_{t-1} = x_{t-1}^{t-1} + J_{t-1}(x_t - x_t^{t-1}), \quad (6.48)$$

where

$$J_{t-1} = \text{cov}(x_{t-1}, x_t - x_t^{t-1})[P_t^{t-1}]^{-1} = P_{t-1}^{t-1} \Phi' [P_t^{t-1}]^{-1}.$$

Finally, because $y_{1:t-1}$, $x_t - x_t^{t-1}$, and η_t generate $y_{1:n} = \{y_1, \dots, y_n\}$,

$$x_{t-1}^n = E\{x_{t-1} \mid y_{1:n}\} = E\{m_{t-1} \mid y_{1:n}\} = x_{t-1}^{t-1} + J_{t-1}(x_t^n - x_t^{t-1}),$$

which establishes (6.45).

The recursion for the error covariance, P_{t-1}^n , is obtained by straight-forward calculation. Using (6.45) we obtain

$$x_{t-1} - x_{t-1}^n = x_{t-1} - x_{t-1}^{t-1} - J_{t-1} \left(x_t^n - \Phi x_{t-1}^{t-1} \right),$$

or

$$(x_{t-1} - x_{t-1}^n) + J_{t-1} x_t^n = \left(x_{t-1} - x_{t-1}^{t-1} \right) + J_{t-1} \Phi x_{t-1}^{t-1}. \quad (6.49)$$

Multiplying each side of (6.49) by the transpose of itself and taking expectation, we have

$$P_{t-1}^n + J_{t-1} E(x_t^n x_t^{n'}) J_{t-1}' = P_{t-1}^{t-1} + J_{t-1} \Phi E(x_{t-1}^{t-1} x_{t-1}^{t-1'}) \Phi' J_{t-1}', \quad (6.50)$$

using the fact the cross-product terms are zero. But,

$$E(x_t^n x_t^{n'}) = E(x_t x_t') - P_t^n = \Phi E(x_{t-1} x_{t-1}') \Phi' + Q - P_t^n,$$

and

$$E(x_{t-1}^{t-1} x_{t-1}^{t-1'}) = E(x_{t-1} x_{t-1}') - P_{t-1}^{t-1},$$

so (6.50) simplifies to (6.46). \square

Example 6.5 Prediction, Filtering and Smoothing for the Local Level Model

For this example, we simulated $n = 50$ observations from the local level trend model discussed in Example 6.4. We generated a random walk

$$\mu_t = \mu_{t-1} + w_t \quad (6.51)$$

with $w_t \sim \text{iid } N(0, 1)$ and $\mu_0 \sim N(0, 1)$. We then supposed that we observe a univariate series y_t consisting of the trend component, μ_t , and a noise component, $v_t \sim \text{iid } N(0, 1)$, where

$$y_t = \mu_t + v_t. \quad (6.52)$$

The sequences $\{w_t\}$, $\{v_t\}$ and μ_0 were generated independently. We then ran the Kalman filter and smoother, Property 6.1 and Property 6.2, using the actual parameters. The top panel of Figure 6.4 shows the actual values of μ_t as points, and

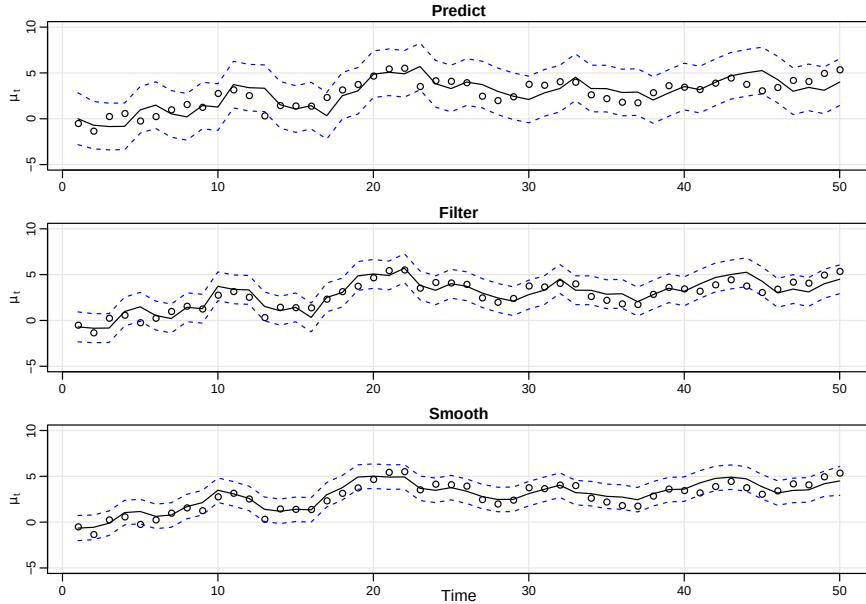


Fig. 6.4. Displays for Example 6.5. The simulated values of μ_t , for $t = 1, \dots, 50$, given by (6.51) are shown as points. The top shows the predictions μ_t^{t-1} as a line with $\pm 2\sqrt{P_t^{t-1}}$ error bounds as dashed lines. The middle is similar, showing $\mu_t^t \pm 2\sqrt{P_t^t}$. The bottom shows $\mu_t^n \pm 2\sqrt{P_t^n}$.

the predictions μ_t^{t-1} , for $t = 1, 2, \dots, 50$, superimposed on the graph as a line. In addition, we display $\mu_t^{t-1} \pm 2\sqrt{P_t^{t-1}}$ as dashed lines on the plot. The middle panel displays the filter, μ_t^t , for $t = 1, \dots, 50$, as a line with $\mu_t^t \pm 2\sqrt{P_t^t}$ as dashed lines. The bottom panel of Figure 6.4 shows a similar plot for the smoother μ_t^n .

Table 6.1 shows the first 10 observations as well as the corresponding state values, the predictions, filters and smoothers. Note that one-step-ahead prediction is more uncertain than the corresponding filtered value, which, in turn, is more uncertain than the corresponding smoother value (that is $P_t^{t-1} \geq P_t^t \geq P_t^n$). Also, in each case, the error variances stabilize quickly.

The R code for this example is as follows. In the example we use `Ksmooth0`, which calls `Kfilter0` for the filtering part. In the returned values from `Ksmooth0`, the letters `p`, `f`, `s` denote prediction, filter, and smooth, respectively (e.g., `xp` is x_t^{t-1} , `xf` is x_t^t , `xs` is x_t^n , and so on). These scripts use a Cholesky-type decomposition^{6.1} of Q and R ; they are denoted by `cQ` and `cR`. Practically, the scripts only require that `Q` or `R` may be reconstructed as `t(cQ)%*%(cQ)` or `t(cR)%*%(cR)`, respectively, which allows more flexibility. For example, the model (6.6) - (6.7) does not pose a problem even though the state noise covariance matrix is not positive definite.

^{6.1} Given a positive definite matrix A , its Cholesky decomposition is an upper triangular matrix U with strictly positive diagonal entries such that $A = U'U$. In R, use `chol(A)`. For the univariate case, it is simply the positive square root of A .

Table 6.1. First 10 Observations of Example 6.5

t	y_t	μ_t	μ_t^{t-1}	P_t^{t-1}	μ_t^t	P_t^t	μ_t^n	P_t^n
0	—	-.63	—	—	.00	1.00	-.32	.62
1	-1.05	-.44	.00	2.00	-.70	.67	-.65	.47
2	-.94	-1.28	-.70	1.67	-.85	.63	-.57	.45
3	-.81	.32	-.85	1.63	-.83	.62	-.11	.45
4	2.08	.65	-.83	1.62	.97	.62	1.04	.45
5	1.81	-.17	.97	1.62	1.49	.62	1.16	.45
6	-.05	.31	1.49	1.62	.53	.62	.63	.45
7	.01	1.05	.53	1.62	.21	.62	.78	.45
8	2.20	1.63	.21	1.62	1.44	.62	1.70	.45
9	1.19	1.32	1.44	1.62	1.28	.62	2.12	.45
10	5.24	2.83	1.28	1.62	3.73	.62	3.48	.45

```
# generate data
set.seed(1); num = 50
w = rnorm(num+1, 0, 1); v = rnorm(num, 0, 1)
mu = cumsum(w) # state: mu[0], mu[1], ..., mu[50]
y = mu[-1] + v # obs: y[1], ..., y[50]
# filter and smooth (Ksmooth0 does both)
ks = Ksmooth0(num, y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)
# start figure
par(mfrow=c(3,1)); Time = 1:num
plot(Time, mu[-1], main='Predict', ylim=c(-5,10))
lines(ks$xp)
lines(ks$xp+2*sqrt(ks$Pp), lty=2, col=4)
lines(ks$xp-2*sqrt(ks$Pp), lty=2, col=4)
plot(Time, mu[-1], main='Filter', ylim=c(-5,10))
lines(ks$xf)
lines(ks$xf+2*sqrt(ks$Pf), lty=2, col=4)
lines(ks$xf-2*sqrt(ks$Pf), lty=2, col=4)
plot(Time, mu[-1], main='Smooth', ylim=c(-5,10))
lines(ks$xs)
lines(ks$xs+2*sqrt(ks$Ps), lty=2, col=4)
lines(ks$xs-2*sqrt(ks$Ps), lty=2, col=4)
mu[1]; ks$x0n; sqrt(ks$P0n) # initial value info
```

When we discuss maximum likelihood estimation via the EM algorithm in the next section, we will need a set of recursions for obtaining $P_{t,t-1}^n$, as defined in (6.17). We give the necessary recursions in the following property.

Property 6.3 The Lag-One Covariance Smoother

For the state-space model specified in (6.3) and (6.4), with K_t , J_t ($t = 1, \dots, n$), and P_n^n obtained from **Property 6.1** and **Property 6.2**, and with initial condition

$$P_{n,n-1}^n = (I - K_n A_n) \Phi P_{n-1}^{n-1}, \quad (6.53)$$

for $t = n, n-1, \dots, 2$,

$$P_{t-1,t-2}^n = P_{t-1}^{t-1} J'_{t-2} + J_{t-1} \left(P_{t,t-1}^n - \Phi P_{t-1}^{t-1} \right) J'_{t-2}. \quad (6.54)$$

Proof: Because we are computing covariances, we may assume $u_t \equiv 0$ without loss of generality. To derive the initial term (6.53), we first define

$$\tilde{x}_t^s = x_t - x_t^s.$$

Then, using (6.20) and (6.45), we write

$$\begin{aligned} P_{t,t-1}^t &= E\left(\tilde{x}_t^t \tilde{x}_{t-1}^{t'}\right) \\ &= E\left\{[\tilde{x}_t^{t-1} - K_t(y_t - A_t x_t^{t-1})][\tilde{x}_{t-1}^{t-1} - J_{t-1} K_t(y_t - A_t x_t^{t-1})]'\right\} \\ &= E\left\{[\tilde{x}_t^{t-1} - K_t(A_t \tilde{x}_t^{t-1} + v_t)][\tilde{x}_{t-1}^{t-1} - J_{t-1} K_t(A_t \tilde{x}_t^{t-1} + v_t)]'\right\}. \end{aligned}$$

Expanding terms and taking expectation, we arrive at

$$P_{t,t-1}^t = P_{t,t-1}^{t-1} - P_t^{t-1} A_t' K_t' J_{t-1}' - K_t A_t P_{t,t-1}^{t-1} + K_t (A_t P_t^{t-1} A_t' + R) K_t' J_{t-1}',$$

noting $E(\tilde{x}_t^{t-1} v_t') = 0$. The final simplification occurs by realizing that $K_t (A_t P_t^{t-1} A_t' + R) = P_t^{t-1} A_t'$, and $P_{t,t-1}^{t-1} = P_t^{t-1} P_{t-1}^{t-1}$. These relationships hold for any $t = 1, \dots, n$, and (6.53) is the case $t = n$.

We give the basic steps in the derivation of (6.54). The first step is to use (6.45) to write

$$\tilde{x}_{t-1}^n + J_{t-1} x_t^n = \tilde{x}_{t-1}^{t-1} + J_{t-1} \Phi x_{t-1}^{t-1} \quad (6.55)$$

and

$$\tilde{x}_{t-2}^n + J_{t-2} x_{t-1}^n = \tilde{x}_{t-2}^{t-2} + J_{t-2} \Phi x_{t-2}^{t-2}. \quad (6.56)$$

Next, multiply the left-hand side of (6.55) by the transpose of the left-hand side of (6.56), and equate that to the corresponding result of the right-hand sides of (6.55) and (6.56). Then, taking expectation of both sides, the left-hand side result reduces to

$$P_{t-1,t-2}^n + J_{t-1} E(x_t^n x_{t-1}^{n'}) J_{t-2}' \quad (6.57)$$

and the right-hand side result reduces to

$$\begin{aligned} &P_{t-1,t-2}^{t-2} - K_{t-1} A_{t-1} P_{t-1,t-2}^{t-2} + J_{t-1} \Phi K_{t-1} A_{t-1} P_{t-1,t-2}^{t-2} \\ &+ J_{t-1} \Phi E(x_{t-1}^{t-1} x_{t-2}^{t-2'}) \Phi' J_{t-2}'. \end{aligned} \quad (6.58)$$

In (6.57), write

$$E(x_t^n x_{t-1}^{n'}) = E(x_t x_{t-1}') - P_{t,t-1}^n = \Phi E(x_{t-1} x_{t-2}') \Phi' + \Phi Q - P_{t,t-1}^n,$$

and in (6.58), write

$$E(x_{t-1}^{t-1} x_{t-2}^{t-2'}) = E(x_{t-1}^{t-2} x_{t-2}^{t-2'}) = E(x_{t-1} x_{t-2}') - P_{t-1,t-2}^{t-2}.$$

Equating (6.57) to (6.58) using these relationships and simplifying the result leads to (6.54). \square

6.3 Maximum Likelihood Estimation

Estimation of the parameters that specify the state space model, (6.3) and (6.4), is quite involved. We use Θ to represent the vector of unknown parameters in the initial mean and covariance μ_0 and Σ_0 , the transition matrix Φ , and the state and observation covariance matrices Q and R and the input coefficient matrices, Υ and Γ . We use maximum likelihood under the assumption that the initial state is normal, $x_0 \sim N_p(\mu_0, \Sigma_0)$, and the errors are normal, $w_t \sim \text{iid } N_p(0, Q)$ and $v_t \sim \text{iid } N_q(0, R)$. We continue to assume, for simplicity, $\{w_t\}$ and $\{v_t\}$ are uncorrelated.

The likelihood is computed using the *innovations* $\epsilon_1, \epsilon_2, \dots, \epsilon_n$, defined by (6.23),

$$\epsilon_t = y_t - A_t x_t^{t-1} - \Gamma u_t.$$

The innovations form of the likelihood of the data $y_{1:n}$, which was first given by Schweppe (1965), is obtained using an argument similar to the one leading to (3.117) and proceeds by noting the innovations are independent Gaussian random vectors with zero means and, as shown in (6.24), covariance matrices

$$\Sigma_t = A_t P_t^{t-1} A_t' + R. \quad (6.59)$$

Hence, ignoring a constant, we may write the likelihood, $L_Y(\Theta)$, as

$$-\ln L_Y(\Theta) = \frac{1}{2} \sum_{t=1}^n \ln |\Sigma_t(\Theta)| + \frac{1}{2} \sum_{t=1}^n \epsilon_t(\Theta)' \Sigma_t(\Theta)^{-1} \epsilon_t(\Theta), \quad (6.60)$$

where we have emphasized the dependence of the innovations on the parameters Θ . Of course, (6.60) is a highly nonlinear and complicated function of the unknown parameters. The usual procedure is to fix x_0 and then develop a set of recursions for the log likelihood function and its first two derivatives (for example, Gupta and Mehra, 1974). Then, a Newton–Raphson algorithm (see 3.30) can be used successively to update the parameter values until the negative of the log likelihood is minimized. This approach is advocated, for example, by Jones (1980), who developed ARMA estimation by putting the ARMA model in state-space form. For the univariate case, (6.60) is identical, in form, to the likelihood for the ARMA model given in (3.117).

The steps involved in performing a Newton–Raphson estimation procedure are as follows.

- (i) Select initial values for the parameters, say, $\Theta^{(0)}$.
- (ii) Run the Kalman filter, [Property 6.1](#), using the initial parameter values, $\Theta^{(0)}$, to obtain a set of innovations and error covariances, say, $\{\epsilon_t^{(0)}; t = 1, \dots, n\}$ and $\{\Sigma_t^{(0)}; t = 1, \dots, n\}$.
- (iii) Run one iteration of a Newton–Raphson procedure with $-\ln L_Y(\Theta)$ as the criterion function (refer to [Example 3.30](#) for details), to obtain a new set of estimates, say $\Theta^{(1)}$.
- (iv) At iteration j , ($j = 1, 2, \dots$), repeat step 2 using $\Theta^{(j)}$ in place of $\Theta^{(j-1)}$ to obtain a new set of innovation values $\{\epsilon_t^{(j)}; t = 1, \dots, n\}$ and $\{\Sigma_t^{(j)}; t = 1, \dots, n\}$.

Then repeat step 3 to obtain a new estimate $\Theta^{(j+1)}$. Stop when the estimates or the likelihood stabilize; for example, stop when the values of $\Theta^{(j+1)}$ differ from $\Theta^{(j)}$, or when $L_Y(\Theta^{(j+1)})$ differs from $L_Y(\Theta^{(j)})$, by some predetermined, but small amount.

Example 6.6 Newton–Raphson for Example 6.3

In this example, we generated $n = 100$ observations, $y_{1:100}$, from the AR with noise model given in [Example 6.3](#), to perform a Newton–Raphson estimation of the parameters ϕ , σ_w^2 , and σ_v^2 . In the notation of [Section 6.2](#), we would have $\Phi = \phi$, $Q = \sigma_w^2$ and $R = \sigma_v^2$. The actual values of the parameters are $\phi = .8$, $\sigma_w^2 = \sigma_v^2 = 1$.

In the simple case of an AR(1) with observational noise, initial estimation can be accomplished using the results of [Example 6.3](#). For example, using (6.15), we set

$$\phi^{(0)} = \hat{\rho}_y(2)/\hat{\rho}_y(1).$$

Similarly, from (6.14), $\gamma_x(1) = \gamma_y(1) = \phi\sigma_w^2/(1 - \phi^2)$, so that, initially, we set

$$\sigma_w^{2(0)} = (1 - \phi^{(0)^2})\hat{\gamma}_y(1)/\phi^{(0)}.$$

Finally, using (6.13) we obtain an initial estimate of σ_v^2 , namely,

$$\sigma_v^{2(0)} = \hat{\gamma}_y(0) - [\sigma_w^{2(0)} / (1 - \phi^{(0)^2})].$$

Newton–Raphson estimation was accomplished using the R program `optim`. The code used for this example is given below. In that program, we must provide an evaluation of the function to be minimized, namely, $-\ln L_Y(\Theta)$. In this case, the function call combines steps 2 and 3, using the current values of the parameters, $\Theta^{(j-1)}$, to obtain first the filtered values, then the innovation values, and then calculating the criterion function, $-\ln L_Y(\Theta^{(j-1)})$, to be minimized. We can also provide analytic forms of the gradient or *score vector*, $-\partial \ln L_Y(\Theta)/\partial \Theta$, and the *Hessian matrix*, $-\partial^2 \ln L_Y(\Theta)/\partial \Theta \partial \Theta'$, in the optimization routine, or allow the program to calculate these values numerically. In this example, we let the program proceed numerically and we note the need to be cautious when calculating gradients numerically. It is suggested in Press et al. (1993, Ch. 10) that it is better to use numerical methods for the derivatives, at least for the Hessian, along with the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method. Details on the gradient and Hessian are provided in [Problem 6.9](#) and [Problem 6.10](#); see Gupta and Mehra (1974).

```
# Generate Data
set.seed(999); num = 100
x = arima.sim(n=num+1, list(ar=.8), sd=1)
y = ts(x[-1] + rnorm(num, 0, 1))
# Initial Estimates
u = ts.intersect(y, lag(y,-1), lag(y,-2))
varu = var(u); coru = cor(u)
phi = coru[1,3]/coru[1,2]
q = (1-phi^2)*varu[1,2]/phi
r = varu[1,1] - q/(1-phi^2)
```

```

(init.par = c(phi, sqrt(q), sqrt(r))) # = .91, .51, 1.03
# Function to evaluate the likelihood
Linn = function(para){
  phi = para[1]; sigw = para[2]; sigv = para[3]
  Sigma0 = (sigw^2)/(1-phi^2); Sigma0[Sigma0<0]=0
  kf = Kfilter0(num, y, 1, mu0=0, Sigma0, phi, sigw, sigv)
  return(kf$like)
}
# Estimation (partial output shown)
est = optim(init.par, Linn, gr=NULL, method='BFGS', hessian=TRUE,
            control=list(trace=1, REPORT=1)))
SE = sqrt(diag(solve(est$hessian)))
cbind(estimate=c(phi=est$par[1],sigw=est$par[2],sigv=est$par[3]),SE)
  estimate      SE
  phi     0.814  0.081
  sigw    0.851  0.175
  sigv    0.874  0.143

```

As seen from the output, the final estimates, along with their standard errors (in parentheses), are $\hat{\phi} = .81 (.08)$, $\hat{\sigma}_w = .85 (.18)$, $\hat{\sigma}_v = .87 (.14)$. The report from `optim` yielded the following results of the estimation procedure:

```

initial value 81.313627
iter  2 value 80.169051
iter  3 value 79.866131
iter  4 value 79.222846
iter  5 value 79.021504
iter  6 value 79.014723
iter  7 value 79.014453
iter  7 value 79.014452
iter  7 value 79.014452
final value 79.014452
converged

```

Note that the algorithm converged in seven steps with the final value of the negative of the log likelihood being 79.014452. The standard errors are a byproduct of the estimation procedure, and we will discuss their evaluation later in this section, after [Property 6.4](#).

Example 6.7 Newton–Raphson for the Global Temperature Deviations

In [Example 6.2](#), we considered two different global temperature series of $n = 136$ observations each, and they are plotted in [Figure 6.3](#). In that example, we argued that both series should be measuring the same underlying climatic signal, x_t , which we model as a random walk with drift,

$$x_t = \delta + x_{t-1} + w_t.$$

Recall that the observation equation was written as

$$\begin{pmatrix} y_{t1} \\ y_{t2} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} x_t + \begin{pmatrix} v_{t1} \\ v_{t2} \end{pmatrix},$$

and the model covariance matrices are given by $Q = q_{11}$ and

$$R = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}.$$

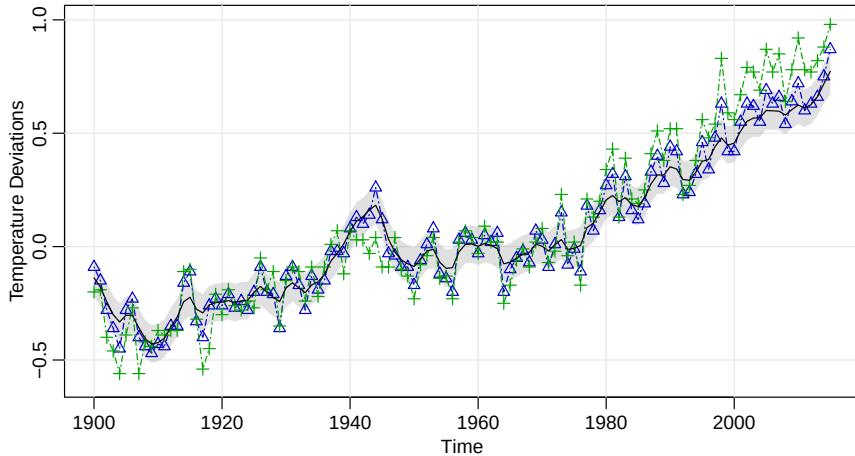


Fig. 6.5. Plot for [Example 6.7](#). The dashed lines with points (+ and Δ) are the two average global temperature deviations shown in [Figure 6.3](#). The solid line is the estimated smoother \hat{x}_t^n , and the corresponding two root mean square error bound is the gray swatch. Only the values later than 1900 are shown.

Hence, there are five parameters to estimate, δ , the drift, and the variance components, $q_{11}, r_{11}, r_{12}, r_{22}$, noting that $r_{21} = r_{12}$. We hold the initial state parameters fixed in this example at $\mu_0 = -0.35$ and $\Sigma_0 = 1$, which is large relative to the data. The final estimates were (the R matrix is reassembled in the code).

	estimate	SE
sigw	0.055	0.011
cR11	0.074	0.010
cR22	0.127	0.015
cR12	0.129	0.038
drift	0.006	0.005

The observations and the smoothed estimate of the signal, $\hat{x}_t^n \pm 2\sqrt{\hat{P}_t^n}$, are displayed in [Figure 6.5](#). The code, which uses `Kfilter1` and `Ksmooth1`, is as follows.

```
# Setup
y = cbind(globtemp, globtemp1); num = nrow(y); input = rep(1,num)
A = array(rep(1,2), dim=c(2,1,num))
mu0 = -.35; Sigma0 = 1; Phi = 1
# Function to Calculate Likelihood
Linn = function(para){
  cQ = para[1] # sigma_w
  cR1 = para[2] # 11 element of chol(R)
  cR2 = para[3] # 22 element of chol(R)
  cR12 = para[4] # 12 element of chol(R)
  cR = matrix(c(cR1,0,cR12,cR2),2) # put the matrix together
  drift = para[5]
  kf = Kfilter1(num,y,A,mu0,Sigma0,Phi,drift,0,cQ,cR,input)
  return(kf$like)
}
# Estimation
init.par = c(.1,.1,.1,0,.05) # initial values of parameters
```

```

est = optim(init.par, Linn, NULL, method='BFGS', hessian=TRUE,
            control=list(trace=1,REPORT=1)) # output not shown
SE = sqrt(diag(solve(est$hessian)))
# Display estimates
u = cbind(estimate=est$par, SE)
rownames(u)=c('sigw','cR11', 'cR22', 'cR12', 'drift'); u
# Smooth (first set parameters to their final estimates)
cQ   = est$par[1]
cR1  = est$par[2]
cR2  = est$par[3]
cR12 = est$par[4]
cR   = matrix(c(cR1,0,cR12,cR2), 2)
(R   = t(cR)%*%cR)    # to view the estimated R matrix
drift = est$par[5]
ks   = Ksmooth1(num,y,A,mu0,Sigma0,Phi,drift,0,cQ,cR,input)
# Plot
xsm = ts(as.vector(ks$xs), start=1880)
rmse = ts(sqrt(as.vector(ks$Ps)), start=1880)
plot(xsm, ylim=c(-.6, 1), ylab='Temperature Deviations')
  xx = c(time(xsm), rev(time(xsm)))
  yy = c(xsm-2*rmse, rev(xsm+2*rmse))
  polygon(xx, yy, border=NA, col=gray(.6, alpha=.25))
lines(globtemp, type='o', pch=2, col=4, lty=6)
lines(globtempl, type='o', pch=3, col=3, lty=6)

```

In addition to Newton–Raphson, Shumway and Stoffer (1982) presented a conceptually simpler estimation procedure based on the Baum-Welch algorithm (Baum et al., 1970), also known as the EM (*expectation-maximization*) algorithm (Dempster et al., 1977). For the sake of brevity, we ignore the inputs and consider the model in the form of (6.1) and (6.2). The basic idea is that if we could observe the states, $x_{0:n} = \{x_0, x_1, \dots, x_n\}$, in addition to the observations $y_{1:n} = \{y_1, \dots, y_n\}$, then we would consider $\{x_{0:n}, y_{1:n}\}$ as the *complete data*, with joint density

$$p_{\Theta}(x_{0:n}, y_{1:n}) = p_{\mu_0, \Sigma_0}(x_0) \prod_{t=1}^n p_{\Phi, Q}(x_t | x_{t-1}) \prod_{t=1}^n p_R(y_t | x_t). \quad (6.61)$$

Under the Gaussian assumption and ignoring constants, the complete data likelihood, (6.61), can be written as

$$\begin{aligned} -2 \ln L_{X,Y}(\Theta) &= \ln |\Sigma_0| + (x_0 - \mu_0)' \Sigma_0^{-1} (x_0 - \mu_0) \\ &\quad + n \ln |Q| + \sum_{t=1}^n (x_t - \Phi x_{t-1})' Q^{-1} (x_t - \Phi x_{t-1}) \\ &\quad + n \ln |R| + \sum_{t=1}^n (y_t - A_t x_t)' R^{-1} (y_t - A_t x_t). \end{aligned} \quad (6.62)$$

Thus, in view of (6.62), if we did have the complete data, we could then use the results from multivariate normal theory to easily obtain the MLEs of Θ . Although we do not have the complete data, the EM algorithm gives us an iterative method for finding the MLEs of Θ based on the *incomplete data*, $y_{1:n}$, by successively maximizing

the conditional expectation of the complete data likelihood. To implement the EM algorithm, we write, at iteration j , ($j = 1, 2, \dots$),

$$Q(\Theta | \Theta^{(j-1)}) = E \left\{ -2 \ln L_{X,Y}(\Theta) \mid y_{1:n}, \Theta^{(j-1)} \right\}. \quad (6.63)$$

Calculation of (6.63) is the *expectation step*. Of course, given the current value of the parameters, $\Theta^{(j-1)}$, we can use [Property 6.2](#) to obtain the desired conditional expectations as smoothers. This property yields

$$\begin{aligned} Q(\Theta | \Theta^{(j-1)}) &= \ln |\Sigma_0| + \text{tr} \left\{ \Sigma_0^{-1} [P_0^n + (x_0^n - \mu_0)(x_0^n - \mu_0)'] \right\} \\ &\quad + n \ln |Q| + \text{tr} \left\{ Q^{-1} [S_{11} - S_{10}\Phi' - \Phi S_{10}' + \Phi S_{00}\Phi'] \right\} \\ &\quad + n \ln |R| + \text{tr} \left\{ R^{-1} \sum_{t=1}^n [(y_t - A_t x_t^n)(y_t - A_t x_t^n)' + A_t P_t^n A_t'] \right\}, \end{aligned} \quad (6.64)$$

where

$$S_{11} = \sum_{t=1}^n (x_t^n x_t^{n'} + P_t^n), \quad (6.65)$$

$$S_{10} = \sum_{t=1}^n (x_t^n x_{t-1}^{n'} + P_{t,t-1}^n), \quad (6.66)$$

and

$$S_{00} = \sum_{t=1}^n (x_{t-1}^n x_{t-1}^{n'} + P_{t-1}^n). \quad (6.67)$$

In (6.64)–(6.67), the smoothers are calculated under the current value of the parameters $\Theta^{(j-1)}$; for simplicity, we have not explicitly displayed this fact. In obtaining $Q(\cdot | \cdot)$, we made repeated use of fact $E(x_s x_t' | y_{1:n}) = x_s^n x_t^{n'} + P_{s,t}^n$; it is important to note that one does not simply replace x_t with x_t^n in the likelihood.

Minimizing (6.64) with respect to the parameters, at iteration j , constitutes the *maximization step*, and is analogous to the usual multivariate regression approach, which yields the updated estimates

$$\Phi^{(j)} = S_{10} S_{00}^{-1}, \quad (6.68)$$

$$Q^{(j)} = n^{-1} \left(S_{11} - S_{10} S_{00}^{-1} S_{10}' \right), \quad (6.69)$$

and

$$R^{(j)} = n^{-1} \sum_{t=1}^n [(y_t - A_t x_t^n)(y_t - A_t x_t^n)' + A_t P_t^n A_t']. \quad (6.70)$$

The updates for the initial mean and variance–covariance matrix are

$$\mu_0^{(j)} = x_0^n \quad \text{and} \quad \Sigma_0^{(j)} = P_0^n \quad (6.71)$$

obtained from minimizing (6.64).

The overall procedure can be regarded as simply alternating between the Kalman filtering and smoothing recursions and the multivariate normal maximum likelihood estimators, as given by (6.68)–(6.71). Convergence results for the EM algorithm under general conditions can be found in Wu (1983). A thorough discussion of the convergence of the EM algorithm and related methods may be found in Douc et al. (2014, Appendix D). We summarize the iterative procedure as follows.

- (i) Initialize by choosing starting values for the parameters in $\{\mu_0, \Sigma_0, \Phi, Q, R\}$, say $\Theta^{(0)}$, and compute the incomplete-data likelihood, $-\ln L_Y(\Theta^{(0)})$; see (6.60).

On iteration j , ($j = 1, 2, \dots$):

- (ii) Perform the E-Step: Using the parameters $\Theta^{(j-1)}$, use Properties 6.1, 6.2, and 6.3 to obtain the smoothed values x_t^n, P_t^n and $P_{t,t-1}^n$, $t = 1, \dots, n$, and calculate S_{11}, S_{10}, S_{00} given in (6.65)–(6.67).
- (iii) Perform the M-Step: Update the estimates in $\{\mu_0, \Sigma_0, \Phi, Q, R\}$ using (6.68)–(6.71), obtaining $\Theta^{(j)}$.
- (iv) Compute the incomplete-data likelihood, $-\ln L_Y(\Theta^{(j)})$.
- (v) Repeat Steps (ii) – (iv) to convergence.

Example 6.8 EM Algorithm for Example 6.3

Using the same data generated in Example 6.6, we performed an EM algorithm estimation of the parameters ϕ , σ_w^2 and σ_v^2 as well as the initial parameters μ_0 and Σ_0 using the script `EM0`. The convergence rate of the EM algorithm compared with the Newton–Raphson procedure is slow. In this example, with convergence being claimed when the relative change in the log likelihood is less than .00001; convergence was attained after 59 iterations. The final estimates, along with their standard errors are listed below and the results are close those in Example 6.6.

	estimate	SE
phi	0.810	0.078
sigw	0.853	0.164
sigv	0.864	0.136
mu0	-1.981	NA
Sigma0	0.022	NA

Evaluation of the standard errors used a call to `fdHess` in the `nlme` R package to evaluate the Hessian at the final estimates. The `nlme` package must be loaded prior to the call to `fdHess`.

```
library(nlme) # loads package nlme
# Generate data (same as Example 6.6)
set.seed(999); num = 100
x = arima.sim(n=num+1, list(ar = .8), sd=1)
y = ts(x[-1] + rnorm(num, 0, 1))
# Initial Estimates (same as Example 6.6)
u = ts.intersect(y, lag(y,-1), lag(y,-2))
varu = var(u); coru = cor(u)
phi = coru[1,3]/coru[1,2]
q = (1-phi^2)*varu[1,2]/phi
r = varu[1,1] - q/(1-phi^2)
# EM procedure - output not shown
```

```

(em = EM0(num, y, A=1, mu0=0, Sigma0=2.8, Phi=phi, cQ=sqrt(q), cR=sqrt(r),
           max.iter=75, tol=.00001))
# Standard Errors (this uses nlme)
phi = em$Phi; cq = sqrt(em$Q); cr = sqrt(em$R)
mu0 = em$mu0; Sigma0 = em$Sigma0
para = c(phi, cq, cr)
Linn = function(para){ # to evaluate likelihood at estimates
  kf = Kfilter0(num, y, 1, mu0, Sigma0, para[1], para[2], para[3])
  return(kf$like)
}
emhess = fdHess(para, function(para) Linn(para))
SE = sqrt(diag(solve(emhess$Hessian)))
# Display Summary of Estimation
estimate = c(para, em$mu0, em$Sigma0); SE = c(SE, NA, NA)
u = cbind(estimate, SE)
rownames(u) = c('phi', 'sigw', 'sigv', 'mu0', 'Sigma0'); u

```

STEADY STATE AND ASYMPTOTIC DISTRIBUTION OF THE MLEs

The asymptotic distribution of estimators of the model parameters, say, $\hat{\theta}_n$, is studied in very general terms in Douc, Moulines, and Stoffer (2014, Chapter 13). Earlier treatments can be found in Caines (1988, Chapters 7 and 8), and in Hannan and Deistler (1988, Chapter 4). In these references, the consistency and asymptotic normality of the estimators are established under general conditions. An essential condition is the stability of the filter. Stability of the filter assures that, for large t , the innovations ϵ_t are basically copies of each other with a stable covariance matrix Σ that does not depend on t and that, asymptotically, the innovations contain all of the information about the unknown parameters. Although it is not necessary, for simplicity, we shall assume here that $A_t \equiv A$ for all t . Details on departures from this assumption can be found in Jazwinski (1970, Sections 7.6 and 7.8). We also drop the inputs and use the model in the form of (6.1) and (6.2).

For stability of the filter, we assume the eigenvalues of Φ are less than one in absolute value; this assumption can be weakened (for example, see Harvey, 1991, Section 4.3), but we retain it for simplicity. This assumption is enough to ensure the stability of the filter in that, as $t \rightarrow \infty$, the filter error covariance matrix P_t^e converges to P , the steady-state error covariance matrix, and the gain matrix K_t converges to K , the steady-state gain matrix. From these facts, it follows that the innovation covariance matrix Σ_t converges to Σ , the steady-state covariance matrix of the stable innovations; details can be found in Jazwinski (1970, Sections 7.6 and 7.8) and Anderson and Moore (1979, Section 4.4). In particular, the steady-state filter error covariance matrix, P , satisfies the Riccati equation:

$$P = \Phi[P - PA'(APA' + R)^{-1}AP]\Phi' + Q;$$

the steady-state gain matrix satisfies $K = PA'[APA' + R]^{-1}$. In [Example 6.5](#) (see [Table 6.1](#)), for all practical purposes, stability was reached by the third observation.

When the process is in steady-state, we may consider x_{t+1}^t as the steady-state predictor and interpret it as $x_{t+1}^t = E(x_{t+1} | y_t, y_{t-1}, \dots)$. As can be seen from (6.18) and (6.20), the steady-state predictor can be written as

$$x_{t+1}^t = \Phi[I - KA]x_t^{t-1} + \Phi Ky_t = \Phi x_t^{t-1} + \Phi K\epsilon_t, \quad (6.72)$$

where ϵ_t is the steady-state innovation process given by

$$\epsilon_t = y_t - E(y_t \mid y_{t-1}, y_{t-2}, \dots).$$

In the Gaussian case, $\epsilon_t \sim \text{iid } N(0, \Sigma)$, where $\Sigma = APA' + R$. In steady-state, the observations can be written as

$$y_t = Ax_t^{t-1} + \epsilon_t. \quad (6.73)$$

Together, (6.72) and (6.73) make up the *steady-state innovations form* of the dynamic linear model.

In the following property, we assume the Gaussian state space model (6.1) and (6.2), is time invariant, i.e., $A_t \equiv A$, the eigenvalues of Φ are within the unit circle and the model has the smallest possible dimension (see Hannan and Deistler, 1988, Section 2.3 for details). We denote the true parameters by Θ_0 , and we assume the dimension of Θ_0 is the dimension of the parameter space. Although it is not necessary to assume w_t and v_t are Gaussian, certain additional conditions would have to apply and adjustments to the asymptotic covariance matrix would have to be made; see Douc et al. (2014, Chapter 13).

Property 6.4 Asymptotic Distribution of the Estimators

Under general conditions, let $\widehat{\Theta}_n$ be the estimator of Θ_0 obtained by maximizing the innovations likelihood, $L_Y(\Theta)$, as given in (6.60). Then, as $n \rightarrow \infty$,

$$\sqrt{n} (\widehat{\Theta}_n - \Theta_0) \xrightarrow{d} N [0, \mathcal{I}(\Theta_0)^{-1}],$$

where $\mathcal{I}(\Theta)$ is the asymptotic information matrix given by

$$\mathcal{I}(\Theta) = \lim_{n \rightarrow \infty} n^{-1} E [-\partial^2 \ln L_Y(\Theta) / \partial \Theta \partial \Theta'].$$

For a Newton procedure, the Hessian matrix (as described in Example 6.6) at the time of convergence can be used as an estimate of $n\mathcal{I}(\Theta_0)$ to obtain estimates of the standard errors. In the case of the EM algorithm, no derivatives are calculated, but we may include a numerical evaluation of the Hessian matrix at the time of convergence to obtain estimated standard errors. Also, extensions of the EM algorithm exist, such as the SEM algorithm (Meng and Rubin, 1991), that include a procedure for the estimation of standard errors. In the examples of this section, the estimated standard errors were obtained from the numerical Hessian matrix of $-\ln L_Y(\widehat{\Theta})$, where $\widehat{\Theta}$ is the vector of parameters estimates at the time of convergence.

6.4 Missing Data Modifications

An attractive feature available within the state space framework is its ability to treat time series that have been observed irregularly over time. For example, Jones (1980) used the state-space representation to fit ARMA models to series with missing observations, and Palma and Chan (1997) used the model for estimation and forecasting of

ARFIMA series with missing observations. Shumway and Stoffer (1982) described the modifications necessary to fit multivariate state-space models via the EM algorithm when data are missing. We will discuss the procedure in detail in this section. Throughout this section, for notational simplicity, we assume the model is of the form (6.1) and (6.2).

Suppose, at a given time t , we define the partition of the $q \times 1$ observation vector into two parts, $y_t^{(1)}$, the $q_{1t} \times 1$ component of observed values, and $y_t^{(2)}$, the $q_{2t} \times 1$ component of unobserved values, where $q_{1t} + q_{2t} = q$. Then, write the partitioned observation equation

$$\begin{pmatrix} y_t^{(1)} \\ y_t^{(2)} \end{pmatrix} = \begin{bmatrix} A_t^{(1)} \\ A_t^{(2)} \end{bmatrix} x_t + \begin{pmatrix} v_t^{(1)} \\ v_t^{(2)} \end{pmatrix}, \quad (6.74)$$

where $A_t^{(1)}$ and $A_t^{(2)}$ are, respectively, the $q_{1t} \times p$ and $q_{2t} \times p$ partitioned observation matrices, and

$$\text{cov} \begin{pmatrix} v_t^{(1)} \\ v_t^{(2)} \end{pmatrix} = \begin{bmatrix} R_{11t} & R_{12t} \\ R_{21t} & R_{22t} \end{bmatrix} \quad (6.75)$$

denotes the covariance matrix of the measurement errors between the observed and unobserved parts.

In the missing data case where $y_t^{(2)}$ is not observed, we may modify the observation equation in the DLM, (6.1)–(6.2), so that the model is

$$x_t = \Phi x_{t-1} + w_t \quad \text{and} \quad y_t^{(1)} = A_t^{(1)} x_t + v_t^{(1)}, \quad (6.76)$$

where now, the observation equation is q_{1t} -dimensional at time t . In this case, it follows directly from [Corollary 6.1](#) that the filter equations hold with the appropriate notational substitutions. If there are no observations at time t , then set the gain matrix, K_t , to the $p \times q$ zero matrix in [Property 6.1](#), in which case $x_t^t = x_t^{t-1}$ and $P_t^t = P_t^{t-1}$.

Rather than deal with varying observational dimensions, it is computationally easier to modify the model by zeroing out certain components and retaining a q -dimensional observation equation throughout. In particular, [Corollary 6.1](#) holds for the missing data case if, at update t , we substitute

$$y_{(t)} = \begin{pmatrix} y_t^{(1)} \\ 0 \end{pmatrix}, \quad A_{(t)} = \begin{bmatrix} A_t^{(1)} \\ 0 \end{bmatrix}, \quad R_{(t)} = \begin{bmatrix} R_{11t} & 0 \\ 0 & I_{22t} \end{bmatrix}, \quad (6.77)$$

for y_t , A_t , and R , respectively, in (6.20)–(6.22), where I_{22t} is the $q_{2t} \times q_{2t}$ identity matrix. With the substitutions (6.77), the innovation values (6.23) and (6.24) will now be of the form

$$\epsilon_{(t)} = \begin{pmatrix} \epsilon_t^{(1)} \\ 0 \end{pmatrix}, \quad \Sigma_{(t)} = \begin{bmatrix} A_t^{(1)} P_t^{t-1} A_t^{(1)'} + R_{11t} & 0 \\ 0 & I_{22t} \end{bmatrix}, \quad (6.78)$$

so that the innovations form of the likelihood given in (6.60) is correct for this case. Hence, with the substitutions in (6.77), maximum likelihood estimation via the innovations likelihood can proceed as in the complete data case.

Once the missing data filtered values have been obtained, Stoffer (1982) also established the smoother values can be processed using [Property 6.2](#) and [Property 6.3](#) with the values obtained from the missing data-filtered values. In the missing data case, the state estimators are denoted

$$x_t^{(s)} = E \left(x_t \mid y_1^{(1)}, \dots, y_s^{(1)} \right), \quad (6.79)$$

with error variance–covariance matrix

$$P_t^{(s)} = E \left\{ \left(x_t - x_t^{(s)} \right) \left(x_t - x_t^{(s)} \right)' \right\}. \quad (6.80)$$

The missing data lag-one smoother covariances will be denoted by $P_{t,t-1}^{(n)}$.

The maximum likelihood estimators in the EM procedure require further modifications for the case of missing data. Now, we consider

$$y_{1:n}^{(1)} = \{y_1^{(1)}, \dots, y_n^{(1)}\} \quad (6.81)$$

as the incomplete data, and $\{x_{0:n}, y_{1:n}\}$, as defined in [\(6.61\)](#), as the complete data. In this case, the complete data likelihood, [\(6.61\)](#), or equivalently [\(6.62\)](#), is the same, but to implement the E-step, at iteration j , we must calculate

$$\begin{aligned} Q(\Theta \mid \Theta^{(j-1)}) &= E \left\{ -2 \ln L_{X,Y}(\Theta) \mid y_{1:n}^{(1)}, \Theta^{(j-1)} \right\} \\ &= E_* \left\{ \ln |\Sigma_0| + \text{tr} \Sigma_0^{-1} (x_0 - \mu_0)(x_0 - \mu_0)' \mid y_{1:n}^{(1)} \right\} \\ &\quad + E_* \left\{ n \ln |Q| + \sum_{t=1}^n \text{tr} [Q^{-1} (x_t - \Phi x_{t-1})(x_t - \Phi x_{t-1})'] \mid y_{1:n}^{(1)} \right\} \\ &\quad + E_* \left\{ n \ln |R| + \sum_{t=1}^n \text{tr} [R^{-1} (y_t - A_t x_t)(y_t - A_t x_t)'] \mid y_{1:n}^{(1)} \right\}, \end{aligned} \quad (6.82)$$

where E_* denotes the conditional expectation under $\Theta^{(j-1)}$ and tr denotes trace. The first two terms in [\(6.82\)](#) will be like the first two terms of [\(6.64\)](#) with the smoothers x_t^n , P_t^n , and $P_{t,t-1}^n$ replaced by their missing data counterparts, $x_t^{(n)}$, $P_t^{(n)}$, and $P_{t,t-1}^{(n)}$. In the third term of [\(6.82\)](#), we must additionally evaluate $E_*(y_t^{(2)} \mid y_{1:n}^{(1)})$ and $E_*(y_t^{(2)} y_t^{(2)'} \mid y_{1:n}^{(1)})$. In Stoffer (1982), it is shown that

$$\begin{aligned} E_* \left\{ (y_t - A_t x_t)(y_t - A_t x_t)' \mid y_{1:n}^{(1)} \right\} \\ = \begin{pmatrix} y_t^{(1)} - A_t^{(1)} x_t^{(n)} \\ R_{*21t} R_{*11t}^{-1} (y_t^{(1)} - A_t^{(1)} x_t^{(n)}) \end{pmatrix} \begin{pmatrix} y_t^{(1)} - A_t^{(1)} x_t^{(n)} \\ R_{*21t} R_{*11t}^{-1} (y_t^{(1)} - A_t^{(1)} x_t^{(n)}) \end{pmatrix}' \\ + \begin{pmatrix} A_t^{(1)} \\ R_{*21t} R_{*11t}^{-1} A_t^{(1)} \end{pmatrix} P_t^{(n)} \begin{pmatrix} A_t^{(1)} \\ R_{*21t} R_{*11t}^{-1} A_t^{(1)} \end{pmatrix}' \\ + \begin{pmatrix} 0 & 0 \\ 0 & R_{*22t} - R_{*21t} R_{*11t}^{-1} R_{*12t} \end{pmatrix}. \end{aligned} \quad (6.83)$$

In (6.83), the values of R_{*ikt} , for $i, k = 1, 2$, are the current values specified by $\Theta^{(j-1)}$. In addition, $x_t^{(n)}$ and $P_t^{(n)}$ are the values obtained by running the smoother under the current parameter estimates specified by $\Theta^{(j-1)}$.

In the case in which observed and unobserved components have uncorrelated errors, that is, R_{*12t} is the zero matrix, (6.83) can be simplified to

$$\begin{aligned} \mathbb{E}_* \{ & (y_t - A_t x_t)(y_t - A_t x_t)' \mid y_{1:n}^{(1)} \} \\ &= (y_{(t)} - A_{(t)} x_t^{(n)}) (y_{(t)} - A_{(t)} x_t^{(n)})' + A_{(t)} P_t^{(n)} A_{(t)}' + \begin{pmatrix} 0 & 0 \\ 0 & R_{*22t} \end{pmatrix}, \end{aligned} \quad (6.84)$$

where $y_{(t)}$ and $A_{(t)}$ are defined in (6.77).

In this simplified case, the missing data M-step looks like the M-step given in (6.65)-(6.71). That is, with

$$S_{(11)} = \sum_{t=1}^n (x_t^{(n)} x_t^{(n)'} + P_t^{(n)}), \quad (6.85)$$

$$S_{(10)} = \sum_{t=1}^n (x_t^{(n)} x_{t-1}^{(n)'} + P_{t,t-1}^{(n)}), \quad (6.86)$$

and

$$S_{(00)} = \sum_{t=1}^n (x_{t-1}^{(n)} x_{t-1}^{(n)'} + P_{t-1}^{(n)}), \quad (6.87)$$

where the smoothers are calculated under the present value of the parameters $\Theta^{(j-1)}$ using the missing data modifications, at iteration j , the *maximization step* is

$$\Phi^{(j)} = S_{(10)} S_{(00)}^{-1}, \quad (6.88)$$

$$Q^{(j)} = n^{-1} \left(S_{(11)} - S_{(10)} S_{(00)}^{-1} S_{(10)}' \right), \quad (6.89)$$

and

$$\begin{aligned} R^{(j)} = n^{-1} \sum_{t=1}^n D_t \left\{ & \left(y_{(t)} - A_{(t)} x_t^{(n)} \right) \left(y_{(t)} - A_{(t)} x_t^{(n)} \right)' \right. \\ & \left. + A_{(t)} P_t^{(n)} A_{(t)}' + \begin{pmatrix} 0 & 0 \\ 0 & R_{22t}^{(j-1)} \end{pmatrix} \right\} D_t', \end{aligned} \quad (6.90)$$

where D_t is a permutation matrix that reorders the variables at time t in their original order and $y_{(t)}$ and $A_{(t)}$ are defined in (6.77). For example, suppose $q = 3$ and at time t , y_{t2} is missing. Then,

$$y_{(t)} = \begin{pmatrix} y_{t1} \\ y_{t3} \\ 0 \end{pmatrix}, \quad A_{(t)} = \begin{bmatrix} A_{t1} \\ A_{t3} \\ 0' \end{bmatrix}, \quad \text{and} \quad D_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

where A_{ti} is the i th row of A_t and $0'$ is a $1 \times p$ vector of zeros. In (6.90), only R_{11t} gets updated, and R_{22t} at iteration j is simply set to its value from the previous iteration, $j - 1$. Of course, if we cannot assume $R_{12t} = 0$, (6.90) must be changed accordingly using (6.83), but (6.88) and (6.89) remain the same. As before, the parameter estimates for the initial state are updated as

$$\mu_0^{(j)} = x_0^{(n)} \quad \text{and} \quad \Sigma_0^{(j)} = P_0^{(n)}. \quad (6.91)$$

Example 6.9 Longitudinal Biomedical Data

We consider the biomedical data in Example 6.1, which have portions of the three-dimensional vector missing after the 40th day. The maximum likelihood procedure yielded the estimators (code at the end of the example):

```
$Phi
      [,1]   [,2]   [,3]
[1,]  0.984 -0.041  0.009
[2,]  0.061  0.921  0.007
[3,] -1.495  2.289  0.794

$Q
      [,1]   [,2]   [,3]
[1,]  0.014 -0.002  0.012
[2,] -0.002  0.003  0.018
[3,]  0.012  0.018  3.494

$R
      [,1]   [,2]   [,3]
[1,]  0.007  0.000  0.000
[2,]  0.000  0.017  0.000
[3,]  0.000  0.000  1.147
```

for the transition, state error covariance and observation error covariance matrices, respectively. The coupling between the first and second series is relatively weak, whereas the third series HCT is strongly related to the first two; that is,

$$\hat{x}_{t3} = -1.495x_{t-1,1} + 2.289x_{t-1,2} + .794x_{t-1,3}.$$

Hence, the HCT is negatively correlated with white blood count (WBC) and positively correlated with platelet count (PLT). Byproducts of the procedure are estimated trajectories for all three longitudinal series and their respective prediction intervals. In particular, Figure 6.6 shows the data as points, the estimated smoothed values $\hat{x}_t^{(n)}$ as solid lines, and error bounds, $\pm 2\sqrt{\hat{P}_t^{(n)}}$ as a gray swatch.

In the following R code we use the script EM1. In this case the observation matrices A_t are either the identity or zero matrix because all the series are either observed or not observed.

```
y = cbind(WBC, PLT, HCT); num = nrow(y)
# make array of obs matrices
A = array(0, dim=c(3,3,num))
for(k in 1:num) { if (y[k,1] > 0) A[, ,k] = diag(1,3) }
# Initial values
mu0 = matrix(0, 3, 1); Sigma0 = diag(c(.1, .1, 1), 3)
Phi = diag(1, 3); cQ = diag(c(.1, .1, 1), 3); cR = diag(c(.1, .1, 1), 3)
# EM procedure - some output previously shown
```

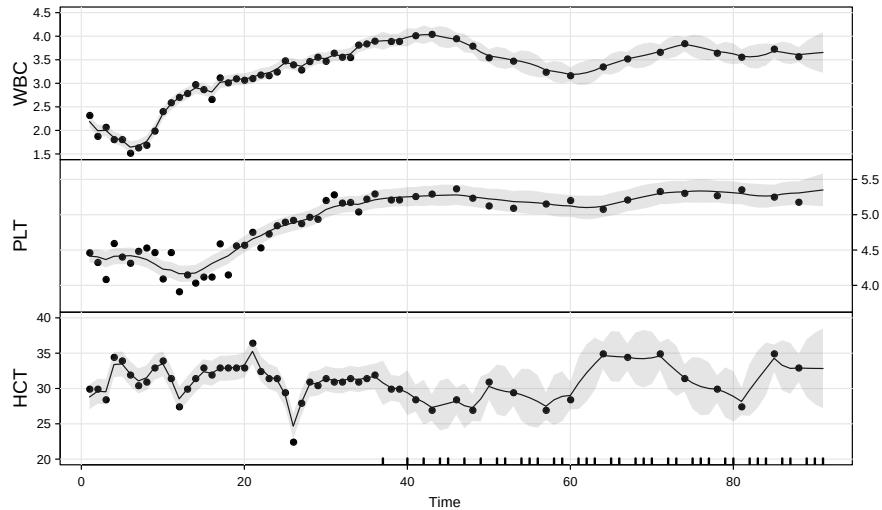


Fig. 6.6. Smoothed values for various components in the blood parameter tracking problem. The actual data are shown as points, the smoothed values are shown as solid lines, and ± 2 standard error bounds are shown as a gray swatch; tick marks indicate days with no observation.

```
(em = EM1(num, y, A, mu0, Sigma0, Phi, cQ, cR, 100, .001))
# Graph smoother
ks = Ksmooth1(num, y, A, em$mu0, em$Sigma0, em$Phi, 0, 0, chol(em$Q),
               chol(em$R), 0)
y1s = ks$xs[1,,]; y2s = ks$xs[2,,]; y3s = ks$xs[3,,]
p1 = 2*sqrt(ks$Ps[1,1,]); p2 = 2*sqrt(ks$Ps[2,2,]); p3 = 2*sqrt(ks$Ps[3,3,])
par(mfrow=c(3,1))
plot(WBC, type='p', pch=19, ylim=c(1,5), xlab='day')
lines(y1s+p1, lty=2, col=4); lines(y1s-p1, lty=2, col=4)
plot(PLT, type='p', ylim=c(3,6), pch=19, xlab='day')
lines(y2s+p2, lty=2, col=4); lines(y2s-p2, lty=2, col=4)
plot(HCT, type='p', pch=19, ylim=c(20,40), xlab='day')
lines(y3s+p3, lty=2, col=4); lines(y3s-p3, lty=2, col=4)
```

6.5 Structural Models: Signal Extraction and Forecasting

Structural models are component models in which each component may be thought of as explaining a specific type of behavior. The models are often some version of the classical time series decomposition of data into trend, seasonal, and irregular components. Consequently, each component has a direct interpretation as to the nature of the variation in the data. Furthermore, the model fits into the state space framework quite easily. To illustrate these ideas, we consider an example that shows how to fit a sum of trend, seasonal, and irregular components to the quarterly earnings data that we have considered before.

Example 6.10 Johnson & Johnson Quarterly Earnings

Here, we focus on the quarterly earnings series from the U.S. company Johnson & Johnson as displayed in [Figure 1.1](#). The series is highly nonstationary, and there is both a trend signal that is gradually increasing over time and a seasonal component that cycles every four quarters or once per year. The seasonal component is getting larger over time as well. Transforming into logarithms or even taking the n th root does not seem to make the series trend stationary, however, such a transformation does help with stabilizing the variance over time; this is explored in [Problem 6.13](#). Suppose, for now, we consider the series to be the sum of a trend component, a seasonal component, and a white noise. That is, let the observed series be expressed as

$$y_t = T_t + S_t + v_t, \quad (6.92)$$

where T_t is trend and S_t is the seasonal component. Suppose we allow the trend to increase exponentially; that is,

$$T_t = \phi T_{t-1} + w_{t1}, \quad (6.93)$$

where the coefficient $\phi > 1$ characterizes the increase. Let the seasonal component be modeled as

$$S_t + S_{t-1} + S_{t-2} + S_{t-3} = w_{t2}, \quad (6.94)$$

which corresponds to assuming the component is expected to sum to zero over a complete period or four quarters. To express this model in state-space form, let $x_t = (T_t, S_t, S_{t-1}, S_{t-2})'$ be the state vector so the observation equation [\(6.2\)](#) can be written as

$$y_t = (1 \ 1 \ 0 \ 0) \begin{pmatrix} T_t \\ S_t \\ S_{t-1} \\ S_{t-2} \end{pmatrix} + v_t,$$

with the state equation written as

$$\begin{pmatrix} T_t \\ S_t \\ S_{t-1} \\ S_{t-2} \end{pmatrix} = \begin{pmatrix} \phi & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} T_{t-1} \\ S_{t-1} \\ S_{t-2} \\ S_{t-3} \end{pmatrix} + \begin{pmatrix} w_{t1} \\ w_{t2} \\ 0 \\ 0 \end{pmatrix},$$

where $R = r_{11}$ and

$$Q = \begin{pmatrix} q_{11} & 0 & 0 & 0 \\ 0 & q_{22} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The model reduces to state-space form, [\(6.1\)](#) and [\(6.2\)](#), with $p = 4$ and $q = 1$. The parameters to be estimated are r_{11} , the noise variance in the measurement equations, q_{11} and q_{22} , the model variances corresponding to the trend and seasonal components and ϕ , the transition parameter that models the growth rate. Growth

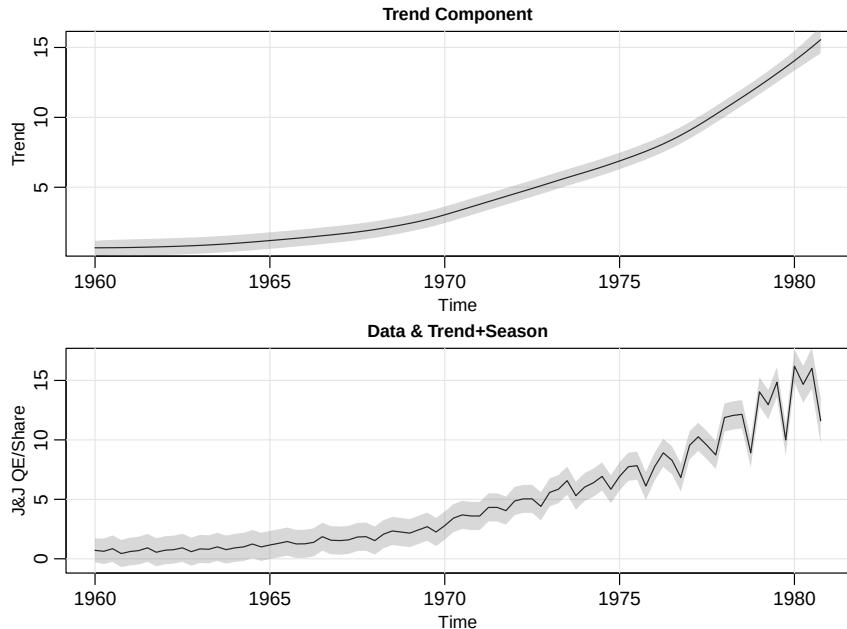


Fig. 6.7. Estimated trend component, T_t^n , and seasonal component, S_t^n , of the Johnson and Johnson quarterly earnings series. Gray areas are three root MSE bounds.

is about 3% per year, and we began with $\phi = 1.03$. The initial mean was fixed at $\mu_0 = (.7, 0, 0, 0)'$, with uncertainty modeled by the diagonal covariance matrix with $\Sigma_{0ii} = .04$, for $i = 1, \dots, 4$. Initial state covariance values were taken as $q_{11} = .01$, $q_{22} = .01$. The measurement error covariance was started at $r_{11} = .25$.

After about 20 iterations of a Newton–Raphson, the transition parameter estimate was $\hat{\phi} = 1.035$, corresponding to exponential growth with inflation at about 3.5% per year. The measurement uncertainty was small at $\sqrt{r_{11}} = .0005$, compared with the model uncertainties $\sqrt{q_{11}} = .1397$ and $\sqrt{q_{22}} = .2209$. Figure 6.7 shows the smoothed trend estimate and the exponentially increasing seasonal components. We may also consider forecasting the Johnson & Johnson series, and the result of a 12-quarter forecast is shown in Figure 6.8 as basically an extension of the latter part of the observed data.

This example uses the `Kfilter0` and `Ksmooth0` scripts as follows.

```

num = length(jj)
A = cbind(1,1,0,0)
# Function to Calculate Likelihood
Linn =function(para){
  Phi = diag(0,4); Phi[1,1] = para[1]
  Phi[2,]=c(0,-1,-1,-1); Phi[3,]=c(0,1,0,0); Phi[4,]=c(0,0,1,0)
  cQ1 = para[2]; cQ2 = para[3]      # sqrt q11 and q22
  cQ = diag(0,4); cQ[1,1]=cQ1; cQ[2,2]=cQ2
  cR = para[4]                      # sqrt r11
  kf = Kfilter0(num, jj, A, mu0, Sigma0, Phi, cQ, cR)
}

```

```

    return(kf$like)  }
# Initial Parameters
mu0 = c(.7,0,0,0); Sigma0 = diag(.04,4)
init.par = c(1.03,.1,.1,.5)      # Phi[1,1], the 2 cQs and cR
# Estimation and Results
est = optim(init.par, Linn,NULL, method='BFGS', hessian=TRUE,
            control=list(trace=1,REPORT=1))
SE = sqrt(diag(solve(est$hessian)))
u = cbind(estimate=est$par, SE)
rownames(u)=c('Phi11','sigw1','sigw2','sigv'); u
# Smooth
Phi = diag(0.4); Phi[1,1] = est$par[1]
Phi[2,]=c(0,-1,-1,-1); Phi[3,]=c(0,1,0,0); Phi[4,]=c(0,0,1,0)
cQ1 = est$par[2]; cQ2 = est$par[3]
cQ = diag(1,4); cQ[1,1]=cQ1; cQ[2,2]=cQ2
cR = est$par[4]
ks = Ksmooth0(num,jj,A,mu0,Sigma0,Phi,cQ,cR)
# Plots
Tsm = ts(ks$xs[,], start=1960, freq=4)
Ssm = ts(ks$xs[,], start=1960, freq=4)
p1 = 3*sqrt(ks$Ps[1,1,]); p2 = 3*sqrt(ks$Ps[2,2,])
par(mfrow=c(2,1))
plot(Tsm, main='Trend Component', ylab='Trend')
xx = c(time(jj), rev(time(jj)))
yy = c(Tsm-p1, rev(Tsm+p1))
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))
plot(jj, main='Data & Trend+Season', ylab='J&J QE/Share', ylim=c(-.5,17))
xx = c(time(jj), rev(time(jj)) )
yy = c((Tsm+Ssm)-(p1+p2), rev((Tsm+Ssm)+(p1+p2)) )
polygon(xx, yy, border=NA, col=gray(.5, alpha = .3))
# Forecast
n.ahead = 12;
y = ts	append(jj, rep(0,n.ahead)), start=1960, freq=4)
rmspe = rep(0,n.ahead); x00 = ks$xf[,num]; P00 = ks$Pf[,num]
Q = t(cQ)%*%cQ; R = t(cR)%*%(cR)
for (m in 1:n.ahead){
  xp = Phi%*%x00; Pp = Phi%*%P00%*%t(Phi)+Q
  sig = A%*%Pp%*%t(A)+R; K = Pp%*%t(A)%*%(1/sig)
  x00 = xp; P00 = Pp-K%*%A%*%Pp
  y[num+m] = A%*%xp; rmspe[m] = sqrt(sig)  }
plot(y, type='o', main='', ylab='J&J QE/Share', ylim=c(5,30),
      xlim=c(1975,1984))
upp = ts(y[(num+1):(num+n.ahead)]+2*rmspe, start=1981, freq=4)
low = ts(y[(num+1):(num+n.ahead)]-2*rmspe, start=1981, freq=4)
xx = c(time(low), rev(time(upp)))
yy = c(low, rev(upp))
polygon(xx, yy, border=8, col=gray(.5, alpha = .3))
abline(v=1981, lty=3)

```

Note that the Cholesky decomposition of `Q` does not exist here, however, the diagonal form allows us to use standard deviations for the first two diagonal elements of `cQ`. This technicality can be avoided using a form of the model that we present in the next section.

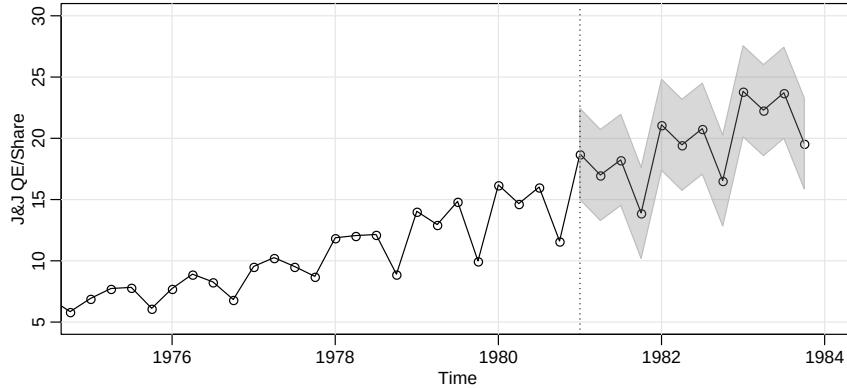


Fig. 6.8. A 12-quarter forecast for the Johnson & Johnson quarterly earnings series. The forecasts are shown as a continuation of the data (points connected by a solid line). The gray area represents two root MSPE bounds.

6.6 State-Space Models with Correlated Errors

Sometimes it is advantageous to write the state-space model in a slightly different way, as is done by numerous authors; for example, Anderson and Moore (1979) and Hannan and Deistler (1988). Here, we write the state-space model as

$$x_{t+1} = \Phi x_t + \Upsilon u_{t+1} + \Theta w_t \quad t = 0, 1, \dots, n \quad (6.95)$$

$$y_t = A_t x_t + \Gamma u_t + v_t \quad t = 1, \dots, n \quad (6.96)$$

where, in the state equation, $x_0 \sim N_p(\mu_0, \Sigma_0)$, Φ is $p \times p$, and Υ is $p \times r$, Θ is $p \times m$ and $w_t \sim \text{iid } N_m(0, Q)$. In the observation equation, A_t is $q \times p$ and Γ is $q \times r$, and $v_t \sim \text{iid } N_q(0, R)$. In this model, while w_t and v_t are still white noise series (both independent of x_0), we also allow the state noise and observation noise to be correlated at time t ; that is,

$$\text{cov}(w_s, v_t) = S \delta_s^t, \quad (6.97)$$

where δ_s^t is Kronecker's delta; note that S is an $m \times q$ matrix. The major difference between this form of the model and the one specified by (6.3)–(6.4) is that this model starts the state noise process at $t = 0$ in order to ease the notation related to the concurrent covariance between w_t and v_t . Also, the inclusion of the matrix Θ allows us to avoid using a singular state noise process as was done in Example 6.10.

To obtain the innovations, $\epsilon_t = y_t - A_t x_t^{t-1} - \Gamma u_t$, and the innovation variance $\Sigma_t = A_t P_t^{t-1} A_t' + R$, in this case, we need the one-step-ahead state predictions. Of course, the filtered estimates will also be of interest, and they will be needed for smoothing. **Property 6.2** (the smoother) as displayed in Section 6.2 still holds. The following property generates the predictor x_{t+1}^t from the past predictor x_t^{t-1} when the noise terms are correlated and exhibits the filter update.

Property 6.5 The Kalman Filter with Correlated Noise

For the state-space model specified in (6.95) and (6.96), with initial conditions x_1^0 and P_1^0 , for $t = 1, \dots, n$,

$$x_{t+1}^t = \Phi x_t^{t-1} + \Upsilon u_{t+1} + K_t \epsilon_t \quad (6.98)$$

$$P_{t+1}^t = \Phi P_t^{t-1} \Phi' + \Theta Q \Theta' - K_t \Sigma_t K_t' \quad (6.99)$$

where $\epsilon_t = y_t - A_t x_t^{t-1} - \Gamma u_t$ and the gain matrix is given by

$$K_t = [\Phi P_t^{t-1} A_t' + \Theta S][A_t P_t^{t-1} A_t' + R]^{-1}. \quad (6.100)$$

The filter values are given by

$$x_t^t = x_t^{t-1} + P_t^{t-1} A_t' [A_t P_t^{t-1} A_t' + R]^{-1} \epsilon_t, \quad (6.101)$$

$$P_t^t = P_t^{t-1} - P_t^{t-1} A_{t+1}' [A_t P_t^{t-1} A_t' + R]^{-1} A_t P_t^{t-1}. \quad (6.102)$$

The derivation of Property 6.5 is similar to the derivation of the Kalman filter in Property 6.1 (Problem 6.17); we note that the gain matrix K_t differs in the two properties. The filter values, (6.101)–(6.102), are symbolically identical to (6.18) and (6.19). To initialize the filter, we note that

$$x_1^0 = E(x_1) = \Phi \mu_0 + \Upsilon u_1, \quad \text{and} \quad P_1^0 = \text{var}(x_1) = \Phi \Sigma_0 \Phi' + \Theta Q \Theta'.$$

In the next two subsections, we show how to use the model (6.95)–(6.96) for fitting ARMAX models and for fitting (multivariate) regression models with autocorrelated errors. To put it succinctly, for ARMAX models, the inputs enter in the state equation and for regression with autocorrelated errors, the inputs enter in the observation equation. It is, of course, possible to combine the two models and we give an example of this at the end of the section.

6.6.1 ARMAX Models

Consider a k -dimensional ARMAX model given by

$$y_t = \Upsilon u_t + \sum_{j=1}^p \Phi_j y_{t-j} + \sum_{k=1}^q \Theta_k v_{t-k} + v_t. \quad (6.103)$$

The observations y_t are a k -dimensional vector process, the Φ s and Θ s are $k \times k$ matrices, Υ is $k \times r$, u_t is the $r \times 1$ input, and v_t is a $k \times 1$ white noise process; in fact, (6.103) and (5.91) are identical models, but here, we have written the observations as y_t . We now have the following property.

Property 6.6 A State-Space Form of ARMAX

For $p \geq q$, let

$$F = \begin{bmatrix} \Phi_1 & I & 0 & \cdots & 0 \\ \Phi_2 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Phi_{p-1} & 0 & 0 & \cdots & I \\ \Phi_p & 0 & 0 & \cdots & 0 \end{bmatrix} \quad G = \begin{bmatrix} \Theta_1 + \Phi_1 \\ \vdots \\ \Theta_q + \Phi_q \\ \Phi_{q+1} \\ \vdots \\ \Phi_p \end{bmatrix} \quad H = \begin{bmatrix} \gamma \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (6.104)$$

where F is $kp \times kp$, G is $kp \times k$, and H is $kp \times r$. Then, the state-space model given by

$$x_{t+1} = Fx_t + Hu_{t+1} + Gv_t, \quad (6.105)$$

$$y_t = Ax_t + v_t, \quad (6.106)$$

where $A = [I, 0, \dots, 0]$ is $k \times pk$ and I is the $k \times k$ identity matrix, implies the ARMAX model (6.103). If $p < q$, set $\Phi_{p+1} = \dots = \Phi_q = 0$, in which case $p = q$ and (6.105)–(6.106) still apply. Note that the state process is kp -dimensional, whereas the observations are k -dimensional.

We do not prove Property 6.6 directly, but the following example should suggest how to establish the general result.

Example 6.11 Univariate ARMAX(1, 1) in State-Space Form

Consider the univariate ARMAX(1, 1) model

$$y_t = \alpha_t + \phi y_{t-1} + \theta v_{t-1} + v_t,$$

where $\alpha_t = \gamma u_t$ to ease the notation. For a simple example, if $\gamma = (\beta_0, \beta_1)$ and $u_t = (1, t)'$, the model for y_t would be ARMA(1,1) with linear trend, $y_t = \beta_0 + \beta_1 t + \phi y_{t-1} + \theta v_{t-1} + v_t$. Using Property 6.6, we can write the model as

$$x_{t+1} = \phi x_t + \alpha_{t+1} + (\theta + \phi)v_t, \quad (6.107)$$

and

$$y_t = x_t + v_t. \quad (6.108)$$

In this case, (6.107) is the state equation with $w_t \equiv v_t$ and (6.108) is the observation equation. Consequently, $\text{cov}(w_t, v_t) = \text{var}(v_t) = R$, and $\text{cov}(w_t, v_s) = 0$ when $s \neq t$, so Property 6.5 would apply. To verify (6.107) and (6.108) specify an ARMAX(1, 1) model, we have

$$\begin{aligned} y_t &= x_t + v_t && \text{from (6.108)} \\ &= \phi x_{t-1} + \alpha_t + (\theta + \phi)v_{t-1} + v_t && \text{from (6.107)} \\ &= \alpha_t + \phi(x_{t-1} + v_{t-1}) + \theta v_{t-1} + v_t && \text{rearrange terms} \\ &= \alpha_t + \phi y_{t-1} + \theta v_{t-1} + v_t, && \text{from (6.108).} \end{aligned}$$

Together, [Property 6.5](#) and [Property 6.6](#) can be used to accomplish maximum likelihood estimation as described in [Section 6.3](#) for ARMAX models. The ARMAX model is only a special case of the model (6.95)–(6.96), which is quite rich, as will be discovered in the next subsection.

6.6.2 Multivariate Regression with Autocorrelated Errors

In regression with autocorrelated errors, we are interested in fitting the regression model

$$y_t = \Gamma u_t + \varepsilon_t \quad (6.109)$$

to a $k \times 1$ vector process, y_t , with r regressors $u_t = (u_{t1}, \dots, u_{tr})'$ where ε_t is vector ARMA(p, q) and Γ is a $k \times r$ matrix of regression parameters. We note that the regressors do not have to vary with time (e.g., $u_{t1} \equiv 1$ includes a constant in the regression) and that the case $k = 1$ was treated in [Section 3.8](#).

To put the model in state-space form, we simply notice that $\varepsilon_t = y_t - \Gamma u_t$ is a k -dimensional ARMA(p, q) process. Thus, if we set $H = 0$ in (6.105), and include Γu_t in (6.106), we obtain

$$x_{t+1} = F x_t + G v_t, \quad (6.110)$$

$$y_t = \Gamma u_t + A x_t + v_t, \quad (6.111)$$

where the model matrices A , F , and G are defined in [Property 6.6](#). The fact that (6.110)–(6.111) is multivariate regression with autocorrelated errors follows directly from [Property 6.6](#) by noticing that together, $x_{t+1} = F x_t + G v_t$ and $\varepsilon_t = A x_t + v_t$ imply $\varepsilon_t = y_t - \Gamma u_t$ is vector ARMA(p, q).

As in the case of ARMAX models, regression with autocorrelated errors is a special case of the state-space model, and the results of [Property 6.5](#) can be used to obtain the innovations form of the likelihood for parameter estimation.

Example 6.12 Mortality, Temperature and Pollution

This example combines both techniques of [Section 6.6.1](#) and [Section 6.6.2](#). We will fit an ARMAX model to the detrended mortality series `cmort`. The detrending part of the example constitutes the regression with autocorrelated errors.

Here, we let M_t denote the weekly cardiovascular mortality series, T_t as the corresponding temperature series `tempr`, and P_t as the corresponding particulate series. A preliminary analysis suggests the following considerations (no output is shown):

- An AR(2) model fits well to detrended M_t :

```
fit1 = sarima(cmort, 2, 0, 0, xreg=time(cmort))
```

- The CCF between the mortality residuals, the temperature series and the particulates series, shows a strong correlation with temperature lagged one week (T_{t-1}), concurrent particulate level (P_t) and the particulate level about one month prior (P_{t-4}).

```
acf(cbind(dmort <- resid(fit1$fit), tempr, part))
lag2.plot(tempr, dmort, 8)
lag2.plot(part, dmort, 8)
```

From these results, we decided to fit the ARMAX model

$$\tilde{M}_t = \phi_1 \tilde{M}_{t-1} + \phi_2 \tilde{M}_{t-2} + \beta_1 T_{t-1} + \beta_2 P_t + \beta_3 P_{t-4} + v_t \quad (6.112)$$

to the detrended mortality series, $\tilde{M}_t = M_t - (\alpha + \beta_4 t)$, where $v_t \sim \text{iid } N(0, \sigma_v^2)$. To write the model in state-space form using [Property 6.6](#), let

$$x_{t+1} = \Phi x_t + \Upsilon u_{t+1} + \Theta v_t \quad t = 0, 1, \dots, n$$

$$y_t = \alpha + Ax_t + \Gamma u_t + v_t \quad t = 1, \dots, n$$

with

$$\Phi = \begin{bmatrix} \phi_1 & 1 \\ \phi_2 & 0 \end{bmatrix} \quad \Upsilon = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \Theta = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}$$

$A = [1 \ 0]$, $\Gamma = [0 \ 0 \ 0 \ \beta_4 \ \alpha]$, $u_t = (T_{t-1}, P_t, P_{t-4}, t, 1)'$, $y_t = M_t$. Note that the state process is bivariate and the observation process is univariate.

Some additional data analysis notes are: (1) Time is centered as $t - \bar{t}$. In this case, α should be close to the average value of M_t . (2) P_t and P_{t-4} are highly correlated, so orthogonalizing these two inputs would be advantageous (although we did not do it here), perhaps by partialling out P_{t-4} from P_t using simple linear regression. (3) T_t and T_t^2 , as in [Chapter 2](#), are not needed in the model when T_{t-1} is included. (4) Initial values of the parameters are taken from a preliminary investigation that we discuss now.

A quick and dirty method for fitting the model is to first detrend `cmort` and then fit (6.112) using `lm` on the detrended series. Rather than use `lm` in the second phase, we use `sarima` because it also provides a thorough analysis of the residuals. The code for this run is quite simple; the residual analysis (not displayed) supports the model.

```
trend = time(cmort) - mean(time(cmort)) # center time
dcmort = resid(fit2 <- lm(cmort~trend, na.action=NULL)); fit2
  (Intercept) trend
  88.699   -1.625
u = ts.intersect(dM=dcmort, dM1=lag(dcmort,-1), dM2=lag(dcmort,-2),
  T1=lag(temp,-1), P=part, P4=lag(part,-4))
# lm(dM~, data=u, na.action=NULL) # and then analyze residuals ... or
sarima(u[,1], 0,0,0, xreg=u[,2:6]) # get residual analysis as a byproduct
Coefficients:
  intercept    dM1      dM2       T1        P       P4
  5.9884  0.3164  0.2989 -0.1826  0.1107  0.0495
  s.e.    2.6401  0.0370  0.0395  0.0309  0.0177  0.0195
  sigma^2 estimated as 25.42
```

We can now use Newton–Raphson and the Kalman filter to fit all the parameters simultaneously because the quick method has given us reasonable starting values. The results are close to the quick and dirty method:

	estimate	SE	
phi1	0.315	0.037	# $\hat{\phi}_1$
phi2	0.318	0.041	# $\hat{\phi}_2$
sigv	5.061	0.161	# $\hat{\sigma}_v$
T1	-0.119	0.031	# $\hat{\beta}_1$

```

P          0.119  0.018  #  $\hat{\beta}_2$ 
P4         0.067  0.019  #  $\hat{\beta}_3$ 
trend     -1.340  0.220  #  $\hat{\beta}_4$ 
constant   88.752  7.015  #  $\hat{\alpha}$ 

```

The R code for the complete analysis is as follows:

```

trend = time(cmort) - mean(time(cmort))    # center time
const = time(cmort)/time(cmort)              # appropriate time series of ls
ded   = ts.intersect(M=cmort, T1=lag(tempy,-1), P=part, P4=lag(part,-4),
                     trend, const)
y     = ded[,1]
input = ded[,2:6]
num   = length(y)
A     = array(c(1,0), dim = c(1,2,num))
# Function to Calculate Likelihood
Linn=function(para){
  phi1=para[1]; phi2=para[2]; cR=para[3]; b1=para[4];
  b2=para[5]; b3=para[6]; b4=para[7]; alf=para[8];
  mu0  = matrix(c(0,0), 2, 1)
  Sigma0 = diag(100, 2)
  Phi   = matrix(c(phi1, phi2, 1, 0), 2)
  Theta = matrix(c(phi1, phi2), 2)
  Ups   = matrix(c(b1, 0, b2, 0, b3, 0, 0, 0, 0, 0, 0, 0), 2, 5)
  Gam   = matrix(c(0, 0, 0, b4, alf), 1, 5); cQ = cR; S = cR^2
  kf = Kfilter2(num, y, A, mu0, Sigma0, Phi, Ups, Gam, Theta, cQ, cR, S,
                input)
  return(kf$like) }
# Estimation
init.par = c(phi1=.3, phi2=.3, cR=5, b1=-.2, b2=.1, b3=.05, b4=-1.6,
            alf=mean(cmort))           # initial parameters
L = c( 0, 0, 1, -1, 0, 0, -2, 70)  # lower bound on parameters
U = c(.5, .5, 10, 0, .5, 0, 90)    # upper bound - used in optim
est = optim(init.par, Linn, NULL, method='L-BFGS-B', lower=L, upper=U,
            hessian=TRUE, control=list(trace=1, REPORT=1, factr=10^8))
SE = sqrt(diag(solve(est$hessian)))
round(cbind(estimate=est$par, SE), 3)  # results

```

The residual analysis involves running the Kalman filter with the final estimated values and then investigating the resulting innovations. We do not display the results, but the analysis supports the model.

```

# Residual Analysis (not shown)
phi1 = est$par[1]; phi2 = est$par[2]
cR = est$par[3]; b1 = est$par[4]
b2 = est$par[5]; b3 = est$par[6]
b4 = est$par[7]; alf = est$par[8]
mu0  = matrix(c(0,0), 2, 1); Sigma0 = diag(100, 2)
Phi   = matrix(c(phi1, phi2, 1, 0), 2)
Theta = matrix(c(phi1, phi2), 2)
Ups   = matrix(c(b1, 0, b2, 0, b3, 0, 0, 0, 0, 0, 0, 0), 2, 5)
Gam   = matrix(c(0, 0, 0, b4, alf), 1, 5)
cQ   = cR
S    = cR^2
kf   = Kfilter2(num, y, A, mu0, Sigma0, Phi, Ups, Gam, Theta, cQ, cR, S,
                input)
res  = ts(as.vector(kf$innov), start=start(cmort), freq=frequency(cmort))
sarima(res, 0,0,0, no.constant=TRUE)  # gives a full residual analysis

```

Finally, a similar and simpler analysis can be fit using a complete ARMAX model. In this case the model would be

$$M_t = \alpha + \phi_1 M_{t-1} + \phi_2 M_{t-2} + \beta_1 T_{t-1} + \beta_2 P_t + \beta_3 P_{t-4} + \beta_4 t + v_t \quad (6.113)$$

where $v_t \sim \text{iid } N(0, \sigma_v^2)$. This model is different from (6.112) in that the mortality process is not detrended, but trend appears as an exogenous variable. In this case, we may use `sarima` to easily perform the regression and get the residual analysis as a byproduct.

```
trend = time(cmort) - mean(time(cmort))
u     = ts.intersect(M=cmort, M1=lag(cmort,-1), M2=lag(cmort,-2),
                     T1=lag(temp, -1), P=part, P4=lag(part, -4), trend)
sarima(u[,1], 0,0,0, xreg=u[,2:7]) # could use lm, but it's more work
Coefficients:
intercept      M1      M2      T1      P      P4      trend
40.3838  0.315  0.2971 -0.1845  0.1113  0.0513 -0.5214
s.e.       4.5982  0.037  0.0394  0.0309  0.0177  0.0195  0.0956
sigma^2 estimated as 25.32
```

We note that the residuals look fine, and the model fit is similar to the fit of (6.112).

6.7 Bootstrapping State Space Models

Although in [Section 6.3](#) we discussed the fact that under general conditions (which we assume to hold in this section) the MLEs of the parameters of a DLM are consistent and asymptotically normal, time series data are often of short or moderate length. Several researchers have found evidence that samples must be fairly large before asymptotic results are applicable (Dent and Min, 1978; Ansley and Newbold, 1980). Moreover, as we discussed in [Example 3.36](#), problems occur if the parameters are near the boundary of the parameter space. In this section, we discuss an algorithm for bootstrapping state space models; this algorithm and its justification, including the non-Gaussian case, along with numerous examples, can be found in Stoffer and Wall (1991) and in Stoffer and Wall (2004). In view of [Section 6.6](#), anything we do or say here about DLMs applies equally to ARMAX models.

Using the DLM given by (6.95)–(6.97) and [Property 6.5](#), we write the *innovations form of the filter* as

$$\epsilon_t = y_t - A_t x_t^{t-1} - \Gamma u_t, \quad (6.114)$$

$$\Sigma_t = A_t P_t^{t-1} A_t' + R, \quad (6.115)$$

$$K_t = [\Phi P_t^{t-1} A_t' + \Theta S] \Sigma_t^{-1}, \quad (6.116)$$

$$x_{t+1}^t = \Phi x_t^{t-1} + \Upsilon u_{t+1} + K_t \epsilon_t, \quad (6.117)$$

$$P_{t+1}^t = \Phi P_t^{t-1} \Phi' + \Theta Q \Theta' - K_t \Sigma_t K_t'. \quad (6.118)$$

This form of the filter is just a rearrangement of the filter given in [Property 6.5](#).

In addition, we can rewrite the model to obtain its innovations form,

$$x_{t+1}^t = \Phi x_t^{t-1} + \Upsilon u_{t+1} + K_t \epsilon_t, \quad (6.119)$$

$$y_t = A_t x_t^{t-1} + \Gamma u_t + \epsilon_t. \quad (6.120)$$

This form of the model is a rewriting of (6.114) and (6.117), and it accommodates the bootstrapping algorithm.

As discussed in [Example 6.5](#), although the innovations ϵ_t are uncorrelated, initially, Σ_t can be vastly different for different time points t . Thus, in a resampling procedure, we can either ignore the first few values of ϵ_t until Σ_t stabilizes or we can work with the *standardized innovations*

$$e_t = \Sigma_t^{-1/2} \epsilon_t, \quad (6.121)$$

so we are guaranteed these innovations have, at least, the same first two moments. In (6.121), $\Sigma_t^{1/2}$ denotes the unique square root matrix of Σ_t defined by $\Sigma_t^{1/2} \Sigma_t^{1/2} = \Sigma_t$. In what follows, we base the bootstrap procedure on the standardized innovations, but we stress the fact that, even in this case, ignoring startup values might be necessary, as noted by Stoffer & Wall (1991).

The model coefficients and the correlation structure of the model are uniquely parameterized by a $k \times 1$ parameter vector Θ_0 ; that is, $\Phi = \Phi(\Theta_0)$, $\Upsilon = \Upsilon(\Theta_0)$, $Q = Q(\Theta_0)$, $A_t = A_t(\Theta_0)$, $\Gamma = \Gamma(\Theta_0)$, and $R = R(\Theta_0)$. Recall the innovations form of the Gaussian likelihood (ignoring a constant) is

$$\begin{aligned} -2 \ln L_Y(\Theta) &= \sum_{t=1}^n [\ln |\Sigma_t(\Theta)| + \epsilon_t(\Theta)' \Sigma_t(\Theta)^{-1} \epsilon_t(\Theta)] \\ &= \sum_{t=1}^n [\ln |\Sigma_t(\Theta)| + e_t(\Theta)' e_t(\Theta)]. \end{aligned} \quad (6.122)$$

We stress the fact that it is not necessary for the model to be Gaussian to consider (6.122) as the criterion function to be used for parameter estimation.

Let $\hat{\Theta}$ denote the MLE of Θ_0 , that is, $\hat{\Theta} = \operatorname{argmax}_{\Theta} L_Y(\Theta)$, obtained by the methods discussed in [Section 6.3](#). Let $\epsilon_t(\hat{\Theta})$ and $\Sigma_t(\hat{\Theta})$ be the innovation values obtained by running the filter, (6.114)–(6.118), under $\hat{\Theta}$. Once this has been done, the nonparametric^{6.2} bootstrap procedure is accomplished by the following steps.

- (i) Construct the standardized innovations

$$e_t(\hat{\Theta}) = \Sigma_t^{-1/2}(\hat{\Theta}) \epsilon_t(\hat{\Theta}).$$

- (ii) Sample, with replacement, n times from the set $\{e_1(\hat{\Theta}), \dots, e_n(\hat{\Theta})\}$ to obtain $\{e_1^*(\hat{\Theta}), \dots, e_n^*(\hat{\Theta})\}$, a bootstrap sample of standardized innovations.
- (iii) Construct a bootstrap data set $\{y_1^*, \dots, y_n^*\}$ as follows. Define the $(p+q) \times 1$ vector $\xi_t = (x_{t+1}', y_t')'$. Stacking (6.119) and (6.120) results in a vector first-order equation for ξ_t given by

^{6.2} Nonparametric refers to the fact that we use the empirical distribution of the innovations rather than assuming they have a parametric form.

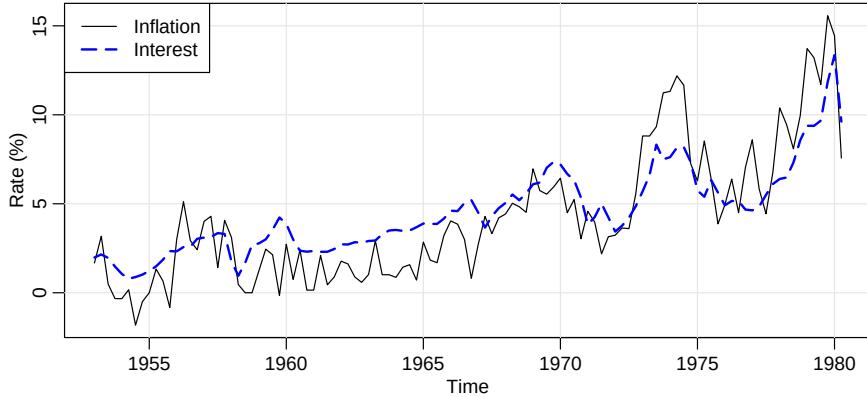


Fig. 6.9. Quarterly interest rate for Treasury bills (dashed line) and quarterly inflation rate (solid line) in the Consumer Price Index.

$$\xi_t = F_t \xi_{t-1} + G u_t + H_t e_t, \quad (6.123)$$

where

$$F_t = \begin{bmatrix} \Phi & 0 \\ A_t & 0 \end{bmatrix}, \quad G = \begin{bmatrix} \gamma \\ \Gamma \end{bmatrix}, \quad H_t = \begin{bmatrix} K_t \Sigma_t^{1/2} \\ \Sigma_t^{1/2} \end{bmatrix}.$$

Thus, to construct the bootstrap data set, solve (6.123) using $e_t^*(\hat{\Theta})$ in place of e_t . The exogenous variables u_t and the initial conditions of the Kalman filter remain fixed at their given values, and the parameter vector is held fixed at $\hat{\Theta}$.

- (iv) Using the bootstrap data set $y_{1:n}^*$, construct a likelihood, $L_{Y^*}(\Theta)$, and obtain the MLE of Θ , say, $\hat{\Theta}^*$.
- (v) Repeat steps 2 through 4, a large number, B , of times, obtaining a bootstrapped set of parameter estimates $\{\hat{\Theta}_b^*; b = 1, \dots, B\}$. The finite sample distribution of $\hat{\Theta} - \Theta_0$ may be approximated by the distribution of $\hat{\Theta}_b^* - \hat{\Theta}$, $b = 1, \dots, B$.

In the next example, we discuss the case of a linear regression model, but where the regression coefficients are stochastic and allowed to vary with time. The state space model provides a convenient setting for the analysis of such models.

Example 6.13 Stochastic Regression

Figure 6.9 shows the quarterly inflation rate (solid line), y_t , in the Consumer Price Index and the quarterly interest rate recorded for Treasury bills (dashed line), z_t , from the first quarter of 1953 through the second quarter of 1980, $n = 110$ observations. These data are taken from Newbold and Bos (1985).

In this example, we consider one analysis that was discussed in Newbold and Bos (1985, pp. 61–73), that focused on the first 50 observations and where quarterly inflation was modeled as being stochastically related to quarterly interest rate,

$$y_t = \alpha + \beta_t z_t + v_t,$$

Table 6.2. Comparison of Standard Errors

Parameter	MLE	Asymptotic Standard Error	Bootstrap Standard Error
ϕ	.865	.223	.463
α	-.686	.487	.557
b	.788	.226	.821
σ_w	.115	.107	.216
σ_v	1.135	.147	.340

where α is a fixed constant, β_t is a stochastic regression coefficient, and v_t is white noise with variance σ_v^2 . The stochastic regression term, which comprises the state variable, is specified by a first-order autoregression,

$$(\beta_t - b) = \phi(\beta_{t-1} - b) + w_t,$$

where b is a constant, and w_t is white noise with variance σ_w^2 . The noise processes, v_t and w_t , are assumed to be uncorrelated.

Using the notation of the state-space model (6.95) and (6.96), we have in the state equation, $x_t = \beta_t$, $\Phi = \phi$, $u_t \equiv 1$, $\Upsilon = (1 - \phi)b$, $Q = \sigma_w^2$, and in the observation equation, $A_t = z_t$, $\Gamma = \alpha$, $R = \sigma_v^2$, and $S = 0$. The parameter vector is $\Theta = (\phi, \alpha, b, \sigma_w, \sigma_v)'$. The results of the Newton–Raphson estimation procedure are listed in Table 6.2. Also shown in the Table 6.2 are the corresponding standard errors obtained from $B = 500$ runs of the bootstrap. These standard errors are simply the standard deviations of the bootstrapped estimates, that is, the square root of $\sum_{b=1}^B (\hat{\Theta}_{ib}^* - \hat{\Theta}_i)^2 / (B - 1)$, where $\hat{\Theta}_i$ represents the MLE of the i th parameter, Θ_i , for $i = 1, \dots, 5$,

The asymptotic standard errors listed in Table 6.2 are typically much smaller than those obtained from the bootstrap. For most of the cases, the bootstrapped standard errors are at least 50% larger than the corresponding asymptotic value. Also, asymptotic theory prescribes the use of normal theory when dealing with the parameter estimates. The bootstrap, however, allows us to investigate the small sample distribution of the estimators and, hence, provides more insight into the data analysis.

For example, Figure 6.10 shows the bootstrap distribution of the estimator of ϕ in the upper left-hand corner. This distribution is highly skewed with values concentrated around .8, but with a long tail to the left. Some quantiles are $-.09$ (5%), $.11$ (10%), $.34$ (25%), $.73$ (50%), $.86$ (75%), $.96$ (90%), $.98$ (95%), and they can be used to obtain confidence intervals. For example, a 90% confidence interval for ϕ would be approximated by $(-.09, .96)$. This interval is ridiculously wide and includes 0 as a plausible value of ϕ ; we will interpret this after we discuss the results of the estimation of σ_w .

Figure 6.10 shows the bootstrap distribution of $\hat{\sigma}_w$ in the lower right-hand corner. The distribution is concentrated at two locations, one at approximately $\hat{\sigma}_w = .25$ (which is the median of the distribution of values away from 0) and

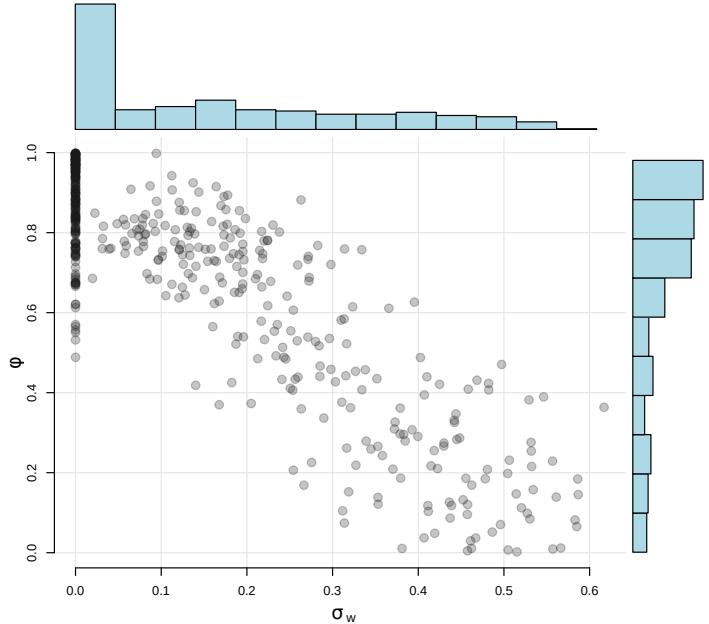


Fig. 6.10. Joint and marginal bootstrap distributions, $B = 500$, of $\hat{\phi}$ and $\hat{\sigma}_w$. Only the values corresponding to $\hat{\phi}^* \geq 0$ are shown.

the other at $\hat{\sigma}_w = 0$. The cases in which $\hat{\sigma}_w \approx 0$ correspond to deterministic state dynamics. When $\sigma_w = 0$ and $|\phi| < 1$, then $\beta_t \approx b$ for large t , so the approximately 25% of the cases in which $\hat{\sigma}_w \approx 0$ suggest a fixed state, or constant coefficient model. The cases in which $\hat{\sigma}_w$ is away from zero would suggest a truly stochastic regression parameter. To investigate this matter further, the off-diagonals of Figure 6.10 show the joint bootstrapped estimates, $(\hat{\phi}, \hat{\sigma}_w)$, for positive values of $\hat{\phi}^*$. The joint distribution suggests $\hat{\sigma}_w > 0$ corresponds to $\hat{\phi} \approx 0$. When $\phi = 0$, the state dynamics are given by $\beta_t = b + w_t$. If, in addition, σ_w is small relative to b , the system is nearly deterministic; that is, $\beta_t \approx b$. Considering these results, the bootstrap analysis leads us to conclude the dynamics of the data are best described in terms of a fixed regression effect.

The following R code was used for this example. We note that the first few lines of the code set the relative tolerance for determining convergence of the numerical optimization and the number of bootstrap replications. *Using the current settings may result in a long run time of the algorithm* and we suggest the tolerance and the number of bootstrap replicates be decreased on slower machines or for demonstration purposes. For example, setting `tol=.001` and `nboot=200` yields reasonable results. In this example, we fixed the first three values of the data for the resampling scheme.

```
library(plyr)                      # used for displaying progress
tol = sqrt(.Machine$double.eps)    # determines convergence of optimizer
```

```

nboot = 500                      # number of bootstrap replicates
y     = window(qinfl, c(1953,1), c(1965,2)) # inflation
z     = window(qintr, c(1953,1), c(1965,2)) # interest
num   = length(y)
A     = array(z, dim=c(1,1,num))
input = matrix(1,num,1)
# Function to Calculate Likelihood
Linn = function(para, y.data){ # pass data also
  phi = para[1]; alpha = para[2]
  b   = para[3]; Ups  = (1-phi)*b
  cQ  = para[4]; cR   = para[5]
  kf  = Kfilter2(num,y.data,A,mu0,Sigma0,phi,Ups,alpha,1,cQ,cR,0,input)
  return(kf$like)    }
# Parameter Estimation
mu0 = 1; Sigma0 = .01
init.par = c(phi=.84, alpha=-.77, b=.85, cQ=.12, cR=1.1) # initial values
est = optim(init.par, Linn, NULL, y.data=y, method="BFGS", hessian=TRUE,
            control=list(trace=1, REPORT=1, reltol=tol))
SE  = sqrt(diag(solve(est$hessian)))
phi = est$par[1]; alpha = est$par[2]
b   = est$par[3]; Ups  = (1-phi)*b
cQ  = est$par[4]; cR   = est$par[5]
round(cbind(estimate=est$par, SE), 3)
      estimate      SE
phi      0.865 0.223
alpha    -0.686 0.487
b        0.788 0.226
cQ       0.115 0.107
cR       1.135 0.147
# BEGIN BOOTSTRAP
# Run the filter at the estimates
kf = Kfilter2(num,y,A,mu0,Sigma0,phi,Ups,alpha,1,cQ,cR,0,input)
# Pull out necessary values from the filter and initialize
xp   = kf$xp
innov = kf$innov
sig   = kf$sig
K     = kf$K
e     = innov/sqrt(sig)
e.star = e                      # initialize values
y.star = y
xp.star = xp
k     = 4:50                     # hold first 3 observations fixed
para.star = matrix(0, nboot, 5) # to store estimates
init.par = c(.84, -.77, .85, .12, 1.1)
pr <- progress_text()           # displays progress
pr$init(nboot)
for (i in 1:nboot){
  pr$step()
  e.star[k] = sample(e[k], replace=TRUE)
  for (j in k){ xp.star[j] = phi*xp.star[j-1] +
    Ups+K[j]*sqrt(sig[j])*e.star[j] }
  y.star[k] = z[k]*xp.star[k] + alpha + sqrt(sig[k])*e.star[k]
  est.star = optim(init.par, Linn, NULL, y.data=y.star, method="BFGS",
                  control=list(reltol=tol))
  para.star[i,] = cbind(est.star$par[1], est.star$par[2], est.star$par[3],
                        abs(est.star$par[4]), abs(est.star$par[5]))  }

```

```

# Some summary statistics
rmse = rep(NA,5)                      # SEs from the bootstrap
for(i in 1:5){rmse[i]=sqrt(sum((para.star[,i]-est$par[i])^2)/nboot)
  cat(i, rmse[i],"\n") }
# Plot phi and sigw
phi = para.star[,1]
sigw = abs(para.star[,4])
phi = ifelse(phi<0, NA, phi)    # any phi < 0 not plotted
library(psych)                      # load psych package for scatter.hist
scatter.hist(sigw, phi, ylab=expression(phi), xlab=expression(sigma[~w]),
             smooth=FALSE, correl=FALSE, density=FALSE, ellipse=FALSE,
             title='', pch=19, col=gray(.1,alpha=.33),
             panel.first=grid(lty=2), cex.lab=1.2)

```

6.8 Smoothing Splines and the Kalman Smoother

There is a connection between smoothing splines, e.g., Eubank (1993), Green (1993), or Wahba (1990) and state space models. The basic idea of smoothing splines (recall Example 2.14) in discrete time is we suppose that data y_t are generated by $y_t = \mu_t + \epsilon_t$ for $t = 1, \dots, n$, where μ_t is a smooth function of t , and ϵ_t is white noise. In cubic smoothing with knots at the time points t , μ_t is estimated by minimizing

$$\sum_{t=1}^n [y_t - \mu_t]^2 + \lambda \sum_{t=1}^n (\nabla^2 \mu_t)^2 \quad (6.124)$$

with respect to μ_t , where $\lambda > 0$ is a smoothing parameter. The parameter λ controls the degree of smoothness, with larger values yielding smoother estimates. For example, if $\lambda = 0$, then the minimizer is the data itself $\hat{\mu}_t = y_t$; consequently, the estimate will not be smooth. If $\lambda = \infty$, then the only way to minimize (6.124) is to choose the second term to be zero, i.e., $\nabla^2 \mu_t = 0$, in which case it is of the form $\mu_t = \alpha + \beta t$, and we are in the setting of linear regression.^{6.3} Hence, the choice of $\lambda > 0$ is seen as a trade-off between fitting a line that goes through all the data points and linear regression.

Now, consider the model given by

$$\nabla^2 \mu_t = w_t \quad \text{and} \quad y_t = \mu_t + v_t, \quad (6.125)$$

where w_t and v_t are independent white noise processes with $\text{var}(w_t) = \sigma_w^2$ and $\text{var}(v_t) = \sigma_v^2$. Rewrite (6.125) as

$$\begin{pmatrix} \mu_t \\ \mu_{t-1} \end{pmatrix} = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} \mu_{t-1} \\ \mu_{t-2} \end{pmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} w_t \quad \text{and} \quad y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{pmatrix} \mu_t \\ \mu_{t-1} \end{pmatrix} + v_t, \quad (6.126)$$

so that the state vector is $x_t = (\mu_t, \mu_{t-1})'$. It is clear then that (6.126) specifies a state space model.

^{6.3} That the unique general solution to $\nabla^2 \mu_t = 0$ is of the form $\mu_t = \alpha + \beta t$ follows from difference equation theory, e.g., see Mickens (1990).

Note that the model is similar to the local level model discussed in [Example 6.5](#). In particular, the state process could be written as $\mu_t = \mu_{t-1} + \eta_t$, where $\eta_t = \eta_{t-1} + w_t$. An example of such a trajectory can be seen in [Figure 6.11](#); note that the generated data in [Figure 6.11](#) look like the global temperature data in [Figure 1.2](#).

Next, we examine the problem of estimating the states, x_t , when the model parameters, $\theta = \{\sigma_w^2, \sigma_v^2\}$, are specified. For ease, we assume x_0 is fixed. Then using the notation surrounding equations [\(6.61\)](#)–[\(6.62\)](#), the goal is to find the MLE of $x_{1:n} = \{x_1, \dots, x_n\}$ given $y_{1:n} = \{y_1, \dots, y_n\}$; i.e., to maximize $\log p_\theta(x_{1:n} | y_{1:n})$ with respect to the states. Because of the Gaussianity, the maximum (or mode) of the distribution is when the states are estimated by x_t^n , the conditional means. These values are, of course, the smoothers obtained via [Property 6.2](#).

But $\log p_\theta(x_{1:n} | y_{1:n}) = \log p_\theta(x_{1:n}, y_{1:n}) - \log p_\theta(y_{1:n})$, so maximizing the complete data likelihood, $\log p_\theta(x_{1:n}, y_{1:n})$ with respect to $x_{1:n}$, is an equivalent problem. Writing [\(6.62\)](#) in the notation of [\(6.125\)](#), we have,

$$-2 \log p_\theta(x_{1:n}, y_{1:n}) \propto \sigma_w^{-2} \sum_{t=1}^n (\nabla^2 \mu_t)^2 + \sigma_v^{-2} \sum_{t=1}^n (y_t - \mu_t)^2, \quad (6.127)$$

where we have kept only the terms involving the states, μ_t . If we set $\lambda = \sigma_v^2 / \sigma_w^2$, we can write

$$-2 \log p_\theta(x_{1:n}, y_{1:n}) \propto \lambda \sum_{t=1}^n (\nabla^2 \mu_t)^2 + \sum_{t=1}^n (y_t - \mu_t)^2, \quad (6.128)$$

so that maximizing $\log p_\theta(x_{1:n}, y_{1:n})$ with respect to the states is equivalent to minimizing [\(6.128\)](#), which is the original problem stated in [\(6.124\)](#).

In the general state space setting, we would estimate σ_w^2 and σ_v^2 via maximum likelihood as described in [Section 6.3](#), and then obtain the smoothed state values by running [Property 6.2](#) with the estimated variances, say $\hat{\sigma}_w^2$ and $\hat{\sigma}_v^2$. In this case, the estimated value of the smoothing parameter would be given by $\hat{\lambda} = \hat{\sigma}_v^2 / \hat{\sigma}_w^2$.

Example 6.14 Smoothing Splines

In this example, we generated the signal, or state process, μ_t and observations y_t from the model [\(6.125\)](#) with $n = 50$, $\sigma_w = .1$ and $\sigma_v = 1$. The state is displayed in [Figure 6.11](#) as a thick solid line, and the observations are displayed as points. We then estimated σ_w and σ_v using Newton-Raphson techniques and obtained $\hat{\sigma}_w = .08$ and $\hat{\sigma}_v = .94$. We then used [Property 6.2](#) to generate the estimated smoothers, say, $\hat{\mu}_t^n$, and those values are displayed in [Figure 6.11](#) as a thick dashed line along with a corresponding 95% (pointwise) confidence band as thin dashed lines. Finally, we used the R function `smooth.spline` to fit a smoothing spline to the data based on the method of generalized cross-validation (gcv). The fitted spline is displayed in [Figure 6.11](#) as a thin solid line, which is close to $\hat{\mu}_t^n$.

The R code to reproduce [Figure 6.11](#) is given below.

```
set.seed(123)
num = 50
w = rnorm(num, 0, 1)
x = cumsum(cumsum(w))
```

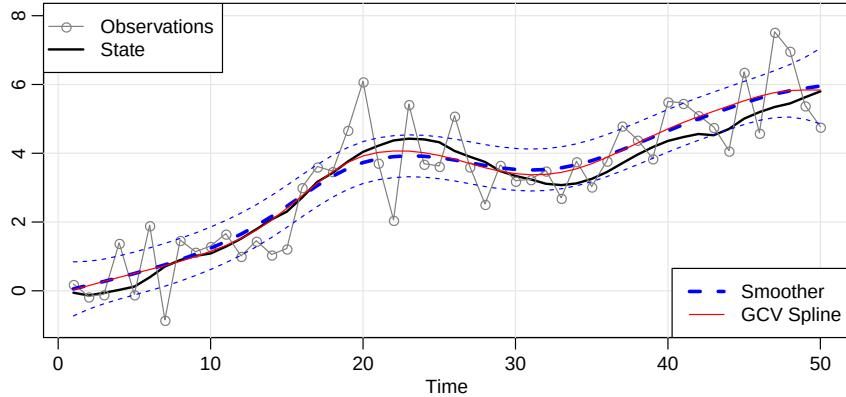


Fig. 6.11. Display for Example 6.14: Simulated state process, μ_t and observations y_t from the model (6.125) with $n = 50$, $\sigma_w = .1$ and $\sigma_v = 1$. Estimated smoother (dashed lines): $\hat{\mu}_t|n$ and corresponding 95% confidence band. GCV smoothing spline (thin solid line).

```

y = x + rnorm(num,0,1)
plot.ts(x, ylab="", lwd=2, ylim=c(-1,8))
lines(y, type='o', col=8)
## State Space ##
Phi = matrix(c(2,1,-1,0),2); A = matrix(c(1,0),1)
mu0 = matrix(0,2); Sigma0 = diag(1,2)
Linn = function(para){
  sigw = para[1]; sigv = para[2]
  cQ = diag(c(sigw,0))
  kf = Kfilter0(num, y, A, mu0, Sigma0, Phi, cQ, sigv)
  return(kf$like) }
## Estimation ##
init.par = c(.1, 1)
(est = optim(init.par, Linn, NULL, method="BFGS", hessian=TRUE,
control=list(trace=1,REPORT=1)))
SE = sqrt(diag(solve(est$hessian)))
# Summary of estimation
estimate = est$par; u = cbind(estimate, SE)
rownames(u) = c("sigw","sigv"); u
# Smooth
sigw = est$par[1]
cQ = diag(c(sigw,0))
sigv = est$par[2]
ks = Ksmooth0(num, y, A, mu0, Sigma0, Phi, cQ, sigv)
xsmoo = ts(ks$xs[,1,]); psmoo = ts(ks$Ps[,1,])
upp = xsmoo+2*sqrt(psmoo); low = xsmoo-2*sqrt(psmoo)
lines(xsmoo, col=4, lty=2, lwd=3)
lines(upp, col=4, lty=2); lines(low, col=4, lty=2)
lines(smooth.spline(y), lty=1, col=2)
legend("topleft", c("Observations", "State"), pch=c(1,-1), lty=1, lwd=c(1,2),
col=c(8,1))
legend("bottomright", c("Smoother", "GCV Spline"), lty=c(2,1), lwd=c(3,1),
col=c(4,2))

```

6.9 Hidden Markov Models and Switching Autoregression

In the introduction to this chapter, we mentioned that the state space model is characterized by two principles. First, there is a hidden state process, $\{x_t; t = 0, 1, \dots\}$, that is assumed to be Markovian. Second, the observations, $\{y_t; t = 1, 2, \dots\}$, are independent given the states. The principles were displayed in [Figure 6.1](#) and written in terms of densities in [\(6.28\)](#) and [\(6.29\)](#).

We have been focusing primarily on linear Gaussian state space models, but there is an entire area that has developed around the case where the states x_t are a discrete-valued Markov chain, and that will be the focus in this section. The basic idea is that the value of the state at time t specifies the distribution of the observation at time t . These models were developed in Goldfeld and Quandt (1973) and Lindgren (1978). Changes can also be modeled in the classical regression setting by allowing the value of the state to determine the design matrix, as in Quandt (1972). An early application to speech recognition was considered by Juang and Rabiner (1985). An application of the idea of switching to the tracking of multiple targets was considered in Bar-Shalom (1978), who obtained approximations to Kalman filtering in terms of weighted averages of the innovations. As another example, some authors (for example, Hamilton, 1989, or McCulloch and Tsay, 1993) have explored the possibility that the dynamics of a country's economy might be different during expansion than during contraction.

In the Markov chain approach, we declare the dynamics of the system at time t are generated by one of m possible regimes evolving according to a Markov chain over time. The case in which the particular regime is unknown to the observer comes under the heading of *hidden Markov models* (HMM), and the techniques related to analyzing these models are summarized in Rabiner and Juang (1986). Although the model satisfies the conditions for being a state space model, HMMs were developed in parallel. If the state process is discrete-valued, one typically uses the term “hidden Markov model” and if the state process is continuous-valued, one uses the term “state space model” or one of its variants. Texts that cover the theory and methods in whole or in part are Cappé, Moulines, & Rydén (2009) and Douc, Moulines, & Stoffer (2014). A recent introductory text that uses R is Zucchini & MacDonald (2009).

Here, we assume the states, x_t , are a Markov chain taking values in a finite state space $\{1, \dots, m\}$, with stationary distribution

$$\pi_j = \Pr(x_t = j), \quad (6.129)$$

and stationary transition probabilities

$$\pi_{ij} = \Pr(x_{t+1} = j \mid x_t = i), \quad (6.130)$$

for $t = 0, 1, 2, \dots$, and $i, j = 1, \dots, m$. Since the second component of the model is that the observations are conditionally independent, we need to specify the distributions, and we denote them by

$$p_j(y_t) = p(y_t \mid x_t = j). \quad (6.131)$$

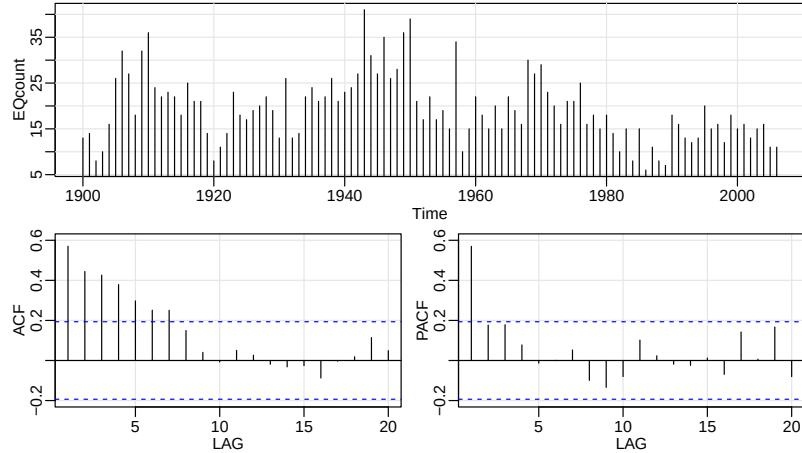


Fig. 6.12. Top: Series of annual counts of major earthquakes (magnitude 7 and above) in the world between 1900–2006. Bottom: Sample ACF and PACF of the counts.

Example 6.15 Poisson HMM – Number of Major Earthquakes

Consider the time series of annual counts of major earthquakes displayed in Figure 6.12 that were discussed in Zucchini & MacDonald (2009). A natural model for unbounded count data is a Poisson distribution, in which case the mean and variance are equal. However, the sample mean and variance of the data are $\bar{x} = 19.4$ and $s^2 = 51.6$, so this model is clearly inappropriate. It would be possible to take into account the overdispersion by using other distributions for counts such as the negative binomial distribution or a mixture of Poisson distributions. This approach, however, ignores the sample ACF and PACF displayed Figure 6.12, which indicate the observations are serially correlated, and further suggest an AR(1)-type correlation structure.

A simple and convenient way to capture both the marginal distribution and the serial dependence is to consider a Poisson-HMM model. Let y_t denote the number of major earthquakes in year t , and consider the state, or latent variable, x_t to be a stationary two-state Markov chain taking values in $\{1, 2\}$. Using the notation in (6.129) and (6.130), we have $\pi_{12} = 1 - \pi_{11}$ and $\pi_{21} = 1 - \pi_{22}$. The stationary distribution of this Markov chain is given by^{6.4}

$$\pi_1 = \frac{\pi_{21}}{\pi_{12} + \pi_{21}}, \quad \text{and} \quad \pi_2 = \frac{\pi_{12}}{\pi_{12} + \pi_{21}}.$$

For $j \in \{1, 2\}$, denote $\lambda_j > 0$ as the parameter of a Poisson distribution,

$$p_j(y) = \frac{\lambda_j^y e^{-\lambda_j}}{y!}, \quad y = 0, 1, \dots.$$

^{6.4} The stationary distribution must satisfy $\pi_j = \sum_i \pi_i \pi_{ij}$.

Since the states are stationary, the marginal distribution of y_t is stationary and a mixture of Poissons,

$$p_{\Theta}(y_t) = \pi_1 p_1(y_t) + \pi_2 p_2(y_t)$$

with $\Theta = \{\lambda_1, \lambda_2\}$. The mean of the stationary distribution is

$$E(y_t) = \pi_1 \lambda_1 + \pi_2 \lambda_2 \quad (6.132)$$

and the variance^{6.5} is

$$\text{var}(y_t) = E(y_t) + \pi_1 \pi_2 (\lambda_2 - \lambda_1)^2 \geq E(y_t), \quad (6.133)$$

implying that the two-state Poisson HMM is overdispersed. Similar calculations (see [Problem 6.21](#)) show that the autocovariance function of y_t is given by

$$\gamma_y(h) = \sum_{i=1}^2 \sum_{j=1}^2 \pi_i (\pi_{ij}^h - \pi_j) \lambda_i \lambda_j = \pi_1 \pi_2 (\lambda_2 - \lambda_1)^2 (1 - \pi_{12} - \pi_{21})^h. \quad (6.134)$$

Thus, a two-state Poisson-HMM has an exponentially decaying autocorrelation function, and this is consistent with the sample ACF seen in [Figure 6.12](#). It is worthwhile to note that if we increase the number of states, more complex dependence structures may be obtained.

As in the linear Gaussian case, we need filters and smoothers of the state in their own right, and additionally for estimation and prediction. We then write

$$\pi_j(t | s) = \Pr(x_t = j | y_{1:s}). \quad (6.135)$$

Straight forward calculations (see [Problem 6.22](#)) give the filter equations as:

Property 6.7 HMM Filter

For $t = 1, \dots, n$,

$$\pi_j(t | t-1) = \sum_{i=1}^m \pi_i(t-1 | t-1) \pi_{ij}, \quad (6.136)$$

$$\pi_j(t | t) = \frac{\pi_j(t) p_j(y_t)}{\sum_{i=1}^m \pi_i(t) p_i(y_t)}, \quad (6.137)$$

with initial condition $\pi_j(1 | 0) = \pi_j$.

Let Θ denote the parameters of interest. Given data $y_{1:n}$, the likelihood is given by

$$L_Y(\Theta) = \prod_{t=1}^n p_{\Theta}(y_t | y_{1:t-1}).$$

But, by the conditional independence,

^{6.5} Recall $\text{var}(U) = E[\text{var}(U | V)] + \text{var}[E(U | V)]$.

$$\begin{aligned} p_{\Theta}(y_t \mid y_{1:t-1}) &= \sum_{j=1}^m \Pr(x_t = j \mid y_{1:t-1}) p_{\Theta}(y_j \mid x_t = j, y_{1:t-1}) \\ &= \sum_{j=1}^m \pi_j(t-1) p_j(y_t). \end{aligned}$$

Consequently,

$$\ln L_Y(\Theta) = \sum_{t=1}^n \ln \left(\sum_{j=1}^m \pi_j(t-1) p_j(y_t) \right). \quad (6.138)$$

Maximum likelihood can then proceed as in the linear Gaussian case discussed in [Section 6.3](#).

In addition, the Baum-Welch (or EM) algorithm discussed in [Section 6.3](#) applies here as well. First, the general complete data likelihood still has the form of (6.61), that is,

$$\ln p_{\Theta}(x_{0:n}, y_{1:n}) = \ln p_{\Theta}(x_0) + \sum_{t=1}^n \ln p_{\Theta}(x_t \mid x_{t-1}) + \sum_{t=1}^n \ln p_{\Theta}(y_t \mid x_t).$$

It is more useful to define $I_j(t) = 1$ if $x_t = j$ and 0 otherwise, and $I_{ij}(t) = 1$ if $(x_{t-1}, x_t) = (i, j)$ and 0 otherwise, for $i, j = 1, \dots, m$. Recall $\Pr[I_j(t) = 1] = \pi_j$ and $\Pr[I_{ij}(t) = 1] = \pi_{ij} \pi_i$. Then the complete data likelihood can be written as (we drop Θ from some of the notation for convenience)

$$\begin{aligned} \ln p_{\Theta}(x_{0:n}, y_{1:n}) &= \sum_{j=1}^m I_j(0) \ln \pi_j + \sum_{t=1}^n \sum_{i=1}^m \sum_{j=1}^m I_{ij}(t) \ln \pi_{ij}(t) \\ &\quad + \sum_{t=1}^n \sum_{j=1}^m I_j(t) \ln p_j(y_t), \end{aligned} \quad (6.139)$$

and, as before, we need to maximize $Q(\Theta \mid \Theta') = E[\ln p_{\Theta}(x_{0:n}, y_{1:n}) \mid y_{1:n}, \Theta']$. In this case, it should be clear that in addition to the filter, (6.137), we will need

$$\pi_j(t \mid n) = E(I_j(t) \mid y_{1:n}) = \Pr(x_t = j \mid y_{1:n}) \quad (6.140)$$

for the first and third terms, and

$$\pi_{ij}(t \mid n) = E(I_{ij}(t) \mid y_{1:n}) = \Pr(x_t = i, x_{t+1} = j \mid y_{1:n}). \quad (6.141)$$

for the second term. In the evaluation of the second term, as will be seen, we must also evaluate

$$\varphi_j(t) = p(y_{t+1:n} \mid x_t = j). \quad (6.142)$$

Property 6.8 HMM Smoother

For $t = n - 1, \dots, 0$,

$$\pi_j(t | n) = \frac{\pi_j(t | t)\varphi_j(t)}{\sum_{j=1}^m \pi_j(t | t)\varphi_j(t)}, \quad (6.143)$$

$$\pi_{ij}(t | n) = \pi_i(t | n)\pi_{ij}\text{p}_j(y_{t+1})\varphi_j(t+1)/\varphi_i(t), \quad (6.144)$$

$$\varphi_i(t) = \sum_{j=1}^m \pi_{ij}\text{p}_j(y_{t+1})\varphi_j(t+1), \quad (6.145)$$

where $\varphi_j(n) = 1$ for $j = 1, \dots, m$.

Proof: We leave the proof of (6.143) to the reader; see [Problem 6.22](#). To verify (6.145), note that

$$\begin{aligned} \varphi_i(t) &= \sum_{j=1}^m \text{p}(y_{t+1:n}, x_{t+1} = j | x_t = i) \\ &= \sum_{j=1}^m \Pr(x_{t+1} = j | x_t = i) \text{p}(y_{t+1} | x_{t+1} = j) \text{p}(y_{t+2:n} | x_{t+1} = j) \\ &= \sum_{j=1}^m \pi_{ij} \text{p}_j(y_{t+1})\varphi_j(t+1). \end{aligned}$$

To verify (6.144), we have

$$\begin{aligned} \pi_{ij}(t | n) &\propto \Pr(x_t = i, x_{t+1} = j, y_{t+1}, y_{t+2:n} | y_{1:t}) \\ &= \Pr(x_t = i | y_{1:t}) \Pr(x_{t+1} = j | x_t = i) \\ &\quad \times \text{p}(y_{t+1} | x_{t+1} = j) \text{p}(y_{t+2:n} | x_{t+1} = j) \\ &= \pi_i(t | t) \pi_{ij} \text{p}_j(y_{t+1}) \varphi_j(t+1). \end{aligned}$$

Finally, to find the constant of proportionality, say C_t , if we sum over j on both sides we get, $\sum_{j=1}^m \pi_{ij}(t | n) = \pi_i(t | n)$ and $\sum_{j=1}^m \pi_{ij} \text{p}_j(y_{t+1}) \varphi_j(t+1) = \varphi_i(t)$. This means that $\pi_i(t | n) = C_t \pi_i(t | t) \varphi_i(t)$, and (6.144) follows. \square

For the Baum-Welch (or EM) algorithm, given the current value of the parameters, say Θ' , run the filter [Property 6.7](#) and smoother [Property 6.8](#), and then, as is evident from (6.139), update the first two estimates as

$$\hat{\pi}_j = \pi'_j(0 | n) \quad \text{and} \quad \hat{\pi}_{ij} = \frac{\sum_{t=1}^n \pi'_{ij}(t | n)}{\sum_{t=1}^n \sum_{k=1}^m \pi'_{ik}(t | n)}. \quad (6.146)$$

Of course, the prime indicates that values have been obtain under Θ' and the hat denotes the update. Although not the MLE, it has been suggested by Lindgren (1978) that a natural estimate of the stationary distribution of the chain would be

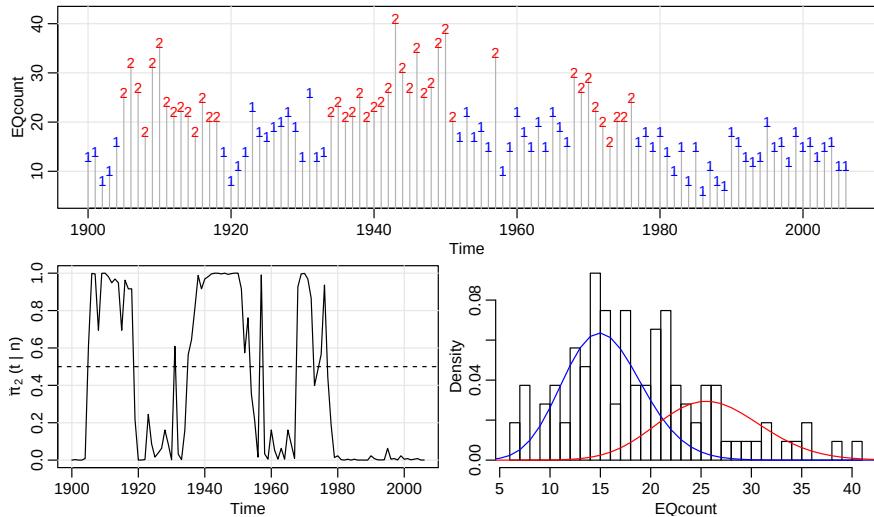


Fig. 6.13. Top: Earthquake count data and estimated states. Bottom left: Smoothing probabilities. Bottom right: Histogram of the data with the two estimated Poisson densities superimposed (solid lines).

$$\hat{\pi}_j = n^{-1} \sum_{t=1}^n \pi'_j(t|n),$$

rather than the value given in (6.146). Finally, the third term in (6.139) will require knowing the distribution of $p_j(y_t)$, and this will depend on the particular model. We will discuss the Poisson distribution in Example 6.15 and the normal distribution in Example 6.17

Example 6.16 Poisson HMM – Number of Major Earthquakes (cont)

To run the EM algorithm in this case, we still need to maximize the conditional expectation of the third term of (6.139). The conditional expectation of the third term at the current parameter value Θ' is

$$\sum_{t=1}^n \sum_{j=1}^m \pi'_j(t|t-1) \ln p_j(y_t),$$

where

$$\log p_j(y_t) \propto y_t \log \lambda_j - \lambda_j.$$

Consequently, maximization with respect to λ_j yields

$$\hat{\lambda}_j = \frac{\sum_{t=1}^n \pi'_j(t|n) y_t}{\sum_{t=1}^n \pi'_j(t|n)}, \quad j = 1, \dots, m.$$

We fit the model to the time series of earthquake counts using the R package `depmixS4`. The package, which uses the EM algorithm, does not provide standard errors, so we obtained them by a parametric bootstrap procedure; see Remillard (2011) for justification. The MLEs of the intensities, along with their standard errors, were $(\hat{\lambda}_1, \hat{\lambda}_2) = (15.4_{(.7)}, 26.0_{(1.1)})$. The MLE of the transition matrix was $[\hat{\pi}_{11}, \hat{\pi}_{12}, \hat{\pi}_{21}, \hat{\pi}_{22}] = [.93_{(.04)}, .07_{(.04)}, .12_{(.09)}, .88_{(.09)}]$. Figure 6.13 displays the counts, the estimated state (displayed as points) and the smoothing distribution for the earthquakes data, modeled as a two-state Poisson HMM model with parameters fitted using the MLEs. Finally, a histogram of the data is displayed along with the two estimated Poisson densities superimposed as solid lines.

The R code for this example is as follows.

```
library(depmixS4)
model <- depmix(EQcount ~1, nstates=2, data=data.frame(EQcount),
                 family=poisson())
set.seed(90210)
summary(fm <- fit(model)) # estimation results
##-- Get Parameters --##
u = as.vector(getpars(fm)) # ensure state 1 has smaller lambda
if (u[7] <= u[8]) { para.mle = c(u[3:6], exp(u[7]), exp(u[8]))}
else { para.mle = c(u[6:3], exp(u[8]), exp(u[7]))}
mtrans = matrix(para.mle[1:4], byrow=TRUE, nrow=2)
lams = para.mle[5:6]
pi1 = mtrans[2,1]/(2 - mtrans[1,1] - mtrans[2,2]); pi2 = 1-pi1
##-- Graphics --##
layout(matrix(c(1,2,1,3), 2))
par(mar = c(3,3,1,1), mgp = c(1.6,.6,0))
# data and states
plot(EQcount, main="", ylab='EQcount', type='h', col=gray(.7))
text(EQcount, col=6*posterior(fm)[,1]-2, labels=posterior(fm)[,1], cex=.9)
# prob of state 2
plot(ts(posterior(fm)[,3], start=1900), ylab =
      expression(hat(pi)[~2]*'(t|n'))); abline(h=.5, lty=2)
# histogram
hist(EQcount, breaks=30, prob=TRUE, main="")
xvals = seq(1,45)
u1 = pi1*dpois(xvals, lams[1])
u2 = pi2*dpois(xvals, lams[2])
lines(xvals, u1, col=4); lines(xvals, u2, col=2)
##-- Bootstrap --##
# function to generate data
pois.HMM.generate_sample = function(n,m,lambda,Mtrans,StatDist=NULL){
  # n = data length, m = number of states, Mtrans = transition matrix,
  # StatDist = stationary distn
  if(is.null(StatDist)) StatDist = solve(t(diag(m)-Mtrans +1),rep(1,m))
  mvect = 1:m
  state = numeric(n)
  state[1] = sample(mvect ,1, prob=StatDist)
  for (i in 2:n)
    state[i] = sample(mvect ,1,prob=Mtrans[state[i-1] ,])
  y = rpois(n,lambda=lambda[state ])
  list(y= y, state= state) }
```

start it up

```
set.seed(10101101)
```

```

nboot = 100
nobs = length(EQcount)
para.star = matrix(NA, nrow=nboot, ncol = 6)
for (j in 1:nboot){
  x.star = pois.HMM.generate_sample(n=nobs, m=2, lambda=lams, Mtrans=mtrans)$y
  model <- depmix(x.star ~1, nstates=2, data=data.frame(x.star),
    family=poisson())
  u = as.vector(getpars(fit(model, verbose=0)))
  # make sure state 1 is the one with the smaller intensity parameter
  if (u[7] <= u[8]) { para.star[j,] = c(u[3:6], exp(u[7]), exp(u[8])) }
  else { para.star[j,] = c(u[6:3], exp(u[8]), exp(u[7])) }           }
# bootstrapped std errors
SE = sqrt(apply(para.star,2,var) +
  (apply(para.star,2,mean)-para.mle)^2)[c(1,4:6)]
names(SE)=c('seM11/M12', 'seM21/M22', 'seLam1', 'seLam2'); SE

```

Next, we present an example using a mixture of normal distributions.

Example 6.17 Normal HMM – S&P500 Weekly Returns

Estimation in the Gaussian case is similar to the Poisson case given in Example 6.16, except that now, $p_j(y_t)$ is the normal density; i.e., $(y_t | x_t = j) \sim N(\mu_j, \sigma_j^2)$ for $j = 1, \dots, m$. Then, dealing with the third term in (6.139) in this case yields

$$\hat{\mu}_j = \frac{\sum_{t=1}^n \pi'_j(t|n) y_t}{\sum_{t=1}^n \pi'_j(t|n)}, \quad \hat{\sigma}_j^2 = \frac{\sum_{t=1}^n \pi'_j(t|n) y_t^2}{\sum_{t=1}^n \pi'_j(t|n)} - \hat{\mu}_j^2.$$

In this example, we fit a normal HMM using the R package `depmixS4` to the weekly S&P 500 returns displayed in Figure 6.14. We chose a three-state model and we leave it to the reader to investigate a two-state model (see Problem 6.24). Standard errors (shown in parentheses below) were obtained via a parametric bootstrap based on a simulation script provided with the package.

If we let $P = \{\pi_{ij}\}$ denote the 3×3 matrix of transition probabilities, the fitted transition matrix was

$$\widehat{P} = \begin{bmatrix} .945_{(.074)} & .055_{(.074)} & .000_{(.000)} \\ .739_{(.275)} & .000_{(.000)} & .261_{(.275)} \\ .032_{(.122)} & .027_{(.057)} & .942_{(.147)} \end{bmatrix},$$

and the three fitted normals were $N(\hat{\mu}_1 = .004_{(.173)}, \hat{\sigma}_1 = .014_{(.968)})$, $N(\hat{\mu}_2 = -.034_{(.909)}, \hat{\sigma}_2 = .009_{(.777)})$, and $N(\hat{\mu}_3 = -.003_{(.317)}, \hat{\sigma}_3 = .044_{(.910)})$. The data, along with the predicted state (based on the smoothing distribution), are plotted in Figure 6.14.

Note that regime 2 appears to represent a somewhat large-in-magnitude negative return, and may be a lone dip, or the start or end of a highly volatile period. States 1 and 3 represent clusters of regular or high volatility, respectively. Note that there is a large amount of uncertainty in the fitted normals, and in the transition matrix involving transitions from state 2 to states 1 or 3. The R code for this example is:

```

library(depmixS4)
y = ts(sp500w, start=2003, freq=52) # make data depmix friendly

```

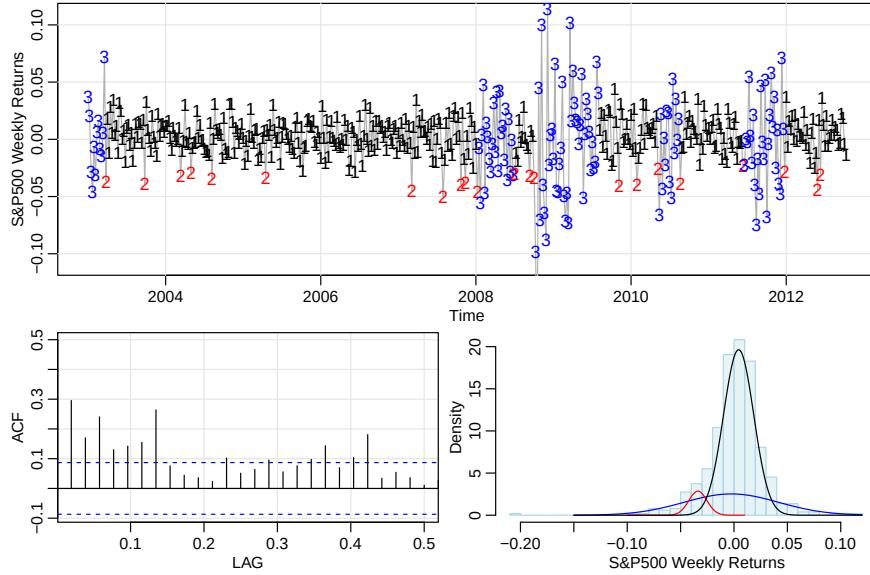


Fig. 6.14. Top: S&P 500 weekly returns with estimated regimes labeled as a number, 1, 2, or 3. The minimum value of -20% during the financial crisis has been truncated to improve the graphics. Bottom left: Sample ACF of the squared returns. Bottom right: Histogram of the data with the three estimated normal densities superimposed.

```

mod3 <- depmix(y~1, nstates=3, data=data.frame(y))
set.seed(2)
summary(fm3 <- fit(mod3))
##-- Graphics --##
layout(matrix(c(1,2, 1,3), 2), heights=c(1,.75))
par(mar=c(2.5,2.5,.5,.5), mgp=c(1.6,.6,0))
plot(y, main="", ylab='S&P500 Weekly Returns', col=gray(.7),
      ylim=c(-.11,.11))
culer = 4-posterior(fm3)[,1]; culer[culer==3]=4 # switch labels 1 and 3
text(y, col=culer, labels=4-posterior(fm3)[,1])
##-- MLEs --##
para.mle   = as.vector(getpars(fm3)[-1:3])
permu      = matrix(c(0,0,1,0,1,0,1,0,0), 3,3) # for the label switch
(mtrans.mle = permu%*%round(t(matrix(para.mle[1:9], 3,3)),3)%*%permu)
(norms.mle = round(matrix(para.mle[10:15], 2,3),3)%*%permu)
acf(y^2, xlim=c(.02,.5), ylim=c(-.09,.5), panel.first=grid(lty=2) )
hist(y, 25, prob=TRUE, main='')
culer=c(1,2,4); pi.hat = colSums(posterior(fm3)[-1,2:4])/length(y)
for (i in 1:3) { mu=norms.mle[1,i]; sig = norms.mle[2,i]
x = seq(-.15,.12, by=.001)
lines(x, pi.hat[4-i]*dnorm(x, mean=mu, sd=sig), col=culer[i]) }
##-- Bootstrap --##
set.seed(666); n.obs = length(y); n.boot = 100
para.star = matrix(NA, nrow=n.boot, ncol = 15)
respst <- para.mle[10:15]; trst <- para.mle[1:9]
for ( nb in 1:n.boot ){

```

```

mod <- simulate(mod3)
y.star = as.vector(mod@response[[1]][[1]]@y)
dfy = data.frame(y.star)
mod.star <- depmix(y.star~1, data=dfy, respst=respst, trst=trst, nst=3)
fm.star = fit(mod.star, emcontrol=em.control(tol = 1e-5), verbose=FALSE)
para.star[nb,] = as.vector(getpars(fm.star)[-c(1:3)])
# bootstrap stdn errors
SE = sqrt(apply(para.star, 2, var) + (apply(para.star, 2, mean)-para.mle)^2)
(SE.mtrans.mle = permu%*%round(t(matrix(SE[1:9], 3, 3)), 3)%*%permu)
(SE.norms.mle = round(matrix(SE[10:15], 2, 3), 3)%*%permu)

```

It is worth mentioning that *switching regressions* also fits into this framework. In this case, we would change μ_j in the model in Example 6.17 to depend on independent inputs, say z_{t1}, \dots, z_{tr} , so that

$$\mu_j = \beta_0^{(j)} + \sum_{i=1}^r \beta_i^{(j)} z_{ti}.$$

This type of model is easily handled using the `depmixS4` R package.

By conditioning on the first few observations, it is also possible to include simple switching linear autoregression into this framework. In this case, we model the observations as being an AR(p), with parameters depending on the state; that is,

$$y_t = \phi_0^{(x_t)} + \sum_{i=1}^p \phi_i^{(x_t)} y_{t-i} + \sigma^{(x_t)} v_t, \quad (6.147)$$

and $v_t \sim \text{iid } N(0, 1)$. The model is similar to the threshold model discussed in Section 5.4, however, the process is not self-exciting or influenced by an observed exogenous process. In (6.147), we are saying that the parameters are random, and the regimes are changing due to a latent Markov process. In a similar fashion to (6.131), we write the conditional distribution of the observations as

$$p_j(y_t) = p(y_t | x_t = j, y_{t-1:t-p}), \quad (6.148)$$

and we note that for $t > p$, $p_j(y_t)$ is the normal density (g),

$$p_j(y_t) = g\left(y_t; \phi_0^{(j)} + \sum_{i=1}^p \phi_i^{(j)} y_{t-i}, \sigma^{2(j)}\right). \quad (6.149)$$

As in (6.138), the conditional likelihood is given by

$$\ln L_Y(\Theta | y_{1:p}) = \sum_{t=p+1}^n \ln \left(\sum_{j=1}^m \pi_j(t | t-1) p_j(y_t) \right).$$

where Property 6.7 still applies, but with the updated evaluation of $p_j(y_t)$ given in (6.149). In addition, the EM algorithm may be used analogously by assessing the smoothers. The smoothers in this case are symbolically the same as given in Property 6.8 with the appropriate definition changes, $p_j(y_t)$ as given in (6.148) and with $\varphi_j(t) = p(y_{t+1:n} | x_t = j, y_{t+1-p:t})$ for $t > p$.

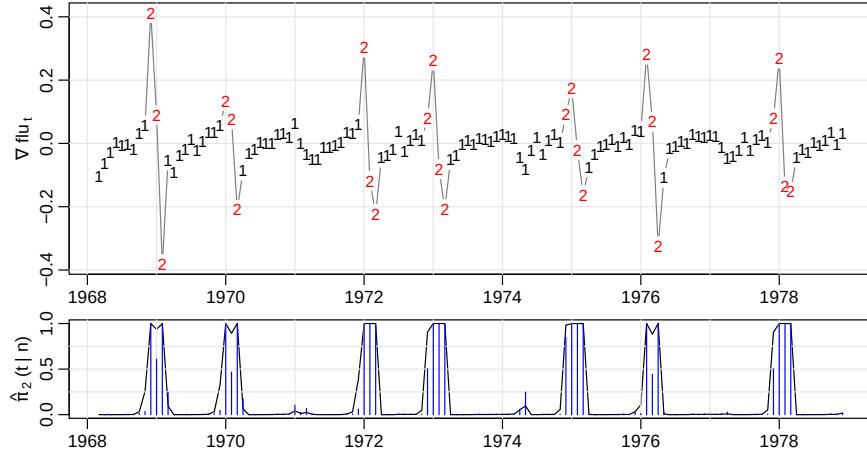


Fig. 6.15. The differenced flu mortality data along with the estimated states (displayed as points). The smoothed state 2 probabilities are displayed in the bottom of the figure as a straight line. The filtered state 2 probabilities are displayed as vertical lines.

Example 6.18 Switching AR – Influenza Mortality

In Example 5.7, we discussed the monthly pneumonia and influenza mortality series shown in Figure 5.7. We pointed out the non-reversibility of the series, which rules out the possibility that the data are generated by a linear Gaussian process. In addition, note that the series is irregular, and while mortality is highest during the winter, the peak does not occur in the same month each year. Moreover, some seasons have very large peaks, indicating flu epidemics, whereas other seasons are mild. In addition, it can be seen from Figure 5.7 that there is a slight negative trend in the data set, indicating that flu prevention is getting better over the eleven year period.

As in Example 5.7, we focus on the differenced data, which removes the trend. In this case, we denote $y_t = \nabla \text{flu}_t$, where flu_t represents the data displayed in Figure 5.7. Since we already fit a threshold model to y_t , we might also consider a switching autoregressive model where there are two hidden regimes, one for epidemic periods and one for more mild periods. In this case, the model is given by

$$y_t = \begin{cases} \phi_0^{(1)} + \sum_{j=1}^p \phi_j^{(1)} y_{t-j} + \sigma^{(1)} v_t, & \text{for } x_t = 1, \\ \phi_0^{(2)} + \sum_{j=1}^p \phi_j^{(2)} y_{t-j} + \sigma^{(2)} v_t, & \text{for } x_t = 2, \end{cases} \quad (6.150)$$

where $v_t \sim \text{iid } N(0, 1)$, and x_t is a latent, two-state Markov chain.

We used the R package `MSwM` to fit the model specified in (6.150), with $p = 2$. The results were

$$\hat{y}_t = \begin{cases} .006_{(.003)} + .293_{(.039)} y_{t-1} + .097_{(.031)} y_{t-2} + .024 v_t, & \text{for } x_t = 1, \\ .199_{(.063)} - .313_{(.281)} y_{t-1} - 1.604_{(.276)} y_{t-2} + .112 v_t, & \text{for } x_t = 2, \end{cases}$$

with estimated transition matrix

$$\hat{P} = \begin{bmatrix} .93 & .07 \\ .30 & .70 \end{bmatrix}.$$

Figure 6.15 displays the data $y_t = \nabla \text{flu}_t$ along with the estimated states (displayed as points labeled 1 or 2). The smoothed state 2 probabilities are displayed in the bottom of the figure as a straight line. The filtered state 2 probabilities are displayed in the same graph as vertical lines. The code for this example is as follows.

```
library(MSwM)
set.seed(90210)
dflu = diff(flu)
model = lm(dflu~ 1)
mod = msmFit(model, k=2, p=2, sw=rep(TRUE,4)) # 2 regimes, AR(2)s
summary(mod)
plotProb(mod, which=3)
```

6.10 Dynamic Linear Models with Switching

In this section, we extend the hidden Markov model discussed in Section 6.9 to more general problems. As previously indicated, the problem of modeling changes in regimes for time series has been of interest in many different fields, and we have explored these ideas in Section 5.4 as well as in Section 6.9.

Generalizations of the state space model to include the possibility of changes occurring over time have been approached by allowing changes in the error covariances (Harrison and Stevens, 1976, Gordon and Smith, 1988, 1990) or by assigning mixture distributions to the observation errors v_t (Peña and Guttman, 1988). Approximations to filtering were derived in all of the aforementioned articles. An application to monitoring renal transplants was described in Smith and West (1983) and in Gordon and Smith (1990). Gerlach et al. (2000) considered an extension of the switching AR model to allow for level shifts and outliers in both the observations and innovations. An application of the idea of switching to the tracking of multiple targets has been considered in Bar-Shalom (1978), who obtained approximations to Kalman filtering in terms of weighted averages of the innovations. For a thorough coverage of these and related techniques, see Cappé, Moulines, & Rydén (2009) and Douc, Moulines, & Stoffer (2014).

In this section, we will concentrate on the method presented in Shumway and Stoffer (1991). One way of modeling change in an evolving time series is by assuming the dynamics of some underlying model changes discontinuously at certain undetermined points in time. Our starting point is the DLM given by (6.1) and (6.2), namely,

$$x_t = \Phi x_{t-1} + w_t, \quad (6.151)$$

to describe the $p \times 1$ state dynamics, and

$$y_t = A_t x_t + v_t \quad (6.152)$$

to describe the $q \times 1$ observation dynamics. Recall w_t and v_t are Gaussian white noise sequences with $\text{var}(w_t) = Q$, $\text{var}(v_t) = R$, and $\text{cov}(w_t, v_s) = 0$ for all s and t .

Example 6.19 Tracking Multiple Targets

The approach of Shumway and Stoffer (1991) was motivated primarily by the problem of tracking a large number of moving targets using a vector y_t of sensors. In this problem, we do not know at any given point in time which target any given sensor has detected. Hence, it is the structure of the measurement matrix A_t in (6.152) that is changing, and not the dynamics of the signal x_t or the noises, w_t or v_t . As an example, consider a 3×1 vector of satellite measurements $y_t = (y_{t1}, y_{t2}, y_{t3})'$ that are observations on some combination of a 3×1 vector of targets or signals, $x_t = (x_{t1}, x_{t2}, x_{t3})'$. For the measurement matrix

$$A_t = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

for example, the first sensor, y_{t1} , observes the second target, x_{t2} ; the second sensor, y_{t2} , observes the first target, x_{t1} ; and the third sensor, y_{t3} , observes the third target, x_{t3} . All possible detection configurations will define a set of possible values for A_t , say, $\{M_1, M_2, \dots, M_m\}$, as a collection of plausible measurement matrices.

Example 6.20 Modeling Economic Change

As another example of the switching model presented in this section, consider the case in which the dynamics of the linear model changes suddenly over the history of a given realization. For example, Lam (1990) has given the following generalization of Hamilton's (1989) model for detecting positive and negative growth periods in the economy. Suppose the data are generated by

$$y_t = z_t + n_t, \quad (6.153)$$

where z_t is an autoregressive series and n_t is a random walk with a drift that switches between two values α_0 and $\alpha_0 + \alpha_1$. That is,

$$n_t = n_{t-1} + \alpha_0 + \alpha_1 S_t, \quad (6.154)$$

with $S_t = 0$ or 1 , depending on whether the system is in state 1 or state 2. For the purpose of illustration, suppose

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + w_t \quad (6.155)$$

is an AR(2) series with $\text{var}(w_t) = \sigma_w^2$. Lam (1990) wrote (6.153) in a differenced form

$$\nabla y_t = z_t - z_{t-1} + \alpha_0 + \alpha_1 S_t, \quad (6.156)$$

which we may take as the observation equation (6.152) with state vector

$$x_t = (z_t, z_{t-1}, \alpha_0, \alpha_1)' \quad (6.157)$$

and

$$M_1 = [1, -1, 1, 0] \quad \text{and} \quad M_2 = [1, -1, 1, 1] \quad (6.158)$$

determining the two possible economic conditions. The state equation, (6.151), is of the form

$$\begin{pmatrix} z_t \\ z_{t-1} \\ \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{bmatrix} \phi_1 & \phi_2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} z_{t-1} \\ z_{t-2} \\ \alpha_0 \\ \alpha_1 \end{pmatrix} + \begin{pmatrix} w_t \\ 0 \\ 0 \\ 0 \end{pmatrix}. \quad (6.159)$$

The observation equation, (6.156), can be written as

$$\nabla y_t = A_t x_t + v_t, \quad (6.160)$$

where we have included the possibility of observational noise, and where $\Pr(A_t = M_1) = 1 - \Pr(A_t = M_2)$, with M_1 and M_2 given in (6.158).

To incorporate a reasonable switching structure for the measurement matrix into the DLM that is compatible with both practical situations previously described, we assume that the m possible configurations are states in a nonstationary, independent process defined by the time-varying probabilities

$$\pi_j(t) = \Pr(A_t = M_j), \quad (6.161)$$

for $j = 1, \dots, m$ and $t = 1, 2, \dots, n$. Important information about the current state of the measurement process is given by the filtered probabilities of being in state j , defined as the conditional probabilities

$$\pi_j(t | t) = \Pr(A_t = M_j | y_{1:t}), \quad (6.162)$$

which also vary as a function of time. Recall that $y_{s':s} = \{y_{s'}, \dots, y_s\}$. The filtered probabilities (6.162) give the time-varying estimates of the probability of being in state j given the data to time t .

It will be important for us to obtain estimators of the configuration probabilities, $\pi_j(t | t)$, the predicted and filtered state estimators, x_t^{t-1} and x_t^t , and the corresponding error covariance matrices P_t^{t-1} and P_t^t . Of course, the predictor and filter estimators will depend on the parameters, Θ , of the DLM. In many situations, the parameters will be unknown and we will have to estimate them. Our focus will be on maximum likelihood estimation, but other authors have taken a Bayesian approach that assigns priors to the parameters, and then seeks posterior distributions of the model parameters; see, for example, Gordon and Smith (1990), Peña and Guttman (1988), or McCulloch and Tsay (1993).

We now establish the recursions for the filters associated with the state x_t and the switching process, A_t . As discussed in Section 6.3, the filters are also an essential part of the maximum likelihood procedure. The predictors, $x_t^{t-1} = E(x_t | y_{1:t-1})$, and filters, $x_t^t = E(x_t | y_{1:t})$, and their associated error variance-covariance matrices, P_t^{t-1} and P_t^t , are given by

$$x_t^{t-1} = \Phi x_{t-1}^{t-1}, \quad (6.163)$$

$$P_t^{t-1} = \Phi P_{t-1}^{t-1} \Phi' + Q, \quad (6.164)$$

$$x_t^t = x_t^{t-1} + \sum_{j=1}^m \pi_j(t|t) K_{tj} \epsilon_{tj}, \quad (6.165)$$

$$P_t^t = \sum_{j=1}^m \pi_j(t|t) (I - K_{tj} M_j) P_t^{t-1}, \quad (6.166)$$

$$K_{tj} = P_t^{t-1} M_j' \Sigma_{tj}^{-1}, \quad (6.167)$$

where the innovation values in (6.165) and (6.167) are

$$\epsilon_{tj} = y_t - M_j x_t^{t-1}, \quad (6.168)$$

$$\Sigma_{tj} = M_j P_t^{t-1} M_j' + R, \quad (6.169)$$

for $j = 1, \dots, m$.

Equations (6.163)–(6.167) exhibit the filter values as weighted linear combinations of the m innovation values, (6.168)–(6.169), corresponding to each of the possible measurement matrices. The equations are similar to the approximations introduced by Bar-Shalom and Tse (1975), by Gordon and Smith (1990), and Peña and Guttman (1988).

To verify (6.165), let the indicator $I(A_t = M_j) = 1$ when $A_t = M_j$, and zero otherwise. Then, using (6.20),

$$\begin{aligned} x_t^t &= E(x_t | y_{1:t}) = E[E(x_t | y_{1:t}, A_t) | y_{1:t}] \\ &= E\left\{\sum_{j=1}^m E(x_t | y_{1:t}, A_t = M_j) I(A_t = M_j) | y_{1:t}\right\} \\ &= E\left\{\sum_{j=1}^m [x_t^{t-1} + K_{tj}(y_t - M_j x_t^{t-1})] I(A_t = M_j) | y_{1:t}\right\} \\ &= \sum_{j=1}^m \pi_j(t | t) [x_t^{t-1} + K_{tj}(y_t - M_j x_t^{t-1})], \end{aligned}$$

where K_{tj} is given by (6.167). Equation (6.166) is derived in a similar fashion; the other relationships, (6.163), (6.164), and (6.167), follow from straightforward applications of the Kalman filter results given in [Property 6.1](#).

Next, we derive the filters $\pi_j(t|t)$. Let $p_j(t | t-1)$ denote the conditional density of y_t given the past $y_{1:t-1}$, and $A_t = M_j$, for $j = 1, \dots, m$. Then,

$$\pi_j(t | t) = \frac{\pi_j(t)p_j(t | t-1)}{\sum_{k=1}^m \pi_k(t)p_k(t | t-1)}, \quad (6.170)$$

where we assume the distribution $\pi_j(t)$, for $j = 1, \dots, m$ has been specified before observing $y_{1:t}$ (details follow as in [Example 6.21](#) below). If the investigator has

no reason to prefer one state over another at time t , the choice of uniform priors, $\pi_j(t) = m^{-1}$, for $j = 1, \dots, m$, will suffice. Smoothness can be introduced by letting

$$\pi_j(t) = \sum_{i=1}^m \pi_i(t-1 | t-1) \pi_{ij}, \quad (6.171)$$

where the non-negative weights π_{ij} are chosen so $\sum_{i=1}^m \pi_{ij} = 1$. If the A_t process was Markov with transition probabilities π_{ij} , then (6.171) would be the update for the filter probability, as shown in the next example.

Example 6.21 Hidden Markov Chain Model

If $\{A_t\}$ is a hidden Markov chain with stationary transition probabilities $\pi_{ij} = \Pr(A_t = M_j | A_{t-1} = M_i)$, for $i, j = 1, \dots, m$, we have

$$\begin{aligned} \pi_j(t | t) &= \frac{p(A_t = M_j, y_t | y_{1:t-1})}{p(y_t | y_{1:t-1})} \\ &= \frac{\Pr(A_t = M_j | y_{1:t-1}) p(y_t | A_t = M_j, y_{1:t-1})}{p(y_t | y_{1:t-1})} \\ &= \frac{\pi_j(t | t-1) p_j(t | t-1)}{\sum_{k=1}^m \pi_k(t | t-1) p_k(t | t-1)}. \end{aligned} \quad (6.172)$$

In the Markov case, the conditional probabilities

$$\pi_j(t | t-1) = \Pr(A_t = M_j | y_{1:t-1})$$

in (6.172) replace the unconditional probabilities, $\pi_j(t) = \Pr(A_t = M_j)$, in (6.170).

To evaluate (6.172), we must be able to calculate $\pi_j(t | t-1)$ and $p_j(t | t-1)$. We will discuss the calculation of $p_j(t | t-1)$ after this example. To derive $\pi_j(t | t-1)$, note,

$$\begin{aligned} \pi_j(t | t-1) &= \Pr(A_t = M_j | y_{1:t-1}) \\ &= \sum_{i=1}^m \Pr(A_t = M_j, A_{t-1} = M_i | y_{1:t-1}) \\ &= \sum_{i=1}^m \Pr(A_t = M_j | A_{t-1} = M_i) \Pr(A_{t-1} = M_i | y_{1:t-1}) \\ &= \sum_{i=1}^m \pi_{ij} \pi_i(t-1 | t-1). \end{aligned} \quad (6.173)$$

Expression (6.171) comes from equation (6.173), where, as previously noted, we replace $\pi_j(t | t-1)$ by $\pi_j(t)$.

The difficulty in extending the approach here to the Markov case is the dependence among the y_t , which makes it necessary to enumerate over all possible histories to derive the filtering equations. This problem will be evident when we derive the

conditional density $p_j(t \mid t-1)$. Equation (6.171) has $\pi_j(t)$ as a function of the past observations, $y_{1:t-1}$, which is inconsistent with our model assumption. Nevertheless, this seems to be a reasonable compromise that allows the data to modify the probabilities $\pi_j(t)$, without having to develop a highly computer-intensive technique.

As previously suggested, the computation of $p_j(t \mid t-1)$, without some approximations, is highly computer-intensive. To evaluate $p_j(t \mid t-1)$, consider the event

$$\{A_1 = M_{j_1}, \dots, A_{t-1} = M_{j_{t-1}}\}, \quad (6.174)$$

for $j_i = 1, \dots, m$, and $i = 1, \dots, t-1$, which specifies a specific set of measurement matrices through the past; we will write this event as $A_{(t-1)} = M_{(\ell)}$. Because m^{t-1} possible outcomes exist for A_1, \dots, A_{t-1} , the index ℓ runs through $\ell = 1, \dots, m^{t-1}$. Using this notation, we may write

$$\begin{aligned} p_j(t \mid t-1) &= \sum_{\ell=1}^{m^{t-1}} \Pr\{A_{(t-1)} = M_{(\ell)} \mid y_{1:t-1}\} p(y_t \mid y_{1:t-1}, A_t = M_j, A_{(t-1)} = M_{(\ell)}) \\ &\stackrel{\text{def}}{=} \sum_{\ell=1}^{m^{t-1}} \alpha(\ell) g(y_t; \mu_{tj}(\ell), \Sigma_{tj}(\ell)), \quad j = 1, \dots, m, \end{aligned} \quad (6.175)$$

where $g(\cdot; \mu, \Sigma)$ represents the normal density with mean vector μ and variance-covariance matrix Σ . Thus, $p_j(t \mid t-1)$ is a mixture of normals with non-negative weights $\alpha(\ell) = \Pr\{A_{(t-1)} = M_{(\ell)} \mid y_{1:t-1}\}$ such that $\sum_{\ell} \alpha(\ell) = 1$, and with each normal distribution having mean vector

$$\mu_{tj}(\ell) = M_j x_t^{t-1}(\ell) = M_j E[x_t \mid y_{1:t-1}, A_{(t-1)} = M_{(\ell)}] \quad (6.176)$$

and covariance matrix

$$\Sigma_{tj}(\ell) = M_j P_t^{t-1}(\ell) M_j' + R. \quad (6.177)$$

This result follows because the conditional distribution of y_t in (6.175) is identical to the fixed measurement matrix case presented in Section 6.2. The values in (6.176) and (6.177), and hence the densities, $p_j(t \mid t-1)$, for $j = 1, \dots, m$, can be obtained directly from the Kalman filter, Property 6.1, with the measurement matrices $A_{(t-1)}$ fixed at $M_{(\ell)}$.

Although $p_j(t \mid t-1)$ is given explicitly in (6.175), its evaluation is highly computer intensive. For example, with $m = 2$ states and $n = 20$ observations, we have to filter over $2+2^2+\dots+2^{20}$ possible sample paths ($2^{20} = 1,048,576$). There are a few remedies to this problem. An algorithm that makes it possible to efficiently compute the most likely sequence of states given the data is known as the *Viterbi algorithm*, which is based on the well-known dynamic programming principle. Details may be found in Douc et al. (2014, §9.2). Another remedy is to trim (remove), at each t , highly improbable sample paths; that is, remove events in (6.174) with extremely small probability of occurring, and then evaluate $p_j(t \mid t-1)$ as if the trimmed sample paths could not have occurred. Another rather simple alternative, as suggested by Gordon

and Smith (1990) and Shumway and Stoffer (1991), is to approximate $p_j(t | t - 1)$ using the closest (in the sense of Kulback–Leibler distance) normal distribution. In this case, the approximation leads to choosing normal distribution with the same mean and variance associated with $p_j(t | t - 1)$; that is, we approximate $p_j(t | t - 1)$ by a normal with mean $M_j x_t^{t-1}$ and variance Σ_{tj} given in (6.169).

To develop a procedure for maximum likelihood estimation, the joint density of the data is

$$\begin{aligned} f(y_1, \dots, y_n) &= \prod_{t=1}^n f(y_t | y_{1:t-1}) \\ &= \prod_{t=1}^n \sum_{j=1}^m \Pr(A_t = M_j | y_{1:t-1}) p(y_t | A_t = M_j, y_{1:t-1}), \end{aligned}$$

and hence, the likelihood can be written as

$$\ln L_Y(\Theta) = \sum_{t=1}^n \ln \left(\sum_{j=1}^m \pi_j(t) p_j(t | t - 1) \right). \quad (6.178)$$

For the hidden Markov model, $\pi_j(t)$ would be replaced by $\pi_j(t | t - 1)$. In (6.178), we will use the normal approximation to $p_j(t | t - 1)$. That is, henceforth, we will consider $p_j(t | t - 1)$ as the normal, $N(M_j x_t^{t-1}, \Sigma_{tj})$, density, where x_t^{t-1} is given in (6.163) and Σ_{tj} is given in (6.169). We may consider maximizing (6.178) directly as a function of the parameters Θ in $\{\mu_0, \Phi, Q, R\}$ using a Newton method, or we may consider applying the EM algorithm to the complete data likelihood.

To apply the EM algorithm as in Section 6.3, we call $x_{0:n}$, $A_{1:n}$, and $y_{1:n}$, the complete data, with likelihood given by

$$\begin{aligned} -2 \ln L_{X,A,Y}(\Theta) &= \ln |\Sigma_0| + (x_0 - \mu_0)' \Sigma_0^{-1} (x_0 - \mu_0) \\ &\quad + n \ln |Q| + \sum_{t=1}^n (x_t - \Phi x_{t-1})' Q^{-1} (x_t - \Phi x_{t-1}) \\ &\quad - 2 \sum_{t=1}^n \sum_{j=1}^m I(A_t = M_j) \ln \pi_j(t) + n \ln |R| \\ &\quad + \sum_{t=1}^n \sum_{j=1}^m I(A_t = M_j) (y_t - A_t x_t)' R^{-1} (y_t - A_t x_t). \end{aligned} \quad (6.179)$$

As discussed in Section 6.3, we require the minimization of the conditional expectation

$$\mathcal{Q}(\Theta | \Theta^{(k-1)}) = E \left\{ -2 \ln L_{X,A,Y}(\Theta) \mid y_{1:n}, \Theta^{(k-1)} \right\}, \quad (6.180)$$

with respect to Θ at each iteration, $k = 1, 2, \dots$. The calculation and maximization of (6.180) is similar to the case of (6.63). In particular, with

$$\pi_j(t | n) = E[I(A_t = M_j) \mid y_{1:n}], \quad (6.181)$$

we obtain on iteration k ,

$$\pi_j^{(k)}(t) = \pi_j(t | n), \quad (6.182)$$

$$\mu_0^{(k)} = x_0^n, \quad (6.183)$$

$$\Phi^{(k)} = S_{10}S_{00}^{-1}, \quad (6.184)$$

$$Q^{(k)} = n^{-1} \left(S_{11} - S_{10}S_{00}^{-1}S_{10}' \right), \quad (6.185)$$

and

$$R^{(k)} = n^{-1} \sum_{t=1}^n \sum_{j=1}^m \pi_j(t | n) \left[(y_t - M_j x_t^n)(y_t - M_j x_t^n)' + M_j P_t^n M_j' \right]. \quad (6.186)$$

where S_{11}, S_{10}, S_{00} are given in (6.65)–(6.67). As before, at iteration k , the filters and the smoothers are calculated using the current values of the parameters, $\Theta^{(k-1)}$, and Σ_0 is held fixed. Filtering is accomplished by using (6.163)–(6.167). Smoothing is derived in a similar manner to the derivation of the filter, and one is led to the smoother given in [Property 6.2](#) and [Property 6.3](#), with one exception, the initial smoother covariance, (6.53), is now

$$P_{n,n-1}^n = \sum_{j=1}^m \pi_j(n | n) (I - K_{tj} M_j) \Phi P_{n-1}^{n-1}. \quad (6.187)$$

Unfortunately, the computation of $\pi_j(t | n)$ is excessively complicated, and requires integrating over mixtures of normal distributions. Shumway and Stoffer (1991) suggest approximating the smoother $\pi_j(t | n)$ by the filter $\pi_j(t | t)$, and find the approximation works well.

Example 6.22 Analysis of the Influenza Data

We use the results of this section to analyze the U.S. monthly pneumonia and influenza mortality data plotted in [Figure 5.7](#). Letting y_t denote the observations at month t , we model y_t in terms of a structural component model coupled with a hidden Markov process that determines whether a flu epidemic exists.

The model consists of three structural components. The first component, x_{t1} , is an AR(2) process chosen to represent the periodic (seasonal) component of the data,

$$x_{t1} = \alpha_1 x_{t-1,1} + \alpha_2 x_{t-2,1} + w_{t1}, \quad (6.188)$$

where w_{t1} is white noise, with $\text{var}(w_{t1}) = \sigma_1^2$. The second component, x_{t2} , is an AR(1) process with a nonzero constant term, which is chosen to represent the sharp rise in the data during an epidemic,

$$x_{t2} = \beta_0 + \beta_1 x_{t-1,2} + w_{t2}, \quad (6.189)$$

where w_{t2} is white noise, with $\text{var}(w_{t2}) = \sigma_2^2$. The third component, x_{t3} , is a fixed trend component given by,

$$x_{t3} = x_{t-1,3} + w_{t3}, \quad (6.190)$$

where $\text{var}(w_{t3}) = 0$. The case in which $\text{var}(w_{t3}) > 0$, which corresponds to a stochastic trend (random walk), was tried here, but the estimation became unstable, and lead to us fitting a fixed, rather than stochastic, trend. Thus, in the final model, the trend component satisfies $\nabla x_{t3} = 0$; recall in [Example 6.18](#) the data were also differenced once before fitting the model.

Throughout the years, periods of normal influenza mortality (state 1) are modeled as

$$y_t = x_{t1} + x_{t3} + v_t, \quad (6.191)$$

where the measurement error, v_t , is white noise with $\text{var}(v_t) = \sigma_v^2$. When an epidemic occurs (state 2), mortality is modeled as

$$y_t = x_{t1} + x_{t2} + x_{t3} + v_t. \quad (6.192)$$

The model specified in [\(6.188\)](#)–[\(6.192\)](#) can be written in the general state-space form. The state equation is

$$\begin{pmatrix} x_{t1} \\ x_{t-1,1} \\ x_{t2} \\ x_{t3} \end{pmatrix} = \begin{bmatrix} \alpha_1 & \alpha_2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \beta_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_{t-1,1} \\ x_{t-2,1} \\ x_{t-1,2} \\ x_{t-1,3} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \beta_0 \\ 0 \end{pmatrix} + \begin{pmatrix} w_{t1} \\ 0 \\ w_{t2} \\ 0 \end{pmatrix}. \quad (6.193)$$

Of course, [\(6.193\)](#) can be written in the standard state-equation form as

$$x_t = \Phi x_{t-1} + \Upsilon u_t + w_t, \quad (6.194)$$

where $x_t = (x_{t1}, x_{t-1,1}, x_{t2}, x_{t3})'$, $\Upsilon = (0, 0, \beta_0, 0)'$, $u_t \equiv 1$, and Φ is a 4×4 matrix with σ_1^2 as the (1,1)-element, σ_2^2 as the (3,3)-element, and the remaining elements set equal to zero. The observation equation is

$$y_t = A_t x_t + v_t, \quad (6.195)$$

where A_t is 1×4 , and v_t is white noise with $\text{var}(v_t) = R = \sigma_v^2$. We assume all components of variance w_{t1} , w_{t2} , and v_t are uncorrelated.

As discussed in [\(6.191\)](#) and [\(6.192\)](#), A_t can take one of two possible forms

$$\begin{aligned} A_t = M_1 &= [1, 0, 0, 1] && \text{no epidemic,} \\ A_t = M_2 &= [1, 0, 1, 1] && \text{epidemic,} \end{aligned}$$

corresponding to the two possible states of (1) no flu epidemic and (2) flu epidemic, such that $\Pr(A_t = M_1) = 1 - \Pr(A_t = M_2)$. In this example, we will assume A_t is a hidden Markov chain, and hence we use the updating equations given in [Example 6.21](#), [\(6.172\)](#) and [\(6.173\)](#), with transition probabilities $\pi_{11} = \pi_{22} = .75$ (and, thus, $\pi_{12} = \pi_{21} = .25$).

Parameter estimation was accomplished using a quasi-Newton–Raphson procedure to maximize the approximate log likelihood given in [\(6.178\)](#), with initial

Table 6.3. Estimation Results for Influenza Data

Parameter	Initial Model Estimates	Final Model Estimates
α_1	1.422 (.100)	1.406 (.079)
α_2	-.634 (.089)	-.622 (.069)
β_0	.276 (.056)	.210 (.025)
β_1	-.312 (.218)	—
σ_1	.023 (.003)	.023 (.005)
σ_2	.108 (.017)	.112 (.017)
σ_v	.002 (.009)	—

Estimated standard errors in parentheses

values of $\pi_1(1 | 0) = \pi_2(1 | 0) = .5$. **Table 6.3** shows the results of the estimation procedure. On the initial fit, two estimates are not significant, namely, $\hat{\beta}_1$ and $\hat{\sigma}_v$. When $\sigma_v^2 = 0$, there is no measurement error, and the variability in data is explained solely by the variance components of the state system, namely, σ_1^2 and σ_2^2 . The case in which $\beta_1 = 0$ corresponds to a simple level shift during a flu epidemic. In the final model, with β_1 and σ_v^2 removed, the estimated level shift ($\hat{\beta}_0$) corresponds to an increase in mortality by about .2 per 1000 during a flu epidemic. The estimates for the final model are also listed in **Table 6.3**.

Figure 6.16(a) shows a plot of the data, y_t , for the ten-year period of 1969–1978 as well as an indicator that takes the value of 1 if $\hat{\pi}_1(t | t - 1) \geq .5$, or 2 if $\hat{\pi}_2(t | t - 1) > .5$. The estimated prediction probabilities do a reasonable job of predicting a flu epidemic, although the peak in 1972 is missed.

Figure 6.16(b) shows the estimated filtered values (that is, filtering is done using the parameter estimates) of the three components of the model, x_{t1}^t , x_{t2}^t , and x_{t3}^t . Except for initial instability (which is not shown), \hat{x}_{t1}^t represents the seasonal (cyclic) aspect of the data, \hat{x}_{t2}^t represents the spikes during a flu epidemic, and \hat{x}_{t3}^t represents the slow decline in flu mortality over the ten-year period of 1969–1978.

One-month-ahead prediction, say, \hat{y}_t^{t-1} , is obtained as

$$\hat{y}_t^{t-1} = M_1 \hat{x}_t^{t-1} \quad \text{if } \hat{\pi}_1(t | t - 1) > \hat{\pi}_2(t | t - 1),$$

$$\hat{y}_t^{t-1} = M_2 \hat{x}_t^{t-1} \quad \text{if } \hat{\pi}_1(t | t - 1) \leq \hat{\pi}_2(t | t - 1).$$

Of course, \hat{x}_t^{t-1} is the estimated state prediction, obtained via the filter presented in (6.163)–(6.167) (with the addition of the constant term in the model) using the estimated parameters. The results are shown in **Figure 6.16(c)**. The precision of the forecasts can be measured by the innovation variances, Σ_{t1} when no epidemic is predicted, and Σ_{t2} when an epidemic is predicted. These values become stable quickly, and when no epidemic is predicted, the estimated standard prediction error is approximately .02 (this is the square root of Σ_{t1} for t large); when a flu epidemic is predicted, the estimated standard prediction error is approximately .11.

The results of this analysis are impressive given the small number of parameters and the degree of approximation that was made to obtain a computationally

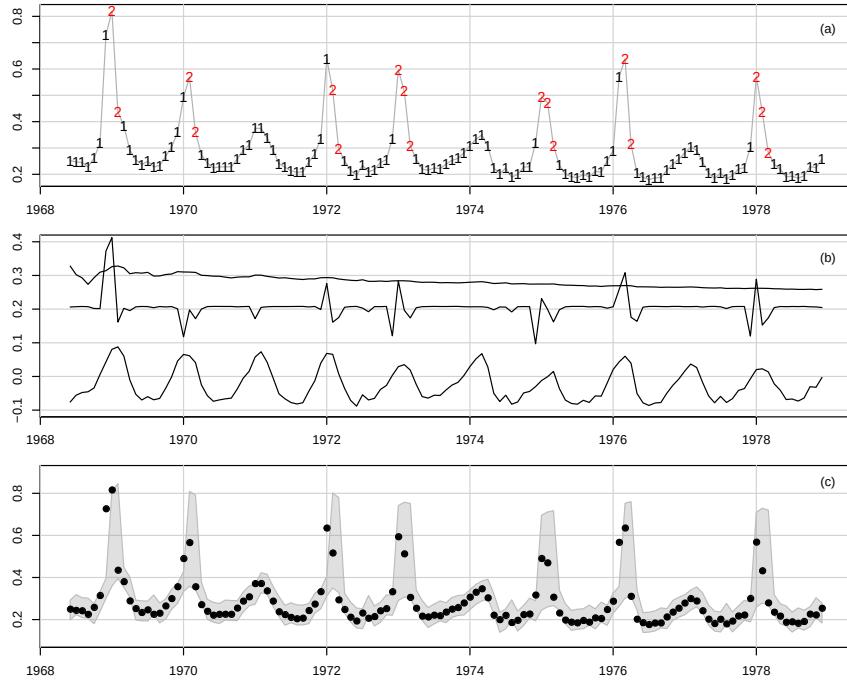


Fig. 6.16. (a) Influenza data, y_t , (line–points) and a prediction indicator (1 or 2) that an epidemic occurs in month t given the data up to month $t-1$ (dashed line). (b) The three filtered structural components of influenza mortality: \hat{x}_{t1}^t (cyclic trace), \hat{x}_{t2}^t (spiked trace), and \hat{x}_{t3}^t (negative linear trace). (c) One-month-ahead predictions shown as upper and lower limits $\hat{y}_t^{t-1} \pm 2\sqrt{\hat{P}_t^{t-1}}$ (gray swatch), of the number of pneumonia and influenza deaths, and y_t (points).

simple method for fitting a complex model. Further evidence of the strength of this technique can be found in the example given in Shumway and Stoffer (1991).

The R code for the final model estimation is as follows.

```

y = as.matrix(flu); num = length(y); nstate = 4;
M1 = as.matrix(cbind(1,0,0,1)) # obs matrix normal
M2 = as.matrix(cbind(1,0,1,1)) # obs matrix flu epi
prob = matrix(0,num,1); yp = y # to store pi2(t/t-1) & y(t/t-1)
xfilter = array(0, dim=c(nstate,1,num)) # to store x(t/t)
# Function to Calculate Likelihood
Linn = function(para){
  alpha1 = para[1]; alpha2 = para[2]; beta0 = para[3]
  sQ1 = para[4]; sQ2 = para[5]; like=0
  xf = matrix(0, nstate, 1) # x filter
  xp = matrix(0, nstate, 1) # x pred
  Pf = diag(.1, nstate) # filter cov
  Pp = diag(.1, nstate) # pred cov
  pi11 <- .75 -> pi22; pi12 <- .25 -> pi21; pif1 <- .5 -> pif2
  phi = matrix(0,nstate,nstate)
}

```

```

phi[1,1] = alpha1; phi[1,2] = alpha2; phi[2,1]=1; phi[4,4]=1
Ups = as.matrix(rbind(0,0,beta0,0))
Q = matrix(0,nstate,nstate)
Q[1,1] = sQ1^2; Q[3,3] = sQ2^2; R=0 # R=0 in final model
# begin filtering #
for(i in 1:num){
  xp = phi%*%xf + Ups; Pp = phi%*%Pf%*%t(phi) + Q
  sig1 = as.numeric(M1%*%Pp%*%t(M1) + R)
  sig2 = as.numeric(M2%*%Pp%*%t(M2) + R)
  k1 = Pp%*%t(M1)/sig1; k2 = Pp%*%t(M2)/sig2
  e1 = y[i]-M1%*%xp; e2 = y[i]-M2%*%xp
  pip1 = pif1*pi11 + pif2*pi21; pip2 = pif1*pi12 + pif2*pi22
  den1 = (1/sqrt(sig1))*exp(-.5*e1^2/sig1)
  den2 = (1/sqrt(sig2))*exp(-.5*e2^2/sig2)
  denom = pip1*den1 + pip2*den2
  pif1 = pip1*den1/denom; pif2 = pip2*den2/denom
  pif1 = as.numeric(pif1); pif2 = as.numeric(pif2)
  e1 = as.numeric(e1); e2=as.numeric(e2)
  xf = xp + pif1*k1*e1 + pif2*k2*e2
  eye = diag(1, nstate)
  Pf = pif1*(eye-k1%*%M1)%*%Pp + pif2*(eye-k2%*%M2)%*%Pp
  like = like - log(pip1*den1 + pip2*den2)
  prob[i]<-pip1; xfilter[,i]<-xf; innov.sig<-c(sig1,sig2)
  yp[i]<-ifelse(pip1 > pip2, M1%*%xp, M2%*%xp) }
return(like) }
# Estimation
alpha1 = 1.4; alpha2 = -.5; beta0 = .3; sQ1 = .1; sQ2 = .1
init.par = c(alpha1, alpha2, beta0, sQ1, sQ2)
(est = optim(init.par, Linn, NULL, method='BFGS', hessian=TRUE,
control=list(trace=1, REPORT=1)))
SE = sqrt(diag(solve(est$hessian)))
u = cbind(estimate=est$par, SE)
rownames(u)=c('alpha1','alpha2','beta0','sQ1','sQ2'); u
            estimate           SE
alpha1   1.40570967 0.078587727
alpha2  -0.62198715 0.068733109
beta0    0.21049042 0.024625302
sQ1     0.02310306 0.001635291
sQ2     0.11217287 0.016684663
# Graphics
predepi = ifelse(prob<.5,0,1); k = 6:length(y)
Time = time(flu)[k]
regime = predepi[k]+1
par(mfrow=c(3,1), mar=c(2,3,1,1)+.1)
plot(Time, y[k], type="n", ylab="")
grid(lty=2); lines(Time, y[k], col=gray(.7))
text(Time, y[k], col=regime, labels=regime, cex=1.1)
text(1979,.95,"(a)")
plot(Time, xfilter[1,,k], type="n", ylim=c(-.1,.4), ylab="")
grid(lty=2); lines(Time, xfilter[1,,k])
lines(Time, xfilter[3,,k]); lines(Time, xfilter[4,,k])
text(1979,.35,"(b)")
plot(Time, y[k], type="n", ylim=c(.1,.9), ylab="")
grid(lty=2); points(Time, y[k], pch=19)
prde1 = 2*sqrt(innov.sig[1]); prde2 = 2*sqrt(innov.sig[2])
prde = ifelse(predepi[k]<.5, prde1,prde2)

```

```

xx = c(Time, rev(Time))
yy = c(yp[k]-prde, rev(yp[k]+prde))
polygon(xx, yy, border=8, col=gray(.6, alpha=.3))
text(1979,.85,"(c)")

```

6.11 Stochastic Volatility

Stochastic volatility (SV) models are an alternative to GARCH-type models that were presented in [Chapter 5](#). Throughout this section, we let r_t denote the returns of some financial asset. Most models for return data used in practice are of a multiplicative form that we have seen in [Section 5.3](#),

$$r_t = \sigma_t \varepsilon_t, \quad (6.196)$$

where ε_t is an iid sequence and the *volatility process*, σ_t , is a non-negative stochastic process such that ε_t is independent of σ_s for all $s \leq t$. It is often assumed that ε_t has zero mean and unit variance.

In SV models, the volatility is a nonlinear transform of a hidden linear autoregressive process where the hidden volatility process, $x_t = \log \sigma_t^2$, follows a first order autoregression,

$$x_t = \phi x_{t-1} + w_t, \quad (6.197a)$$

$$r_t = \beta \exp(x_t/2) \varepsilon_t, \quad (6.197b)$$

where $w_t \sim \text{iid } N(0, \sigma_w^2)$ and ε_t is iid noise having finite moments. The error processes w_t and ε_t are assumed to be mutually independent and $|\phi| < 1$. As w_t is normally distributed, x_t is also normally distributed. All moments of ε_t exist, so that all moments of r_t in (6.197) exist as well. Assuming that $x_0 \sim N(0, \sigma_w^2/(1 - \phi^2))$ [the stationary distribution] the kurtosis^{6.6} of r_t is given by

$$\kappa_4(r_t) = \kappa_4(\varepsilon_t) \exp(\sigma_x^2), \quad (6.198)$$

where $\sigma_x^2 = \sigma_w^2/(1 - \phi^2)$ is the (stationary) variance of x_t . Thus $\kappa_4(r_t) > \kappa_4(\varepsilon_t)$, so that if $\varepsilon_t \sim \text{iid } N(0, 1)$, the distribution of r_t is leptokurtic. The autocorrelation function of $\{r_t^{2m}; t = 1, 2, \dots\}$ for any integer m is given by (see [Problem 6.29](#))

$$\text{corr}(r_{t+h}^{2m}, r_t^{2m}) = \frac{\exp(m^2 \sigma_x^2 \phi^h) - 1}{\kappa_{4m}(\varepsilon_t) \exp(m^2 \sigma_x^2) - 1}. \quad (6.199)$$

The decay rate of the autocorrelation function is faster than exponential at small time lags and then stabilizes to ϕ for large lags.

Sometimes it is easier to work with the linear form of the model where we define

$$y_t = \log r_t^2 \quad \text{and} \quad v_t = \log \varepsilon_t^2,$$

^{6.6} For an integer m and a random variable U , $\kappa_m(U) := E[|U|^m]/(E[|U|^2])^{m/2}$. Typically, κ_3 is called *skewness* and κ_4 is called *kurtosis*.

in which case we may write

$$y_t = \alpha + x_t + v_t. \quad (6.200)$$

A constant is usually needed in either the state equation or the observation equation (but not typically both), so we write the state equation as

$$x_t = \phi_0 + \phi_1 x_{t-1} + w_t, \quad (6.201)$$

where w_t is white Gaussian noise with variance σ_w^2 . The constant ϕ_0 is sometimes referred to as the *leverage effect*. Together, (6.200) and (6.201) make up the stochastic volatility model due to Taylor (1982).

If ε_t^2 had a log-normal distribution, (6.200)–(6.201) would form a Gaussian state-space model, and we could then use standard DLM results to fit the model to data. Unfortunately, that assumption does not seem to work well. Instead, one often keeps the ARCH normality assumption on $\varepsilon_t \sim \text{iid } N(0, 1)$, in which case, $v_t = \log \varepsilon_t^2$ is distributed as the log of a chi-squared random variable with one degree of freedom. This density is given by

$$f(v) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(e^v - v)\right\} \quad -\infty < v < \infty. \quad (6.202)$$

The mean of the distribution is $-(\gamma + \log 2)$, where $\gamma \approx 0.5772$ is Euler's constant, and the variance of the distribution is $\pi^2/2$. It is a highly skewed density (see Figure 6.18) but it is not flexible because there are no free parameters to be estimated.

Various approaches to the fitting of stochastic volatility models have been examined; these methods include a wide range of assumptions on the observational noise process. A good summary of the proposed techniques, both Bayesian (via MCMC) and non-Bayesian approaches (such as quasi-maximum likelihood estimation and the EM algorithm), can be found in Jacquier et al. (1994), and Shephard (1996). Simulation methods for classical inference applied to stochastic volatility models are discussed in Danielson (1994) and Sandmann and Koopman (1998).

Kim, Shephard and Chib (1998) proposed modeling the log of a chi-squared random variable by a mixture of seven normals to approximate the first four moments of the observational error distribution; the mixture is fixed and no additional model parameters are added by using this technique. The basic model assumption that ε_t is Gaussian is unrealistic for most applications. In an effort to keep matters simple but more general (in that we allow the observational error dynamics to depend on parameters that will be fitted), our method of fitting stochastic volatility models is to retain the Gaussian state equation (6.201), but to write the observation equation, as

$$y_t = \alpha + x_t + \eta_t, \quad (6.203)$$

where η_t is white noise, whose distribution is a mixture of two normals, one centered at zero. In particular, we write

$$\eta_t = I_t z_{t0} + (1 - I_t) z_{t1}, \quad (6.204)$$

where I_t is an iid Bernoulli process, $\Pr\{I_t = 0\} = \pi_0$, $\Pr\{I_t = 1\} = \pi_1$ ($\pi_0 + \pi_1 = 1$), $z_{t0} \sim \text{iid } N(0, \sigma_0^2)$, and $z_{t1} \sim \text{iid } N(\mu_1, \sigma_1^2)$.

The advantage to this model is that it is easy to fit because it uses normality. In fact, the model equations (6.201) and (6.203)–(6.204) are similar to those presented in Peña and Guttman (1988), who used the idea to obtain a robust Kalman filter, and, as previously mentioned, in Kim et al. (1998). The material presented in Section 6.10 applies here, and in particular, the filtering equations for this model are

$$x_{t+1}^t = \phi_0 + \phi_1 x_t^{t-1} + \sum_{j=0}^1 \pi_{tj} K_{tj} \epsilon_{tj}, \quad (6.205)$$

$$P_{t+1}^t = \phi_1^2 P_t^{t-1} + \sigma_w^2 - \sum_{j=0}^1 \pi_{tj} K_{tj}^2 \Sigma_{tj}, \quad (6.206)$$

$$\epsilon_{t0} = y_t - \alpha - x_t^{t-1}, \quad \epsilon_{t1} = y_t - \alpha - x_t^{t-1} - \mu_1, \quad (6.207)$$

$$\Sigma_{t0} = P_t^{t-1} + \sigma_0^2, \quad \Sigma_{t1} = P_t^{t-1} + \sigma_1^2, \quad (6.208)$$

$$K_{t0} = \phi_1 P_t^{t-1} / \Sigma_{t0}, \quad K_{t1} = \phi_1 P_t^{t-1} / \Sigma_{t1}. \quad (6.209)$$

To complete the filtering, we must be able to assess the probabilities $\pi_{t1} = \Pr(I_t = 1 | y_{1:t})$, for $t = 1, \dots, n$; of course, $\pi_{t0} = 1 - \pi_{t1}$. Let $p_j(t | t-1)$ denote the conditional density of y_t given the past $y_{1:t-1}$, and $I_t = j$ for $j = 0, 1$. Then,

$$\pi_{t1} = \frac{\pi_1 p_1(t | t-1)}{\pi_0 p_0(t | t-1) + \pi_1 p_1(t | t-1)}, \quad (6.210)$$

where we assume the distribution π_j , for $j = 0, 1$ has been specified *a priori*. If the investigator has no reason to prefer one state over another the choice of uniform priors, $\pi_1 = 1/2$, will suffice. Unfortunately, it is computationally difficult to obtain the exact values of $p_j(t | t-1)$; although we can give an explicit expression of $p_j(t | t-1)$, the actual computation of the conditional density is prohibitive. A viable approximation, however, is to choose $p_j(t | t-1)$ to be the normal density, $N(x_t^{t-1} + \mu_j, \Sigma_{tj})$, for $j = 0, 1$ and $\mu_0 = 0$; see Section 6.10 for details.

The innovations filter given in (6.205)–(6.210) can be derived from the Kalman filter by a simple conditioning argument; e.g., to derive (6.205), write

$$\begin{aligned} E(x_{t+1} | y_{1:t}) &= \sum_{j=0}^1 E(x_{t+1} | y_{1:t}, I_t = j) \Pr(I_t = j | y_{1:t}) \\ &= \sum_{j=0}^1 \left(\phi_0 + \phi_1 x_t^{t-1} + K_{tj} \epsilon_{tj} \right) \pi_{tj} \\ &= \phi_0 + \phi_1 x_t^{t-1} + \sum_{j=0}^1 \pi_{tj} K_{tj} \epsilon_{tj}. \end{aligned}$$

Estimation of the parameters, $\Theta = (\phi_0, \phi_1, \sigma_0^2, \mu_1, \sigma_1^2, \sigma_w^2)'$, is accomplished via MLE based on the likelihood given by

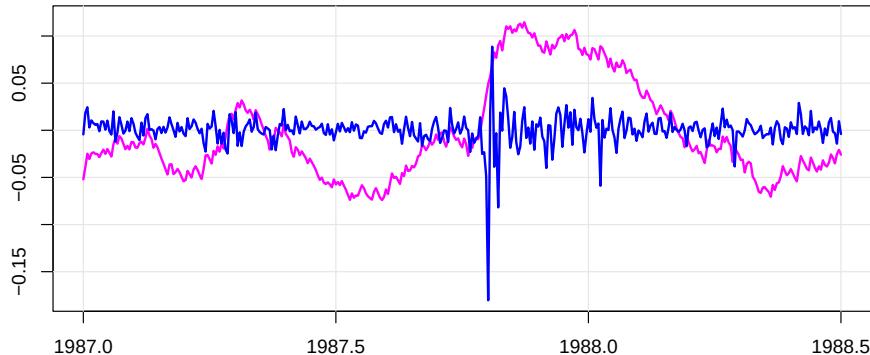


Fig. 6.17. Approximately four hundred observations of r_t , the daily returns of the NYSE surrounding the crash of October 19, 1987. Also displayed is the corresponding one-step-ahead predicted log volatility, \hat{x}_t^{t-1} where $x_t = \log \sigma_t^2$, scaled by .1 to fit on the plot.

$$\ln L_Y(\Theta) = \sum_{t=1}^n \ln \left(\sum_{j=0}^1 \pi_j f_j(t \mid t-1) \right), \quad (6.211)$$

where the density $p_j(t \mid t-1)$ is approximated by the normal density, $N(x_t^{t-1} + \mu_j, \sigma_j^2)$, previously mentioned. We may consider maximizing (6.211) directly as a function of the parameters Θ using a Newton method, or we may consider applying the EM algorithm to the complete data likelihood.

Example 6.23 Analysis of the New York Stock Exchange Returns

Figure 6.17 shows the returns, r_t , for about 400 of the 2000 trading days of the NYSE. Model (6.201) and (6.203)–(6.204), with π_1 fixed at .5, was fit to the data using a quasi-Newton–Raphson method to maximize (6.211). The results are given in Table 6.4. Figure 6.18 compares the density of the log of a χ_1^2 with the fitted normal mixture; we note the data indicate a substantial amount of probability in the upper tail that the log- χ_1^2 distribution misses.

Finally, Figure 6.17 also displays the one-step-ahead predicted log volatility, \hat{x}_t^{t-1} where $x_t = \log \sigma_t^2$, surrounding the crash of October 19, 1987. The analysis indicates that ϕ_0 is not needed. The R code when ϕ_0 is included in the model is as follows.

```

y = log(nyse^2)
num = length(y)
# Initial Parameters
phi0 = 0; phi1 = .95; sQ = .2; alpha = mean(y)
sR0 = 1; mu1 = -3; sR1 = 2
init.par = c(phi0, phi1, sQ, alpha, sR0, mu1, sR1)
# Innovations Likelihood
Linn = function(para){
  phi0 = para[1]; phi1 = para[2]; sQ = para[3]; alpha = para[4]
  sR0 = para[5]; mu1 = para[6]; sR1 = para[7]
  sv = SVfilter(num, y, phi0, phi1, sQ, alpha, sR0, mu1, sR1)
}

```

Table 6.4. Estimation Results for the NYSE Fit

Parameter	Estimate	Estimated Standard Error
ϕ_0	-.006	.016 [†]
ϕ_1	.988	.007
σ_w	.091	.027
α	-9.613	1.269
σ_0	1.220	.065
μ_1	-2.292	.205
σ_1	2.683	.105

[†] not significant

```

return(sv$like)      }
# Estimation
(est = optim(init.par, Linn, NULL, method='BFGS', hessian=TRUE,
            control=list(trace=1,REPORT=1)))
SE = sqrt(diag(solve(est$hessian)))
u = cbind(estimate=est$par, SE)
rownames(u)=c('phi0','phi1','sQ','alpha','sigv0','mu1','sigv1'); u
# Graphics (need filters at the estimated parameters)
phi0 = est$par[1]; phi1 = est$par[2]; sQ = est$par[3]; alpha = est$par[4]
sR0 = est$par[5]; mu1 = est$par[6]; sR1 = est$par[7]
sv = SVfilter(num,y,phi0,phi1,sQ,alpha,sR0,mu1,sR1)
# densities plot (f is chi-sq, fm is fitted mixture)
x = seq(-15,6,by=.01)
f = exp(-.5*(exp(x)-x))/(sqrt(2*pi))
f0 = exp(-.5*(x^2)/sR0^2)/(sR0*sqrt(2*pi))
f1 = exp(-.5*(x-mu1)^2/sR1^2)/(sR1*sqrt(2*pi))
fm = (f0+f1)/2
plot(x, f, type='l'); lines(x, fm, lty=2, lwd=2)
dev.new(); Time=701:1100
plot (Time, nyse[Time], type='l', col=4, lwd=2, ylab='', xlab='',
      ylim=c(-.18,.12))
lines(Time, sv$xp[Time]/10, lwd=2, col=6)

```

It is possible to use the bootstrap procedure described in Section 6.7 for the stochastic volatility model, with some minor changes. The following procedure was described in Stoffer and Wall (2004). We develop a vector first-order equation, as was done in (6.123). First, using (6.207), and noting that $y_t = \pi_{t0}y_t + \pi_{t1}y_t$, we may write

$$y_t = \alpha + x_t^{t-1} + \pi_{t0}\epsilon_{t0} + \pi_{t1}(\epsilon_{t1} + \mu_1). \quad (6.212)$$

Consider the standardized innovations

$$\epsilon_{tj} = \Sigma_{tj}^{-1/2} \epsilon_{tj}, \quad j = 0, 1, \quad (6.213)$$

and define the 2×1 vector

$$\epsilon_t = \begin{bmatrix} \epsilon_{t0} \\ \epsilon_{t1} \end{bmatrix}.$$

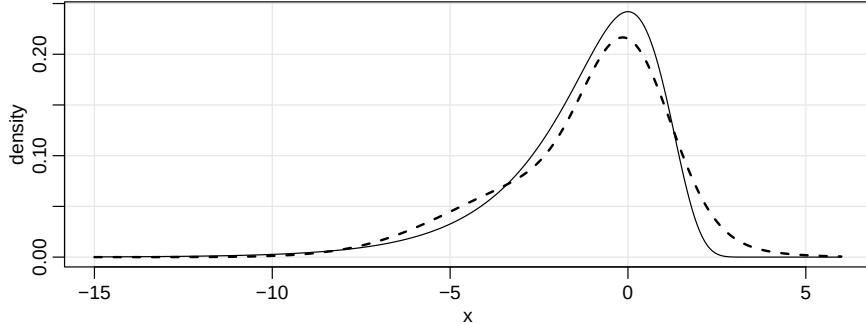


Fig. 6.18. Density of the log of a χ^2_1 as given by (6.202) (solid line) and the fitted normal mixture (dashed line) from Example 6.23.

Also, define the 2×1 vector

$$\xi_t = \begin{bmatrix} x_{t+1}^t \\ y_t \end{bmatrix}.$$

Combining (6.205) and (6.212) results in a vector first-order equation for ξ_t given by

$$\xi_t = F\xi_{t-1} + G_t + H_t e_t, \quad (6.214)$$

where

$$F = \begin{bmatrix} \phi_1 & 0 \\ 1 & 0 \end{bmatrix}, \quad G_t = \begin{bmatrix} \phi_0 \\ \alpha + \pi_{t1}\mu_1 \end{bmatrix}, \quad H_t = \begin{bmatrix} \pi_{t0}K_{t0}\Sigma_{t0}^{1/2} & \pi_{t1}K_{t1}\Sigma_{t1}^{1/2} \\ \pi_{t0}\Sigma_{t0}^{1/2} & \pi_{t1}\Sigma_{t1}^{1/2} \end{bmatrix}.$$

Hence, the steps in bootstrapping for this case are the same as steps (i) through (v) described in Section 6.7, but with (6.123) replaced by the following first-order equation:

$$\xi_t^* = F(\hat{\Theta})\xi_{t-1}^* + G_t(\hat{\Theta}; \hat{\pi}_{t1}) + H_t(\hat{\Theta}; \hat{\pi}_{t1})e_t^*, \quad (6.215)$$

where $\hat{\Theta} = \{\hat{\phi}_0, \hat{\phi}_1, \hat{\sigma}_0^2, \hat{\alpha}, \hat{\mu}_1, \hat{\sigma}_1^2, \hat{\sigma}_w^2\}$ is the MLE of Θ , and $\hat{\pi}_{t1}$ is estimated via (6.210), replacing $p_1(t | t-1)$ and $p_0(t | t-1)$ by their respective estimated normal densities ($\hat{\pi}_{t0} = 1 - \hat{\pi}_{t1}$).

Example 6.24 Analysis of the U.S. GNP Growth Rate

In Example 5.4, we fit an ARCH model to the U.S. GNP growth rate. In this example, we will fit a stochastic volatility model to the residuals from the AR(1) fit on the growth rate (see Example 3.39). Figure 6.19 shows the log of the squared residuals, say y_t , from the fit on the U.S. GNP series. The stochastic volatility model (6.200)–(6.204) was then fit to y_t . Table 6.5 shows the MLEs of the model parameters along with their asymptotic SEs assuming the model is correct. Also displayed in Table 6.5 are the SEs of $B = 500$ bootstrapped samples. There is little agreement between most of the asymptotic values and the bootstrapped values. The interest here, however, is not so much in the SEs, but in the actual sampling distribution

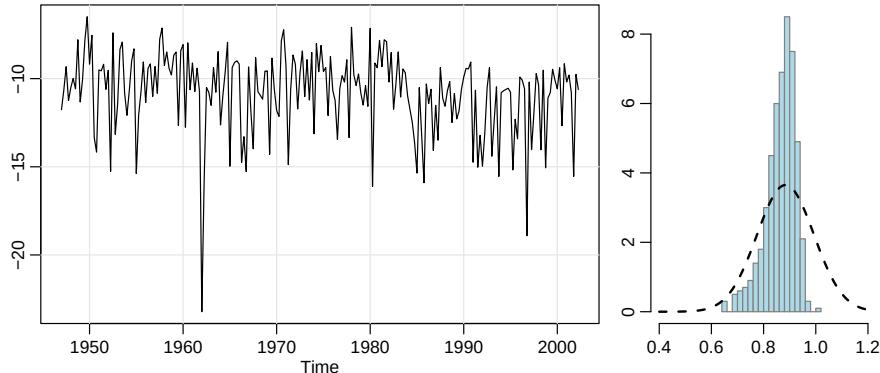


Fig. 6.19. Results for Example 6.24: Log of the squared residuals from an AR(1) fit on GNP growth rate. Bootstrap histogram and asymptotic distribution of $\hat{\phi}_1$.

of the estimates. For example, Figure 6.19 compares the bootstrap histogram and asymptotic normal distribution of $\hat{\phi}_1$. In this case, the bootstrap distribution exhibits positive kurtosis and skewness which is missed by the assumption of asymptotic normality.

The R code for this example is as follows. We held ϕ_0 at 0 for this analysis because it was not significantly different from 0 in an initial analysis.

```

n.boot = 500 # number of bootstrap replicates
tol = sqrt(.Machine$double.eps) # convergence tolerance
gnpgr = diff(log(gnp))
fit = arima(gnpgr, order=c(1,0,0))
y = as.matrix(log(resid(fit)^2))
num = length(y)
plot.ts(y, ylab='')
# Initial Parameters
phi1 = .9; sQ = .5; alpha = mean(y); sR0 = 1; mu1 = -3; sR1 = 2.5
init.par = c(phi1, sQ, alpha, sR0, mu1, sR1)
# Innovations Likelihood
Linn = function(para, y.data){
  phi1 = para[1]; sQ = para[2]; alpha = para[3]
  sR0 = para[4]; mu1 = para[5]; sR1 = para[6]
  sv = SVfilter(num, y.data, 0, phi1, sQ, alpha, sR0, mu1, sR1)
  return(sv$like)
}
# Estimation
est = optim(init.par, Linn, NULL, y.data=y, method='BFGS', hessian=TRUE,
            control=list(trace=1,REPORT=1))
SE = sqrt(diag(solve(est$hessian)))
u = rbind(estimate=est$par, SE)
colnames(u)=c('phi1','sQ','alpha','sig0','mu1','sig1')
phi1    sQ   alpha   sig0   mu1   sig1
estimates 0.884 0.381 -9.654 0.835 -2.350 2.453
SE       0.109 0.221  0.343 0.204  0.495 0.293
# Bootstrap
para.star = matrix(0, n.boot, 6) # to store parameter estimates
for (jb in 1:n.boot){

```

Table 6.5. Estimates and Standard Errors for GNP Example

Parameter	MLE	Asymptotic SE	Bootstrap [†] SE
ϕ_1	0.884	0.109	0.057
σ_w	0.381	0.221	0.324
α	-9.654	0.343	1.529
σ_0	0.835	0.204	0.527
μ_1	-2.350	0.495	0.410
σ_1	2.453	0.293	0.375

[†] Based on 500 bootstrapped samples.

```

cat('iteration:', jb, '\n')
phi1 = est$par[1]; sQ = est$par[2]; alpha = est$par[3]
sR0 = est$par[4]; mu1 = est$par[5]; sR1 = est$par[6]
Q = sQ^2; R0 = sR0^2; R1 = sR1^2
sv = SVfilter(num, y, 0, phi1, sQ, alpha, sR0, mu1, sR1)
sig0 = sv$Pp+R0; sig1 = sv$Pp+R1;
K0 = sv$Pp/sig0; K1 = sv$Pp/sig1
inn0 = y-sv$xp-alpha; inn1 = y-sv$xp-mu1-alpha
den1 = (1/sqrt(sig1))*exp(-.5*inn1^2/sig1)
den0 = (1/sqrt(sig0))*exp(-.5*inn0^2/sig0)
fpi1 = den1/(den0+den1)
# start resampling at t=4
e0 = inn0/sqrt(sig0); e1 = inn1/sqrt(sig1)
indx = sample(4:num, replace=TRUE)
sinn = cbind(c(e0[1:3], e0[indx]), c(e1[1:3], e1[indx]))
eF = matrix(c(phi1, 1, 0, 0), 2, 2)
xi = cbind(sv$xp,y) # initialize
for (i in 4:num){ # generate boot sample
  G = matrix(c(0, alpha+fpi1[i]*mu1), 2, 1)
  h21 = (1-fpi1[i])*sqrt(sig0[i]); h11 = h21*K0[i]
  h22 = fpi1[i]*sqrt(sig1[i]); h12 = h22*K1[i]
  H = matrix(c(h11,h21,h12,h22),2,2)
  xi[i,] = t(eF%*%as.matrix(xi[i-1,],2) + G + H%*%as.matrix(sinn[i,],2))}
# Estimates from boot data
y.star = xi[,2]
phi1=.9; sQ=.5; alpha=mean(y.star); sR0=1; mu1=-3; sR1=2.5
init.par = c(phi1, sQ, alpha, sR0, mu1, sR1) # same as for data
est.star = optim(init.par, Linn, NULL, y.data=y.star, method='BFGS',
control=list(reltol=tol))
para.star[jb,] = cbind(est.star$par[1], abs(est.star$par[2]),
est.star$par[3], abs(est.star$par[4]), est.star$par[5],
abs(est.star$par[6])) }
# Some summary statistics and graphics
rmse = rep(NA,6) # SEs from the bootstrap
for(i in 1:6){
  rmse[i] = sqrt(sum((para.star[,i]-est$par[i])^2)/n.boot)
  cat(i, rmse[i],'\n') }
dev.new(); phi = para.star[,1]
hist(phi, 15, prob=TRUE, main='', xlim=c(.4,1.2), xlab='')
xx = seq(.4, 1.2, by=.01)
lines(xx, dnorm(xx, mean=u[1,1], sd=u[2,1]), lty='dashed', lwd=2)

```

6.12 Bayesian Analysis of State Space Models

We now consider some Bayesian approaches to fitting linear Gaussian state space models via Markov chain Monte Carlo (MCMC) methods. We assume that the model is given by (6.1)–(6.2); inputs are allowed in the model, but we do not display them for the sake of brevity. In this case, Frühwirth-Schnatter (1994) and Carter and Kohn (1994) established the MCMC procedure that we will discuss here. A comprehensive text that we highly recommend for this case is Petris et al. (2009) and the corresponding R package `dlm`. For nonlinear and non-Gaussian models, the reader is referred to Douc, Moulines, & Stoffer (2014). As in previous sections, we have n observations denoted by $y_{1:n} = \{y_1, \dots, y_n\}$, whereas the states are denoted as $x_{0:n} = \{x_0, x_1, \dots, x_n\}$, with x_0 being the initial state.

MCMC methods refer to Monte Carlo integration methods that use a Markovian updating scheme to sample from intractable posterior distributions. The most common MCMC method is the Gibbs sampler, which is essentially a modification of the Metropolis algorithm (Metropolis et al., 1953) developed by Hastings (1970) in the statistical setting and by Geman and Geman (1984) in the context of image restoration. Later, Tanner and Wong (1987) used the ideas in their substitution sampling approach, and Gelfand and Smith (1990) developed the Gibbs sampler for a wide class of parametric models. The basic strategy is to use conditional distributions to set up a Markov chain to obtain samples from a joint distribution. The following simple case demonstrates this idea.

Example 6.25 Gibbs Sampling for the Bivariate Normal

Suppose we wish to obtain samples from a bivariate normal distribution,

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right],$$

where $|\rho| < 1$, but we can only generate samples from a univariate normal.

- The univariate conditionals are [see (B.9)–(B.10)]

$$(X | Y = y) \sim N(\rho y, 1 - \rho^2) \quad \text{and} \quad (Y | X = x) \sim N(\rho x, 1 - \rho^2),$$

and we can simulate from these distributions.

- Construct a Markov chain: Pick $X^{(0)} = x_0$, and then iterate the process $X^{(0)} = x_0 \mapsto Y^{(0)} \mapsto X^{(1)} \mapsto Y^{(1)} \mapsto \dots \mapsto X^{(k)} \mapsto Y^{(k)} \mapsto \dots$, where

$$\begin{aligned} (Y^{(k)} | X^{(k)} = x_k) &\sim N(\rho x_k, 1 - \rho^2) \\ (X^{(k)} | Y^{(k-1)} = y_{k-1}) &\sim N(\rho y_{k-1}, 1 - \rho^2). \end{aligned}$$

- The joint distribution of $(X^{(k)}, Y^{(k)})$ is (see Problem 3.2)

$$\begin{pmatrix} X^{(k)} \\ Y^{(k)} \end{pmatrix} \sim N \left[\begin{pmatrix} \rho^{2k} x_0 \\ \rho^{2k+1} x_0 \end{pmatrix}, \begin{pmatrix} 1 - \rho^{4k} & \rho(1 - \rho^{4k}) \\ \rho(1 - \rho^{4k}) & 1 - \rho^{4k+2} \end{pmatrix} \right].$$

- Thus, for any starting value, x_0 , $(X^{(k)}, Y^{(k)}) \rightarrow_d (X, Y)$ as $k \rightarrow \infty$; the speed depends on ρ . Then one would run the chain and throw away the initial n_0 sampled values (burnin) and retain the rest.

For state space models, the main objective is to obtain the posterior density of the parameters $p(\Theta | y_{1:n})$ or $p(x_{0:n} | y_{1:n})$ if the states are meaningful. For example, the states do not have any meaning for an ARMA model, but they are important for a stochastic volatility model. It is generally easier to get samples from the full posterior $p(\Theta, x_{0:n} | y_{1:n})$ and then marginalize (“average”) to obtain $p(\Theta | y_{1:n})$ or $p(x_{0:n} | y_{1:n})$. As previously mentioned, the most popular method is to run a full Gibbs sampler, alternating between sampling model parameters and latent state sequences from their respective full conditional distributions.

Procedure 6.1 Gibbs Sampler for State Space Models

- (i) Draw $\Theta' \sim p(\Theta | x_{0:n}, y_{1:n})$
- (ii) Draw $x'_{0:n} \sim p(x_{0:n} | \Theta', y_{1:n})$

Procedure 6.1-(i) is generally much easier because it conditions on the complete data $\{x_{0:n}, y_{1:n}\}$, which we saw in [Section 6.3](#) can simplify the problem. **Procedure 6.1-(ii)** amounts to sampling from the joint smoothing distribution of the latent state sequence and is generally difficult. For linear Gaussian models, however, both parts of **Procedure 6.1** are relatively easy to perform.

To accomplish **Procedure 6.1-(i)**, note that

$$p(\Theta | x_{0:n}, y_{1:n}) \propto \pi(\Theta) p(x_0 | \Theta) \prod_{t=1}^n p(x_t | x_{t-1}, \Theta) p(y_t | x_t, \Theta) \quad (6.216)$$

where $\pi(\Theta)$ is the prior on the parameters. The prior often depends on “hyperparameters” that add another level to the hierarchy. For simplicity, these hyperparameters are assumed to be known. The parameters are typically conditionally independent with distributions from standard parametric families (at least as long as the prior distribution is conjugate relative to the Bayesian model specification). For non-conjugate models, one option is to replace **Procedure 6.1-(i)** with a Metropolis-Hastings step, which is feasible since the complete data density $p(\Theta, x_{0:n}, y_{1:n})$ can be evaluated pointwise.

For example, in the univariate model

$$x_t = \phi x_{t-1} + w_t \quad \text{and} \quad y_t = x_t + v_t$$

where $w_t \sim \text{iid } N(0, \sigma_w^2)$ independent of $v_t \sim \text{iid } N(0, \sigma_v^2)$, we can use the normal and inverse gamma (IG) distributions for priors. In this case, the priors on the variance components are chosen from a conjugate family, that is, $\sigma_w^2 \sim \text{IG}(a_0/2, b_0/2)$ independent of $\sigma_v^2 \sim \text{IG}(c_0/2, d_0/2)$, where IG denotes the inverse (reciprocal) gamma distribution. Then, for example, if the prior on ϕ is Gaussian, $\phi \sim N(\mu_\phi, \sigma_\phi^2)$, then $\phi | \sigma_w, x_{0:n}, y_{1:n} \sim N(Bb, B)$, where

$$B^{-1} = \frac{1}{\sigma_\phi^2} + \frac{1}{\sigma_w^2} \sum_{t=1}^n x_{t-1}^2, \quad b = \frac{\mu_\phi}{\sigma_\phi^2} + \frac{1}{\sigma_w^2} \sum_{t=1}^n x_t x_{t-1}.$$

and

$$\begin{aligned}\sigma_w^2 | \phi, x_{0:n}, y_{1:n} &\sim \text{IG}\left(\frac{1}{2}(a_0 + n), \frac{1}{2}\{b_0 + \sum_{t=1}^n [x_t - \phi x_{t-1}]^2\}\right); \\ \sigma_v^2 | x_{0:n}, y_{1:n} &\sim \text{IG}\left(\frac{1}{2}(c_0 + n), \frac{1}{2}\{c_0 + \sum_{t=1}^n [y_t - x_t]^2\}\right).\end{aligned}$$

For **Procedure 6.1-(ii)**, the goal is to sample the entire set of state vectors, $x_{0:n}$, from the posterior density $p(x_{0:n} | \Theta, y_{1:n})$, where Θ is a fixed set of parameters obtained from the previous step. We will write the posterior as $p_\Theta(x_{0:n} | y_{1:n})$ to save space. Because of the Markov structure, we can write,

$$p_\Theta(x_{0:n} | y_{1:n}) = p_\Theta(x_n | y_{1:n}) p_\Theta(x_{n-1} | x_n, y_{1:n-1}) \cdots p_\Theta(x_0 | x_1). \quad (6.217)$$

In view of (6.217), it is possible to sample the entire set of state vectors, $x_{0:n}$, by sequentially simulating the individual states backward. This process yields a simulation method that Frühwirth-Schnatter (1994) called the forward-filtering, backward-sampling (FFBS) algorithm. From (6.217), we see that we must obtain the densities

$$p_\Theta(x_t | x_{t+1}, y_{1:t}) \propto p_\Theta(x_t | y_{1:t}) p_\Theta(x_{t+1} | x_t).$$

In particular, we know that $x_t | y_{1:t} \sim N_p^\Theta(x_t^t, P_t^t)$ and $x_{t+1} | x_t \sim N_p^\Theta(\Phi x_t, Q)$. And because the processes are Gaussian, we need only obtain the conditional means and variances, say, $m_t = E_\Theta(x_t | y_{1:t}, x_{t+1})$ and $V_t = \text{var}_\Theta(x_t | y_{1:t}, x_{t+1})$. In particular,

$$m_t = x_t^t + J_t(x_{t+1} - x_{t+1}^t) \quad \text{and} \quad V_t = P_t^t - J_t P_{t+1}^t J_t', \quad (6.218)$$

for $t = n-1, n-2, \dots, 0$, where J_t is defined in (6.47). We note that m_t has already been derived in (6.48). To derive m_t and V_t using standard normal theory, use a strategy similar to the derivation of the filter in **Property 6.1**. That is,

$$\begin{pmatrix} x_t \\ x_{t+1} \end{pmatrix} | y_{1:t} \sim N \left(\begin{bmatrix} x_t^t \\ x_{t+1}^t \end{bmatrix}, \begin{bmatrix} P_t^t & P_t^t \Phi' \\ \Phi P_t^t & P_{t+1}^t \end{bmatrix} \right);$$

now use (B.9), (B.10), and the definition of J_t in (6.47). Also, recall the proof of **Property 6.3** wherein we noted the off-diagonal $P_{t+1,t}^t = \Phi P_t^t$.

Hence, given Θ , the algorithm is to first sample x_n from a $N_p^\Theta(x_n^n, P_n^n)$, where x_n^n and P_n^n are obtained from the Kalman filter, **Property 6.1**, and then sample x_t from a $N_p^\Theta(m_t, V_t)$, for $t = n-1, n-2, \dots, 0$, where the conditioning value of x_{t+1} is the value previously sampled.

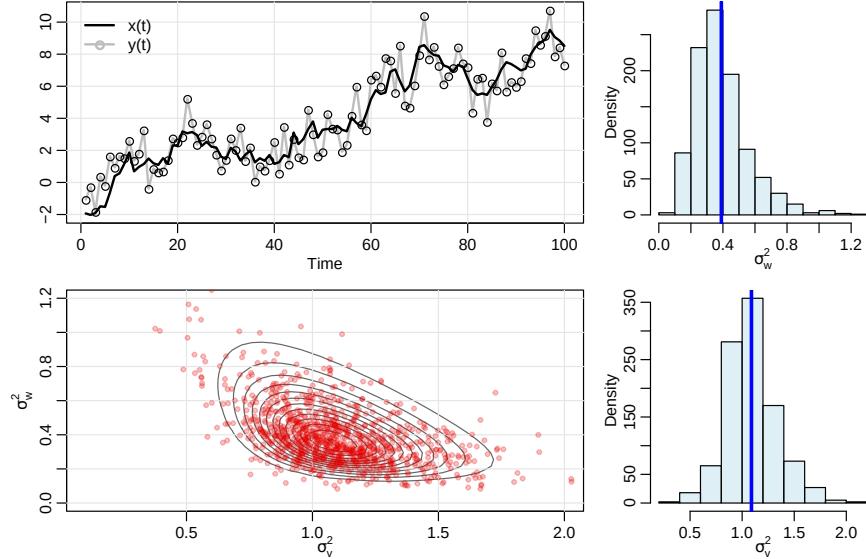


Fig. 6.20. Display for [Example 6.26](#): Left: Generated states, x_t and data y_t . Contours of the likelihood (solid line) of the data and sampled posterior values as points. Right : Marginal sampled posteriors and posterior means (vertical lines) of each variance component. The true values are $\sigma_w^2 = .5$ and $\sigma_v^2 = 1$.

Example 6.26 Local Level Model

In this example, we consider the local level model previously discussed in [Example 6.4](#). Here, we consider the model

$$y_t = x_t + v_t \quad \text{and} \quad x_t = x_{t-1} + w_t$$

where $v_t \sim \text{iid } N(0, \sigma_v^2 = 1)$ independent of $w_t \sim \text{iid } N(0, \sigma_w^2 = .5)$. This is the univariate model we just discussed, but where $\phi = 1$. In this case, we used IG priors for each of the variance components.

For the prior distributions, all parameters (a_0, b_0, c_0, d_0) were set to .02. We generated 1010 samples, using the first 10 as burn-in. [Figure 6.20](#) displays the simulated data and states, the contours of the likelihood of the data, the sampled posterior values as points, and the marginal sampled posteriors of each variance component along with the posterior means. [Figure 6.21](#) compares the actual smoother x_t^n with the posterior mean of the sampled smoothed values. In addition, a pointwise 95% credible interval is displayed as a filled area.

The following code was used in this example.

```
##-- Notation --##
#           y(t) = x(t) + v(t);   v(t) ~ iid N(0, V)
#           x(t) = x(t-1) + w(t); w(t) ~ iid N(0, W)
# priors: x(0) ~ N(m0, C0); V ~ IG(a,b); W ~ IG(c,d)
# FFBS: x(t|t) ~ N(m,C); x(t|n) ~ N(mu,CC); x(t|t+1) ~ N(a,R)
##--
```

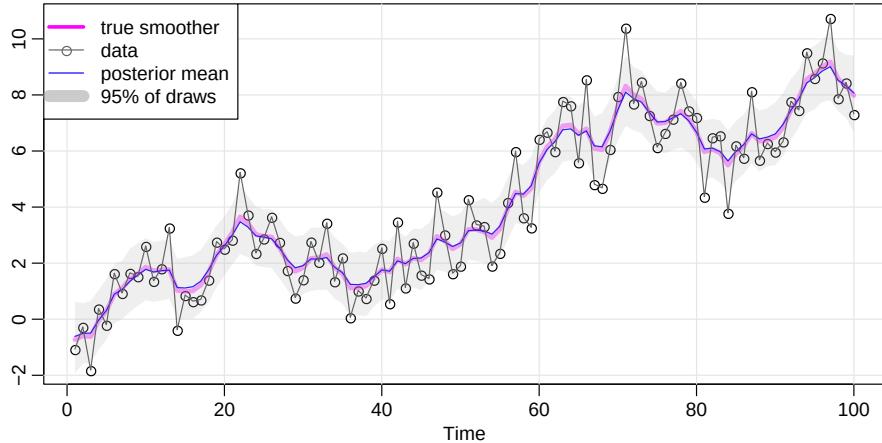


Fig. 6.21. Display for Example 6.26: True smoother, x_t^n , the data y_t , and the posterior mean of the sampled smoother values; the filled in area shows 2.5% to 97.5%-tiles of the draws.

```

ffbs = function(y,V,W,m0,C0){
  n = length(y); a = rep(0,n); R = rep(0,n)
  m = rep(0,n); C = rep(0,n); B = rep(0,n-1)
  H = rep(0,n-1); mm = rep(0,n); CC = rep(0,n)
  x = rep(0,n); llike = 0.0
  for (t in 1:n){
    if(t==1){a[1] = m0; R[1] = C0 + W
    }else{ a[t] = m[t-1]; R[t] = C[t-1] + W }
    f = a[t]
    Q = R[t] + V
    A = R[t]/Q
    m[t] = a[t]+A*(y[t]-f)
    C[t] = R[t]-Q*A**2
    B[t-1] = C[t-1]/R[t]
    H[t-1] = C[t-1]-R[t]*B[t-1]**2
    llike = llike + dnorm(y[t],f,sqrt(Q),log=TRUE) }
    mm[n] = m[n]; CC[n] = C[n]
    x[n] = rnorm(1,m[n],sqrt(C[n]))
    for (t in (n-1):1){
      mm[t] = m[t] + C[t]/R[t+1]*(mm[t+1]-a[t+1])
      CC[t] = C[t] - (C[t]^2)/(R[t+1]^2)*(R[t+1]-CC[t+1])
      x[t] = rnorm(1,m[t]+B[t]*(x[t+1]-a[t+1]),sqrt(H[t])) }
    return(list(x=x,m=m,C=C,mm=mm,CC=CC,llike=llike)) }
  # Simulate states and data
  set.seed(1); W = 0.5; V = 1.0
  n = 100; m0 = 0.0; C0 = 10.0; x0 = 0
  w = rnorm(n,0,sqrt(W))
  v = rnorm(n,0,sqrt(V))
  x = y = rep(0,n)
  x[1] = x0 + w[1]
  y[1] = x[1] + v[1]
  for (t in 2:n){
    x[t] = x[t-1] + w[t]
    y[t] = x[t] + v[t]
  }
}

```

```

y[t] = x[t] + v[t]    }
# actual smoother (for plotting)
ks = Ksmooth0(num=n, y, A=1, m0, C0, Phi=1, cQ=sqrt(W), cR=sqrt(V))
xsmooth = as.vector(ks$xs)
#
run = ffbs(y,V,W,m0,C0)
m = run$m; C = run$C; mm = run$mm
CC = run$CC; L1 = m-2*C; U1 = m+2*C
L2 = mm-2*CC; U2 = mm+2*CC
N = 50
Vs = seq(0.1,2,length=N)
Ws = seq(0.1,2,length=N)
likes = matrix(0,N,N)
for (i in 1:N){
  for (j in 1:N){
    V = Vs[i]
    W = Ws[j]
    run = ffbs(y,V,W,m0,C0)
    likes[i,j] = run$llike  }
  # Hyperparameters
  a = 0.01; b = 0.01; c = 0.01; d = 0.01
  # MCMC step
  set.seed(90210)
  burn = 10; M = 1000
  niter = burn + M
  V1 = V; W1 = W
  draws = NULL
  all_draws = NULL
  for (iter in 1:niter){
    run = ffbs(y,V1,W1,m0,C0)
    x = run$x
    V1 = 1/rgamma(1,a+n/2,b+sum((y-x)^2)/2)
    W1 = 1/rgamma(1,c+(n-1)/2,d+sum(diff(x)^2)/2)
    draws = rbind(draws,c(V1,W1,x))  }
  all_draws = draws[,1:2]
  q025 = function(x){quantile(x,0.025)}
  q975 = function(x){quantile(x,0.975)}
  draws = draws[(burn+1):(niter),]
  xs = draws[,3:(n+2)]
  lx = apply(xs,2,q025)
  mx = apply(xs,2,mean)
  ux = apply(xs,2,q975)
  ## plot of the data
  par(mfrow=c(2,2), mgp=c(1.6,.6,0), mar=c(3,3.2,1,1))
  ts.plot(ts(x), ts(y), ylab='', col=c(1,8), lwd=2)
  points(y)
  legend(0, 11, legend=c("x(t)","y(t)'), lty=1, col=c(1,8), lwd=2, bty="n",
         pch=c(-1,1))
  contour(Vs, Ws, exp(likes), xlab=expression(sigma[v]^2),
         ylab=expression(sigma[w]^2), drawlabels=FALSE, ylim=c(0,1.2))
  points(draws[,1:2], pch=16, col=rgb(.9,0,0,.3), cex=.7)
  hist(draws[,1], ylab="Density",main="", xlab=expression(sigma[v]^2))
  abline(v=mean(draws[,1]), col=3, lwd=3)
  hist(draws[,2],main="", ylab="Density", xlab=expression(sigma[w]^2))
  abline(v=mean(draws[,2]), col=3, lwd=3)
  ## plot states

```

```

par(mgp=c(1.6,.6,0), mar=c(2,1,.5,0)+.5)
plot(ts(mx), ylab='', type='n', ylim=c(min(y),max(y)))
grid(lty=2); points(y)
lines(xsmooth, lwd=4, col=rgb(1,0,1,alpha=.4))
lines(mx, col= 4)
xx=c(1:100, 100:1)
yy=c(lx, rev(ux))
polygon(xx, yy, border=NA, col= gray(.6,alpha=.2))
lines(y, col=gray(.4))
legend('topleft', c('true smoother', 'data', 'posterior mean', '95% of
draws'), lty=1, lwd=c(3,1,1,10), pch=c(-1,1,-1,-1), col=c(6,
gray(.4) ,4, gray(.6, alpha=.5)), bg='white' )

```

Next, we consider a more complicated model.

Example 6.27 Structural Model

Consider the Johnson & Johnson quarterly earnings per share series that was discussed in [Example 6.10](#). Recall that the model is

$$y_t = (1 \ 1 \ 0 \ 0) x_t + v_t,$$

$$x_t = \begin{pmatrix} T_t \\ S_t \\ S_{t-1} \\ S_{t-2} \end{pmatrix} = \begin{pmatrix} \phi & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} T_{t-1} \\ S_{t-1} \\ S_{t-2} \\ S_{t-3} \end{pmatrix} + \begin{pmatrix} w_{t1} \\ w_{t2} \\ 0 \\ 0 \end{pmatrix}$$

where $R = \sigma_v^2$ and

$$Q = \begin{pmatrix} \sigma_{w,11}^2 & 0 & 0 & 0 \\ 0 & \sigma_{w,22}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The parameters to be estimated are the transition parameter associated with the growth rate, $\phi > 1$, the observation noise variance, σ_v^2 , and the state noise variances associated with the trend and the seasonal components, $\sigma_{w,11}^2$ and $\sigma_{w,22}^2$, respectively.

In this case, sampling from $p(x_{0:n} | \Theta, y_{1:n})$ follows directly from [\(6.217\)](#)–[\(6.218\)](#). Next, we discuss how to sample from $p(\Theta | x_{0:n}, y_{1:n})$. For the transition parameter, write $\phi = 1 + \beta$, where $0 < \beta \ll 1$; recall that in [Example 6.10](#), ϕ was estimated to be 1.035, which indicated a growth rate, β , of 3.5%. Note that the trend component may be rewritten as

$$\nabla T_t = T_t - T_{t-1} = \beta T_{t-1} + w_{t1}.$$

Consequently, conditional on the states, the parameter β is the slope in the linear regression (through the origin) of ∇T_t on T_{t-1} , for $t = 1, \dots, n$, and w_{t1} is the error. As is typical, we put a Normal–Inverse Gamma (IG) prior on $(\beta, \sigma_{w,11}^2)$, i.e., $\beta | \sigma_{w,11}^2 \sim N(b_0, \sigma_{w,11}^2 B_0)$ and $\sigma_{w,11}^2 \sim IG(n_0/2, n_0 s_0^2/2)$, with known hyperparameters b_0, B_0, n_0, s_0^2 .

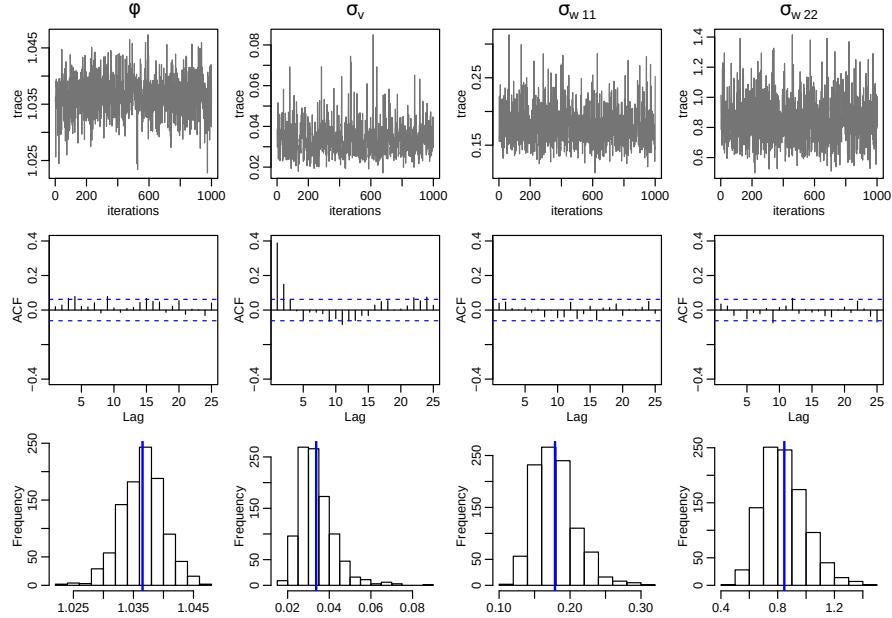


Fig. 6.22. Parameter estimation results for [Example 6.27](#). The top row displays the traces of 1000 draws after burn-in. The middle row displays the ACF of the traces. The sampled posteriors are displayed in the last row (the mean is marked by a solid vertical line).

We also used IG priors for the other two variance components, σ_v^2 and $\sigma_{w,22}^2$. In this case, if the prior $\sigma_v^2 \sim \text{IG}(n_0/2, n_0 s_0^2/2)$, then the posterior is

$$\sigma_v^2 \mid x_{0:n}, y_{1:n} \sim \text{IG}(n_v/2, n_v s_v^2/2),$$

where $n_v = n_0 + n$, and $n_v s_v^2 = n_0 s_0^2 + \sum_{t=1}^n (Y_t - T_t - S_t)^2$. Similarly, if the prior $\sigma_{w,22}^2 \sim \text{IG}(n_0/2, n_0 s_0^2/2)$, then the posterior is

$$\sigma_{w,22}^2 \mid x_{0:n}, y_{1:n} \sim \text{IG}(n_w/2, n_w s_w^2/2),$$

where $n_w = n_0 + (n - 3)$, and $n_w s_w^2 = n_0 s_0^2 + \sum_{t=1}^{n-3} (S_t - S_{t-1} - S_{t-2} - S_{t-3})^2$.

[Figure 6.22](#) displays the results of the posterior estimates of the parameters. The top row of the figure displays the traces of 1000 draws, after a burn-in of 100, with a step size of 10 (i.e., every 10th sampled value is retained). The middle row of the figure displays the ACF of the traces, and the sampled posteriors are displayed in the last row of the figure. The results of this analysis are comparable to the results obtained in [Example 6.10](#); the posterior mean and median for ϕ indicates a 3.7% growth rate in the Johnson & Johnson quarterly earnings over this time period.

[Figure 6.23](#) displays the smoothers of trend (T_t) and season ($T_t + S_t$) along with 99% credible intervals. Again, these results are comparable to the results obtained in [Example 6.10](#). The R code for this example is as follows:

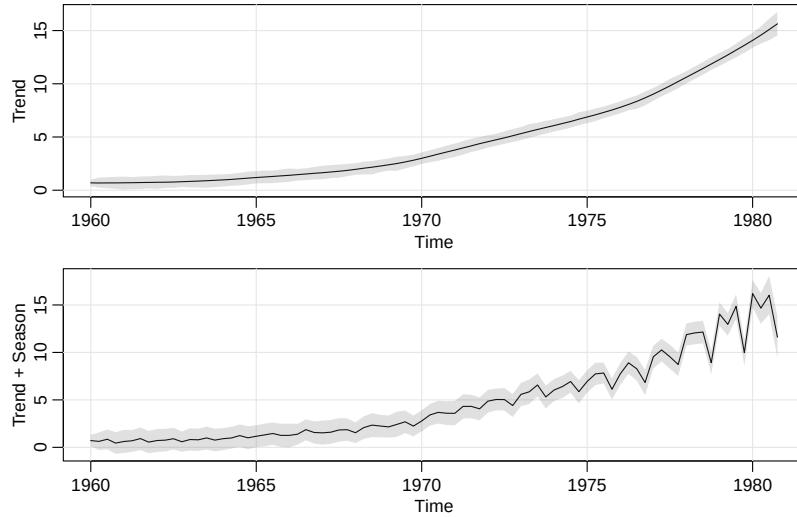


Fig. 6.23. Example 6.27 smoother estimates of trend (T_t) and trend plus season ($T_t + S_t$) along with corresponding 99% credible intervals.

```

library(plyr)    # used to view progress (install it if you don't have it)
y = jj
### setup - model and initial parameters
set.seed(90210)
n = length(y)
F = c(1,1,0,0)      # this is A
G = diag(0,4)        # G is Phi
G[1,1] = 1.03
G[2,] = c(0,-1,-1,-1); G[3,]=c(0,1,0,0); G[4,]=c(0,0,1,0)
a1 = rbind(.7,0,0,0) # this is mu0
R1 = diag(.04,4)     # this is Sigma0
V = .1
W11 = .1
W22 = .1
##-- FFBS --##
ffbs = function(y,F,G,V,W11,W22,a1,R1){
  n = length(y)
  Ws = diag(c(W11,W22,1,1)) # this is Q with 1s as a device only
  iW = diag(1/diag(Ws),4)
  a = matrix(0,n,4)          # this is m_t
  R = array(0,c(n,4,4))     # this is V_t
  m = matrix(0,n,4)
  C = array(0,c(n,4,4))
  a[1,] = a1[,1]
  R[1,,] = R1
  f = t(F)%*%a[1,]
  Q = t(F)%*%R[1,,]%*%F + V
  A = R[1,,]%*%F/Q[1,1]
  m[1,] = a[1,]+A%*%(y[1]-f)
  C[1,,] = R[1,,]-A%*%t(A)*Q[1,1]
}

```

```

for (t in 2:n){
  a[t,] = G%*%m[t-1,]
  R[t,,] = G%*%C[t-1,,]*%*%t(G) + Ws
  f      = t(F)%*%a[t,]
  Q      = t(F)%*%R[t,,]*%*%F + V
  A      = R[t,,]*%*%F/Q[1,1]
  m[t,] = a[t,] + A%*%(y[t]-f)
  C[t,,] = R[t,,] - A%*%t(A)*Q[1,1]      }
  xb     = matrix(0,n,4)
  xb[n,] = m[n,] + t(chol(C[n,,]))%*%rnorm(4)
  for (t in (n-1):1){
    iC = solve(C[t,,])
    CCC = solve(t(G)%*%iW%*%G + iC)
    mmm = CCC%*%(t(G)%*%iW%*%xb[t+1,] + iC%*%m[t,])
    xb[t,] = mmm + t(chol(CCC))%*%rnorm(4)  }
  return(xb)                                }

##-- Prior hyperparameters --#
# b0 = 0      # mean for beta = phi -1
# B0 = Inf    # var for beta (non-informative => use OLS for sampling beta)
n0 = 10      # use same for all- the prior is 1/Gamma(n0/2, n0*s20_/2)
s20v = .001   # for V
s20w = .05    # for Ws
##-- MCMC scheme --#
set.seed(90210)
burnin = 100
step   = 10
M      = 1000
niter  = burnin+step*M
pars   = matrix(0,niter,4)
xbs   = array(0,c(niter,n,4))
pr <- progress_text()                      # displays progress
pr$init(niter)
for (iter in 1:niter){
  xb = ffbs(y,F,G,V,W11,W22,a1,R1)
  u = xb[,1]
  yu = diff(u); xu = u[-n]      # for phihat and se(phihat)
  regu = lm(yu~0+xu)            # est of beta = phi-1
  phies = as.vector(coef(summary(regu)))[1:2] + c(1,0) # phi estimate and SE
  dft = df.residual(regu)
  G[1,1] = phies[1] + rt(1,dft)*phies[2]  # use a t
  V = 1/rgamma(1, (n0+n)/2, (n0*s20v/2) + sum((y-xb[,1]-xb[,2])^2)/2)
  W11 = 1/rgamma(1, (n0+n-1)/2, (n0*s20w/2) +
               sum((xb[-1,1]-phies[1])*xb[-n,1])^2)/2)
  W22 = 1/rgamma(1, (n0+ n-3)/2, (n0*s20w/2) + sum((xb[4:n,2] +
               xb[3:(n-1),2]+ xb[2:(n-2),2] +xb[1:(n-3),2])^2)/2)
  xbs[iter,,] = xb
  pars[iter,] = c(G[1,1], sqrt(V), sqrt(W11), sqrt(W22))
  pr$step()                                }

# Plot results
ind = seq(burnin+1,niter,by=step)
names= c(expression(phi), expression(sigma[v]), expression(sigma[w-11]),
        expression(sigma[w-22]))
dev.new(height=5)
par(mfcol=c(3,4), mar=c(2,2,.25,0)+.75, mgp=c(1.6,.6,0), oma=c(0,0,1,0))
for (i in 1:4){
  plot.ts(pars[ind,i],xlab="iterations", ylab="trace", main="")
}

```

```

mtext(names[i], side=3, line=.5, cex=1)
acf(pars[ind,i],main="", lag.max=25, xlim=c(1,25), ylim=c(-.4,.4))
hist(pars[ind,i],main="",xlab="")
abline(v=mean(pars[ind,i]), lwd=2, col=3) }
par(mfrow=c(2,1), mar=c(2,2,0,0)+.7, mgp=c(1.6,.6,0))
mxb = cbind(apply(xbs[ind,,1],2,mean), apply(xbs[,,2],2,mean))
lxb = cbind(apply(xbs[ind,,1],2,quantile,0.005),
            apply(xbs[ind,,2],2,quantile,0.005))
uxb = cbind(apply(xbs[ind,,1],2,quantile,0.995),
            apply(xbs[ind,,2],2,quantile,0.995))
mxb = ts(cbind(mxb, rowSums(mxb)), start = tsp(jj)[1], freq=4)
lxb = ts(cbind(lxb, rowSums(lxb)), start = tsp(jj)[1], freq=4)
uxb = ts(cbind(uxb, rowSums(uxb)), start = tsp(jj)[1], freq=4)
names=c('Trend', 'Season', 'Trend + Season')
L = min(lxb[,1])-01; U = max(uxb[,1]) +.01
plot(mxb[,1], ylab=names[1], ylim=c(L,U), type='n')
grid(lty=2); lines(mxb[,1])
xx=c(time(jj), rev(time(jj)))
yy=c(lxb[,1], rev(uxb[,1]))
polygon(xx, yy, border=NA, col=gray(.4, alpha = .2))
L = min(lxb[,3])-01; U = max(uxb[,3]) +.01
plot(mxb[,3], ylab=names[3], ylim=c(L,U), type='n')
grid(lty=2); lines(mxb[,3])
xx=c(time(jj), rev(time(jj)))
yy=c(lxb[,3], rev(uxb[,3]))
polygon(xx, yy, border=NA, col=gray(.4, alpha = .2))

```

Problems

Section 6.1

6.1 Consider a system process given by

$$x_t = -0.9x_{t-2} + w_t \quad t = 1, \dots, n$$

where $x_0 \sim N(0, \sigma_0^2)$, $x_{-1} \sim N(0, \sigma_1^2)$, and w_t is Gaussian white noise with variance σ_w^2 . The system process is observed with noise, say,

$$y_t = x_t + v_t,$$

where v_t is Gaussian white noise with variance σ_v^2 . Further, suppose x_0 , x_{-1} , $\{w_t\}$ and $\{v_t\}$ are independent.

- (a) Write the system and observation equations in the form of a state space model.
- (b) Find the values of σ_0^2 and σ_1^2 that make the observations, y_t , stationary.
- (c) Generate $n = 100$ observations with $\sigma_w = 1$, $\sigma_v = 1$ and using the values of σ_0^2 and σ_1^2 found in (b). Do a time plot of x_t and of y_t and compare the two processes.
Also, compare the sample ACF and PACF of x_t and of y_t .
- (d) Repeat (c), but with $\sigma_v = 10$.

6.2 Consider the state-space model presented in [Example 6.3](#). Let $x_t^{t-1} = E(x_t | y_{t-1}, \dots, y_1)$ and let $P_t^{t-1} = E(x_t - x_t^{t-1})^2$. The innovation sequence or residuals are $\epsilon_t = y_t - y_t^{t-1}$, where $y_t^{t-1} = E(y_t | y_{t-1}, \dots, y_1)$. Find $\text{cov}(\epsilon_s, \epsilon_t)$ in terms of x_t^{t-1} and P_t^{t-1} for (i) $s \neq t$ and (ii) $s = t$.

Section 6.2

6.3 Simulate $n = 100$ observations from the following state-space model:

$$x_t = .8x_{t-1} + w_t \quad \text{and} \quad y_t = x_t + v_t$$

where $x_0 \sim N(0, 2.78)$, $w_t \sim \text{iid } N(0, 1)$, and $v_t \sim \text{iid } N(0, 1)$ are all mutually independent. Compute and plot the data, y_t , the one-step-ahead predictors, y_t^{t-1} along with the root mean square prediction errors, $E^{1/2}(y_t - y_t^{t-1})^2$ using [Example 6.5](#) as a guide.

6.4 Suppose the vector $z = (x', y')'$, where $x (p \times 1)$ and $y (q \times 1)$ are jointly distributed with mean vectors μ_x and μ_y and with covariance matrix

$$\text{cov}(z) = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}.$$

Consider projecting x on $\mathcal{M} = \overline{\text{sp}}\{1, y\}$, say, $\hat{x} = b + By$.

(a) Show the orthogonality conditions can be written as

$$E(x - b - By) = 0,$$

$$E[(x - b - By)y'] = 0,$$

leading to the solutions

$$b = \mu_x - B\mu_y \quad \text{and} \quad B = \Sigma_{xy}\Sigma_{yy}^{-1}.$$

(b) Prove the mean square error matrix is

$$MSE = E[(x - b - By)x'] = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}.$$

(c) How can these results be used to justify the claim that, in the absence of normality, [Property 6.1](#) yields the best linear estimate of the state x_t given the data Y_t , namely, x_t^t , and its corresponding MSE, namely, P_t^t ?

6.5 Projection Theorem Derivation of Property 6.2. Throughout this problem, we use the notation of [Property 6.2](#) and of the Projection Theorem given in [Appendix B](#), where \mathcal{H} is L^2 . If $\mathcal{L}_{k+1} = \overline{\text{sp}}\{y_1, \dots, y_{k+1}\}$, and $\mathcal{V}_{k+1} = \overline{\text{sp}}\{y_{k+1} - y_{k+1}^k\}$, for $k = 0, 1, \dots, n-1$, where y_{k+1}^k is the projection of y_{k+1} on \mathcal{L}_k , then, $\mathcal{L}_{k+1} = \mathcal{L}_k \oplus \mathcal{V}_{k+1}$. We assume $P_0^0 > 0$ and $R > 0$.

(a) Show the projection of x_k on \mathcal{L}_{k+1} , that is, x_k^{k+1} , is given by

$$x_k^{k+1} = x_k^k + H_{k+1}(y_{k+1} - y_{k+1}^k),$$

where H_{k+1} can be determined by the orthogonality property

$$E \left\{ \left(x_k - H_{k+1}(y_{k+1} - y_{k+1}^k) \right) \left(y_{k+1} - y_{k+1}^k \right)' \right\} = 0.$$

Show

$$H_{k+1} = P_k^k \Phi' A'_{k+1} [A_{k+1} P_{k+1}^k A'_{k+1} + R]^{-1}.$$

(b) Define $J_k = P_k^k \Phi' [P_{k+1}^k]^{-1}$, and show

$$x_k^{k+1} = x_k^k + J_k(x_{k+1}^{k+1} - x_{k+1}^k).$$

(c) Repeating the process, show

$$x_k^{k+2} = x_k^k + J_k(x_{k+1}^{k+2} - x_{k+1}^k) + H_{k+2}(y_{k+2} - y_{k+2}^{k+1}),$$

solving for H_{k+2} . Simplify and show

$$x_k^{k+2} = x_k^k + J_k(x_{k+1}^{k+2} - x_{k+1}^k).$$

(d) Using induction, conclude

$$x_k^n = x_k^k + J_k(x_{k+1}^n - x_{k+1}^k),$$

which yields the smoother with $k = t - 1$.

Section 6.3

6.6 Consider the univariate state-space model given by state conditions $x_0 = w_0$, $x_t = x_{t-1} + w_t$ and observations $y_t = x_t + v_t$, $t = 1, 2, \dots$, where w_t and v_t are independent, Gaussian, white noise processes with $\text{var}(w_t) = \sigma_w^2$ and $\text{var}(v_t) = \sigma_v^2$.

- (a) Show that y_t follows an IMA(1,1) model, that is, ∇y_t follows an MA(1) model.
- (b) Fit the model specified in part (a) to the logarithm of the glacial varve series and compare the results to those presented in [Example 3.33](#).

6.7 Consider the model

$$y_t = x_t + v_t,$$

where v_t is Gaussian white noise with variance σ_v^2 , x_t are independent Gaussian random variables with mean zero and $\text{var}(x_t) = r_t \sigma_x^2$ with x_t independent of v_t , and r_1, \dots, r_n are known constants. Show that applying the EM algorithm to the problem of estimating σ_x^2 and σ_v^2 leads to updates (represented by hats)

$$\hat{\sigma}_x^2 = \frac{1}{n} \sum_{t=1}^n \frac{\sigma_t^2 + \mu_t^2}{r_t} \quad \text{and} \quad \hat{\sigma}_v^2 = \frac{1}{n} \sum_{t=1}^n [(y_t - \mu_t)^2 + \sigma_t^2],$$

where, based on the current estimates (represented by tildes),

$$\mu_t = \frac{r_t \tilde{\sigma}_x^2}{r_t \tilde{\sigma}_x^2 + \tilde{\sigma}_v^2} y_t \quad \text{and} \quad \sigma_t^2 = \frac{r_t \tilde{\sigma}_x^2 \tilde{\sigma}_v^2}{r_t \tilde{\sigma}_x^2 + \tilde{\sigma}_v^2}.$$

6.8 To explore the stability of the filter, consider a univariate state-space model. That is, for $t = 1, 2, \dots$, the observations are $y_t = x_t + v_t$ and the state equation is $x_t = \phi x_{t-1} + w_t$, where $\sigma_w = \sigma_v = 1$ and $|\phi| < 1$. The initial state, x_0 , has zero mean and variance one.

- (a) Exhibit the recursion for P_t^{t-1} in [Property 6.1](#) in terms of P_{t-1}^{t-2} .
- (b) Use the result of (a) to verify P_t^{t-1} approaches a limit ($t \rightarrow \infty$) P that is the positive solution of $P^2 - \phi^2 P - 1 = 0$.
- (c) With $K = \lim_{t \rightarrow \infty} K_t$ as given in [Property 6.1](#), show $|1 - K| < 1$.
- (d) Show, in steady-state, the one-step-ahead predictor, $y_{n+1}^n = E(y_{n+1} | y_n, y_{n-1}, \dots)$, of a future observation satisfies

$$y_{n+1}^n = \sum_{j=0}^{\infty} \phi^j K (1 - K)^{j-1} y_{n+1-j}.$$

6.9 In [Section 6.3](#), we discussed that it is possible to obtain a recursion for the gradient vector, $-\partial \ln L_Y(\Theta)/\partial \Theta$. Assume the model is given by [\(6.1\)](#) and [\(6.2\)](#) and A_t is a known design matrix that does not depend on Θ , in which case [Property 6.1](#) applies. For the gradient vector, show

$$\begin{aligned} \partial \ln L_Y(\Theta)/\partial \Theta_i &= \sum_{t=1}^n \left\{ \epsilon_t' \Sigma_t^{-1} \frac{\partial \epsilon_t}{\partial \Theta_i} - \frac{1}{2} \epsilon_t' \Sigma_t^{-1} \frac{\partial \Sigma_t}{\partial \Theta_i} \Sigma_t^{-1} \epsilon_t \right. \\ &\quad \left. + \frac{1}{2} \text{tr} \left(\Sigma_t^{-1} \frac{\partial \Sigma_t}{\partial \Theta_i} \right) \right\}, \end{aligned}$$

where the dependence of the innovation values on Θ is understood. In addition, with the general definition $\partial_i g = \partial g(\Theta)/\partial \Theta_i$, show the following recursions, for $t = 2, \dots, n$ apply:

- (i) $\partial_i \epsilon_t = -A_t \partial_i x_t^{t-1}$,
- (ii) $\partial_i x_t^{t-1} = \partial_i \Phi x_{t-1}^{t-2} + \Phi \partial_i x_{t-1}^{t-2} + \partial_i K_{t-1} \epsilon_{t-1} + K_{t-1} \partial_i \epsilon_{t-1}$,
- (iii) $\partial_i \Sigma_t = A_t \partial_i P_t^{t-1} A_t' + \partial_i R$,
- (iv) $\partial_i K_t = [\partial_i \Phi P_t^{t-1} A_t' + \Phi \partial_i P_t^{t-1} A_t' - K_t \partial_i \Sigma_t] \Sigma_t^{-1}$,
- (v) $\partial_i P_t^{t-1} = \partial_i \Phi P_{t-1}^{t-2} \Phi' + \Phi \partial_i P_{t-1}^{t-2} \Phi' + \Phi P_{t-1}^{t-2} \partial_i \Phi' + \partial_i Q$,
 $- \partial_i K_{t-1} \Sigma_t K_{t-1}' - K_{t-1} \partial_i \Sigma_t K_{t-1}' - K_{t-1} \Sigma_t \partial_i K_{t-1}'$,

using the fact that $P_t^{t-1} = \Phi P_{t-1}^{t-2} \Phi' + Q - K_{t-1} \Sigma_t K_{t-1}'$.

6.10 Continuing with the previous problem, consider the evaluation of the Hessian matrix and the numerical evaluation of the asymptotic variance–covariance matrix of the parameter estimates. The information matrix satisfies

$$E \left\{ -\frac{\partial^2 \ln L_Y(\Theta)}{\partial \Theta \partial \Theta'} \right\} = E \left\{ \left(\frac{\partial \ln L_Y(\Theta)}{\partial \Theta} \right) \left(\frac{\partial \ln L_Y(\Theta)}{\partial \Theta} \right)' \right\};$$

see Anderson (1984, Section 4.4), for example. Show the (i, j) -th element of the information matrix, say, $I_{ij}(\Theta) = E \{ -\partial^2 \ln L_Y(\Theta)/\partial \Theta_i \partial \Theta_j \}$, is

$$\begin{aligned}\mathcal{I}_{ij}(\boldsymbol{\theta}) = \sum_{t=1}^n E\Big\{ & \partial_i \boldsymbol{\epsilon}'_t \boldsymbol{\Sigma}_t^{-1} \partial_j \boldsymbol{\epsilon}_t + \frac{1}{2} \text{tr}(\boldsymbol{\Sigma}_t^{-1} \partial_i \boldsymbol{\Sigma}_t \boldsymbol{\Sigma}_t^{-1} \partial_j \boldsymbol{\Sigma}_t) \\ & + \frac{1}{4} \text{tr}(\boldsymbol{\Sigma}_t^{-1} \partial_i \boldsymbol{\Sigma}_t) \text{tr}(\boldsymbol{\Sigma}_t^{-1} \partial_j \boldsymbol{\Sigma}_t)\Big\}.\end{aligned}$$

Consequently, an approximate Hessian matrix can be obtained from the sample by dropping the expectation, E, in the above result and using only the recursions needed to calculate the gradient vector.

Section 6.4

6.11 As an example of the way the state-space model handles the missing data problem, suppose the first-order autoregressive process

$$x_t = \phi x_{t-1} + w_t$$

has an observation missing at $t = m$, leading to the observations $y_t = A_t x_t$, where $A_t = 1$ for all t , except $t = m$ wherein $A_t = 0$. Assume $x_0 = 0$ with variance $\sigma_w^2/(1 - \phi^2)$, where the variance of w_t is σ_w^2 . Show the Kalman smoother estimators in this case are

$$x_t^n = \begin{cases} \phi y_1 & t = 0, \\ \frac{\phi}{1+\phi^2}(y_{m-1} + y_{m+1}) & t = m, \\ y, & t \neq 0, m, \end{cases}$$

with mean square covariances determined by

$$P_t^n = \begin{cases} \sigma_w^2 & t = 0, \\ \sigma_w^2/(1 + \phi^2) & t = m, \\ 0 & t \neq 0, m. \end{cases}$$

6.12 The data set `ar1miss` is $n = 100$ observations generated from an AR(1) process, $x_t = \phi x_{t-1} + w_t$, with $\phi = .9$ and $\sigma_w = 1$, where 10% of the data have been deleted at random (replaced with `NA`). Use the results of [Problem 6.11](#) to estimate the parameters of the model, ϕ and σ_w , using the EM algorithm, and then estimate the missing values.

Section 6.5

6.13 Redo [Example 6.10](#) on the *logged* Johnson & Johnson quarterly earnings per share.

6.14 Fit a structural model to quarterly unemployment as follows. Use the data in `unemp`, which are monthly. The series can be made quarterly by aggregating and averaging: `y = aggregate(unemp, nfrequency=4, FUN=mean)`, so that `y` is the quarterly average unemployment. Use [Example 6.10](#) as a guide.

Section 6.6

- 6.15** (a) Fit an AR(2) to the recruitment series, R_t in `rec`, and consider a lag-plot of the residuals from the fit versus the SOI series, S_t in `soi`, at various lags, S_{t-h} , for $h = 0, 1, \dots$. Use the lag-plot to argue that S_{t-5} is reasonable to include as an exogenous variable.
 (b) Fit an ARX(2) to R_t using S_{t-5} as an exogenous variable and comment on the results; include an examination of the innovations.

6.16 Use [Property 6.6](#) to complete the following exercises.

- (a) Write a univariate AR(1) model, $y_t = \phi y_{t-1} + v_t$, in state-space form. Verify your answer is indeed an AR(1).
 (b) Repeat (a) for an MA(1) model, $y_t = v_t + \theta v_{t-1}$.
 (c) Write an IMA(1,1) model, $y_t = y_{t-1} + v_t + \theta v_{t-1}$, in state-space form.

6.17 Verify [Property 6.5](#).

6.18 Verify [Property 6.6](#).

Section 6.7

- 6.19** Repeat the bootstrap analysis of [Example 6.13](#) on the entire three-month Treasury bills and rate of inflation data set of 110 observations. Do the conclusions of [Example 6.13](#)—that the dynamics of the data are best described in terms of a fixed, rather than stochastic, regression—still hold?

Section 6.8

6.20 Let y_t represent the global temperature series (`globtemp`) shown in [Figure 1.2](#).

- (a) Fit a smoothing spline using gcv (the default) to y_t and plot the result superimposed on the data. Repeat the fit using `spar=.7`; the gcv method yields `spar=.5` approximately. ([Example 2.14](#) on page 70 may help. Also in R, see the help file `?smooth.spline`.)
 (b) Write the model $y_t = x_t + v_t$ with $\nabla^2 x_t = w_t$, in state-space form. Fit this state-space model to y_t , and exhibit a time plot the estimated smoother, \hat{x}_t^n and the corresponding error limits, $\hat{x}_t^n \pm 2\sqrt{\hat{P}_t^n}$ superimposed on the data.
 (c) Superimpose all the fits from parts (a) and (b) [include the error bounds] on the data and briefly compare and contrast the results.

Section 6.9

6.21 Verify (6.132), (6.133), and (6.134).

6.22 Prove [Property 6.7](#) and verify (6.143).

6.23 Fit a Poisson-HMM to the dataset `polio` from the `gamlss.data` package. The data are reported polio cases in the U.S. for the years 1970 to 1983. To get started, install the package and then type

```
library(gamlss.data)      # load package
plot(polio, type='s')    # view the data
```

6.24 Fit a two-state HMM model to the weekly S&P 500 returns that were analyzed in [Example 6.17](#) and compare the results.

Section 6.10

6.25 Fit the switching model described in [Example 6.20](#) to the growth rate of GNP. The data are in `gnp` and, in the notation of the example, y_t is log-GNP and ∇y_t is the growth rate. Use the code in [Example 6.22](#) as a guide.

6.26 Argue that a switching model is reasonable in explaining the behavior of the number of sunspots (see [Figure 4.22](#)) and then fit a switching model to the sunspot data.

Section 6.11

6.27 Fit a stochastic volatility model to the returns of one (or more) of the four financial time series available in the R datasets package as `EuStockMarkets`.

6.28 Fit a stochastic volatility model to the residuals of the GNP (`gnp`) returns analyzed in [Example 3.39](#).

6.29 We consider the stochastic volatility model (6.197).

(a) Show that for any integer m ,

$$\mathbb{E}[r_t^{2m}] = \beta^{2m} \mathbb{E}[r_t^{2m}] \exp(m^2 \sigma_x^2 / 2),$$

where $\sigma_x^2 = \sigma^2 / (1 - \phi^2)$.

(b) Show (6.198).

(c) Show that for any positive integer h , $\text{var}(X_t + X_{t+h}) = 2\sigma_X^2(1 + \phi^h)$.

(d) Show that

$$\text{cov}(r_t^{2m}, r_{t+h}^{2m}) = \beta^{4m} \left(\mathbb{E}[r_t^{2m}] \right)^2 \left(\exp(m^2 \sigma_x^2 (1 + \phi^h)) - \exp(m^2 \sigma_x^2) \right).$$

(e) Establish (6.199).

Section 6.12

6.30 Verify the distributional statements made in [Example 6.25](#).

6.31 Repeat [Example 6.27](#) on the log of the Johnson & Johnson data.

6.32 Fit an AR(1) to the returns of the US GNP (`gnp`) using a Bayesian approach via MCMC.

Chapter 7

Statistical Methods in the Frequency Domain

In previous chapters, we saw many applied time series problems that involved relating series to each other or to evaluating the effects of treatments or design parameters that arise when time-varying phenomena are subjected to periodic stimuli. In many cases, the nature of the physical or biological phenomena under study are best described by their Fourier components rather than by the difference equations involved in ARIMA or state-space models. The fundamental tools we use in studying periodic phenomena are the discrete Fourier transforms (DFTs) of the processes and their statistical properties. Hence, in [Section 7.2](#), we review the properties of the DFT of a multivariate time series and discuss various approximations to the likelihood function based on the large-sample properties and the properties of the complex multivariate normal distribution. This enables extension of the classical techniques such as ANOVA and principal component analysis to the multivariate time series case, which is the focus of this chapter.

7.1 Introduction

An extremely important class of problems in classical statistics develops when we are interested in relating a collection of input series to some output series. For example, in [Chapter 2](#), we have previously considered relating temperature and various pollutant levels to daily mortality, but have not investigated the frequencies that appear to be driving the relation and have not looked at the possibility of leading or lagging effects. In [Chapter 4](#), we isolated a definite lag structure that could be used to relate sea surface temperature to the number of new recruits. In [Problem 5.10](#), the possible driving processes that could be used to explain inflow to Lake Shasta were hypothesized in terms of the possible inputs precipitation, cloud cover, temperature, and other variables. Identifying the combination of input factors that produce the best prediction for inflow is an example of multiple regression in the frequency domain, with the models treated theoretically by considering the regression, conditional on the random input processes.



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Applied Time Series Analysis

SS 2014

Dr. Marcel Dettling

Institute for Data Analysis and Process Design
Zurich University of Applied Sciences
CH-8401 Winterthur

Table of Contents

<u>1 INTRODUCTION</u>	<u>1</u>
1.1 PURPOSE	1
1.2 EXAMPLES	2
1.3 GOALS IN TIME SERIES ANALYSIS	8
<u>2 MATHEMATICAL CONCEPTS</u>	<u>11</u>
2.1 DEFINITION OF A TIME SERIES	11
2.2 STATIONARITY	11
2.3 TESTING STATIONARITY	13
<u>3 TIME SERIES IN R</u>	<u>15</u>
3.1 TIME SERIES CLASSES	15
3.2 DATES AND TIMES IN R	17
3.3 DATA IMPORT	21
<u>4 DESCRIPTIVE ANALYSIS</u>	<u>23</u>
4.1 VISUALIZATION	23
4.2 TRANSFORMATIONS	26
4.3 DECOMPOSITION	27
4.4 AUTOCORRELATION	46
4.5 PARTIAL AUTOCORRELATION	60
<u>5 STATIONARY TIME SERIES MODELS</u>	<u>63</u>
5.1 WHITE NOISE	63
5.2 ESTIMATING THE CONDITIONAL MEAN	64
5.3 AUTOREGRESSIVE MODELS	65
5.4 MOVING AVERAGE MODELS	79
5.5 ARMA(p,q) MODELS	87
<u>6 SARIMA AND GARCH MODELS</u>	<u>91</u>
6.1 ARIMA MODELS	91
6.2 SARIMA MODELS	94
6.3 ARCH/GARCH MODELS	98
<u>7 TIME SERIES REGRESSION</u>	<u>103</u>
7.1 WHAT IS THE PROBLEM?	103
7.2 FINDING CORRELATED ERRORS	107
7.3 COCHRANE-ORCUTT METHOD	114

7.4 GENERALIZED LEAST SQUARES	115
7.5 MISSING PREDICTOR VARIABLES	121
8 FORECASTING	127
8.1 FORECASTING ARMA	128
8.2 EXPONENTIAL SMOOTHING	134
9 MULTIVARIATE TIME SERIES ANALYSIS	143
9.1 PRACTICAL EXAMPLE	143
9.2 CROSS CORRELATION	147
9.3 PREWHITENING	150
9.4 TRANSFER FUNCTION MODELS	152
10 SPECTRAL ANALYSIS	157
10.1 DECOMPOSING IN THE FREQUENCY DOMAIN	157
11 STATE SPACE MODELS	163
11.1 STATE SPACE FORMULATION	163
11.2 AR PROCESSES WITH MEASUREMENT NOISE	164
11.3 DYNAMIC LINEAR MODELS	167

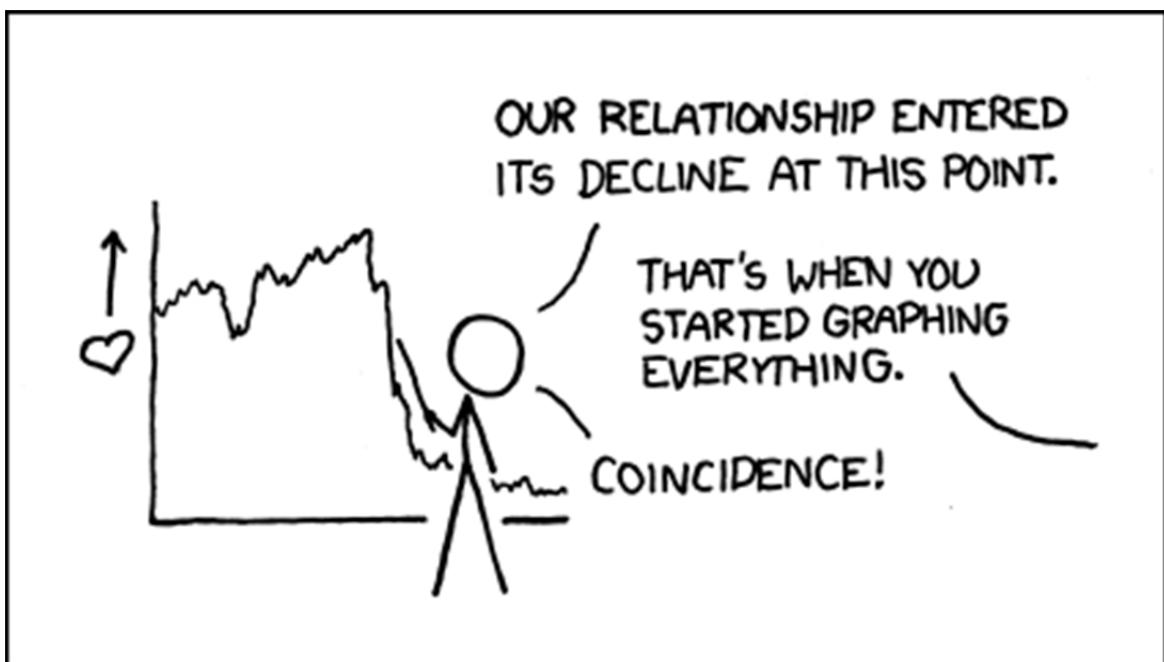
1 Introduction

1.1 Purpose

Time series data, i.e. records which are measured sequentially over time, are extremely common. They arise in virtually every application field, such as e.g.:

- **Business**
Sales figures, production numbers, customer frequencies, ...
- **Economics**
Stock prices, exchange rates, interest rates, ...
- **Official Statistics**
Census data, personal expenditures, road casualties, ...
- **Natural Sciences**
Population sizes, sunspot activity, chemical process data, ...
- **Environmetrics**
Precipitation, temperature or pollution recordings, ...

In contrast to basic data analysis where the assumption of identically and independently distributed data is key, time series are serially correlated. The purpose of time series analysis is to visualize and understand these dependences in past data, and to exploit them for forecasting future values. While some simple descriptive techniques do often considerably enhance the understanding of the data, a full analysis usually involves modeling the stochastic mechanism that is assumed to be the generator of the observed time series.



Once a good model is found and fitted to data, the analyst can use that model to forecast future values and produce prediction intervals, or he can generate simulations, for example to guide planning decisions. Moreover, fitted models are used as a basis for statistical tests: they allow determining whether fluctuations in monthly sales provide evidence of some underlying change, or whether they are still within the range of usual random variation.

The dominant main features of many time series are *trend* and *seasonal variation*. These can either be modeled deterministically by mathematical functions of time, or are estimated using non-parametric smoothing approaches. Yet another key feature of most time series is that adjacent observations tend to be correlated, i.e. serially dependent. Much of the methodology in time series analysis is aimed at explaining this correlation using appropriate statistical models.

While the theory on mathematically oriented time series analysis is vast and may be studied without necessarily fitting any models to data, the focus of our course will be applied and directed towards data analysis. We study some basic properties of time series processes and models, but mostly focus on how to visualize and describe time series data, on how to fit models to data correctly, on how to generate forecasts, and on how to adequately draw conclusions from the output that was produced.

1.2 Examples

1.2.1 Air Passenger Bookings

The numbers of international passenger bookings (in thousands) per month on an airline (*PanAm*) in the United States were obtained from the Federal Aviation Administration for the period 1949-1960. The company used the data to predict future demand before ordering new aircraft and training aircrew. The data are available as a time series in **R**. Here, we here show how to access them, and how to first gain an impression.

```
> data(AirPassengers)
> AirPassengers
   Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

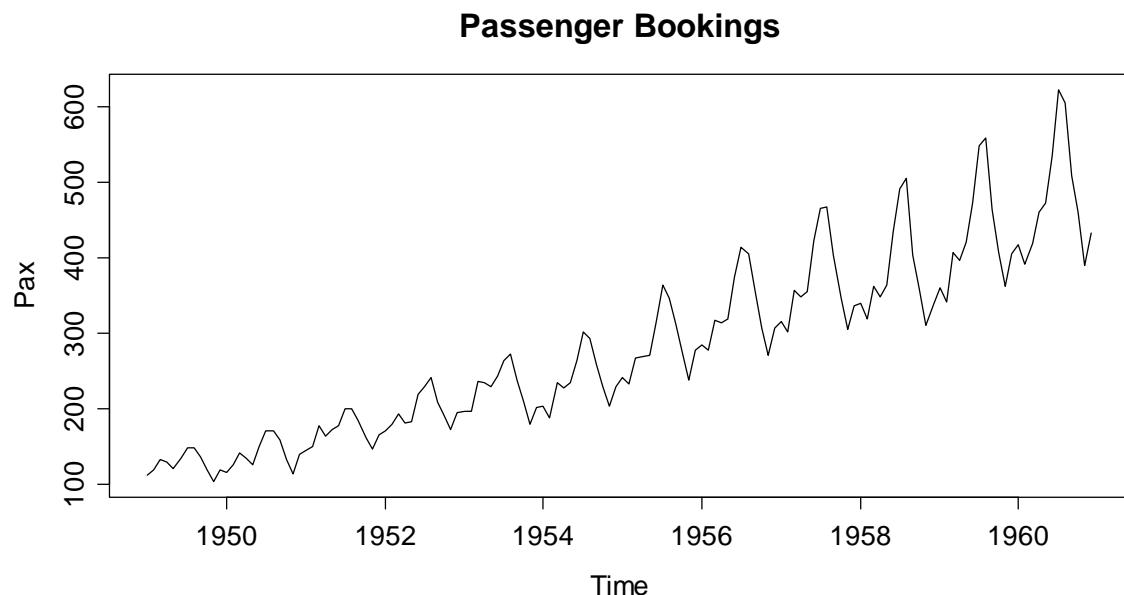
Some further information about this dataset can be obtained by typing `?AirPassengers` in R. The data are stored in an R-object of class `ts`, which is the specific class for time series data. However, for further details on how time series are handled in R, we refer to section 3.

One of the most important steps in time series analysis is to visualize the data, i.e. create a time series plot, where the air passenger bookings are plotted versus the time of booking. For a time series object, this can be done very simply in R, using the generic plot function:

```
> plot(AirPassengers, ylab="Pax", main="Passenger Bookings")
```

The result is displayed on the next page. There are a number of features in the plot which are common to many time series. For example, it is apparent that the number of passengers travelling on the airline is increasing with time. In general, a systematic change in the mean level of a time series that does not appear to be periodic is known as a *trend*. The simplest model for a trend is a linear increase or decrease, an often adequate approximation. We will discuss how to estimate trends, and how to decompose time series into trend and other components in section 4.3.

The data also show a repeating pattern within each year, i.e. in summer, there are always more passengers than in winter. This is known as a *seasonal effect*, or *seasonality*. Please note that this term is applied more generally to any repeating pattern over a fixed period, such as for example restaurant bookings on different days of week.



We can naturally attribute the increasing trend of the series to causes such as rising prosperity, greater availability of aircraft, cheaper flights and increasing population. The seasonal variation coincides strongly with vacation periods. For this reason, we here consider both trend and seasonal variation as deterministic

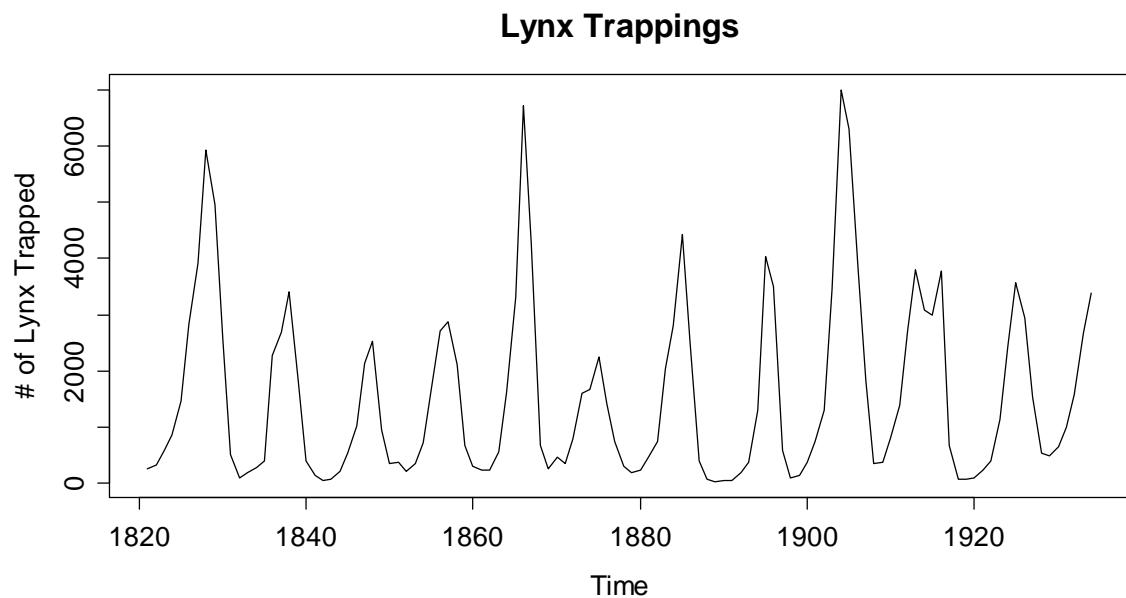
components. As mentioned before, section 4.3 discusses visualization and estimation of these components, while in section 7, time series regression models will be specified to allow for underlying causes like these, and finally section 8 discusses exploiting these for predictive purposes.

1.2.2 Lynx Trappings

The next series which we consider here is the annual number of lynx trappings for the years 1821-1934 in Canada. We again load the data and visualize them using a time series plot:

```
> data(lynx)
> plot(lynx, ylab="# of Lynx Trapped", main="Lynx Trappings")
```

The plot on the next page shows that the number of trapped lynx reaches high and low values every about 10 years, and some even larger figure every about 40 years. To our knowledge, there is no fixed natural period which suggests these results. Thus, we will attribute this behavior not to a deterministic periodicity, but to a random, stochastic one.



This leads us to the heart of time series analysis: while understanding and modeling trend and seasonal variation is a very important aspect, much of the time series methodology is aimed at *stationary series*, i.e. data which do not show deterministic, but only random (cyclic) variation.



1.2.3 Luteinizing Hormone Measurements

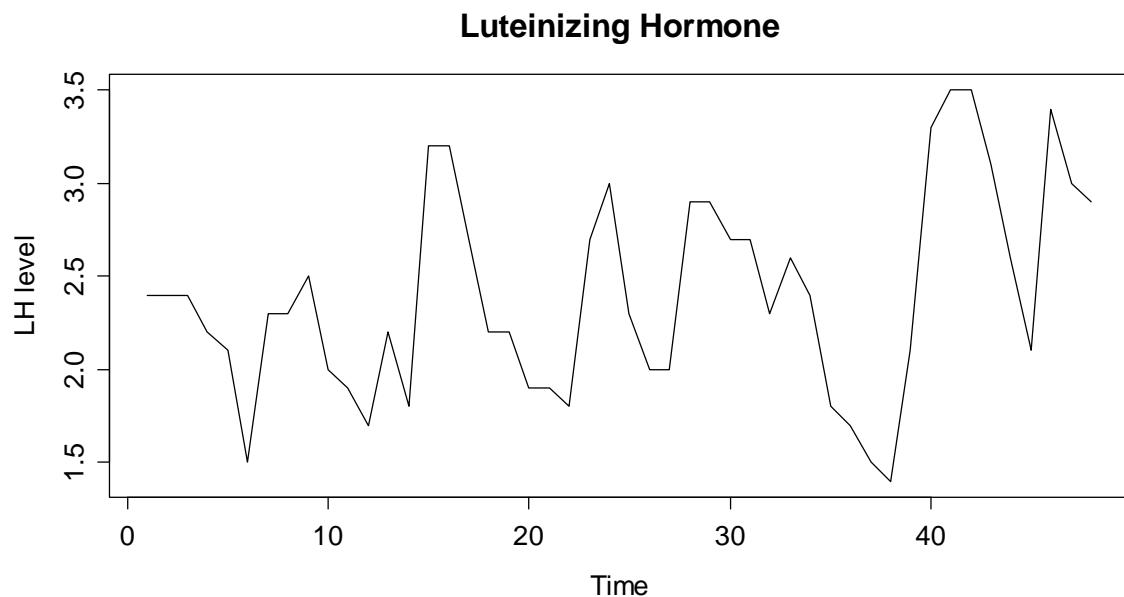
One of the key features of the above lynx trappings series is that the observations apparently do not stem from independent random variables, but there is some serial correlation. If the previous value was high (or low, respectively), the next one is likely to be similar to the previous one. To explore, model and exploit such dependence lies at the root of time series analysis.

We here show another series, where 48 luteinizing hormone levels were recorded from blood samples that were taken at 10 minute intervals from a human female. This hormone, also called *lutropin*, triggers ovulation.

```
> data(lh)
> lh
Time Series:
Start = 1; End = 48; Frequency = 1
[1] 2.4 2.4 2.4 2.2 2.1 1.5 2.3 2.3 2.5 2.0 1.9 1.7 2.2 1.8
[15] 3.2 3.2 2.7 2.2 2.2 1.9 1.9 1.8 2.7 3.0 2.3 2.0 2.0 2.9
[29] 2.9 2.7 2.7 2.3 2.6 2.4 1.8 1.7 1.5 1.4 2.1 3.3 3.5 3.5
[43] 3.1 2.6 2.1 3.4 3.0 2.9
```

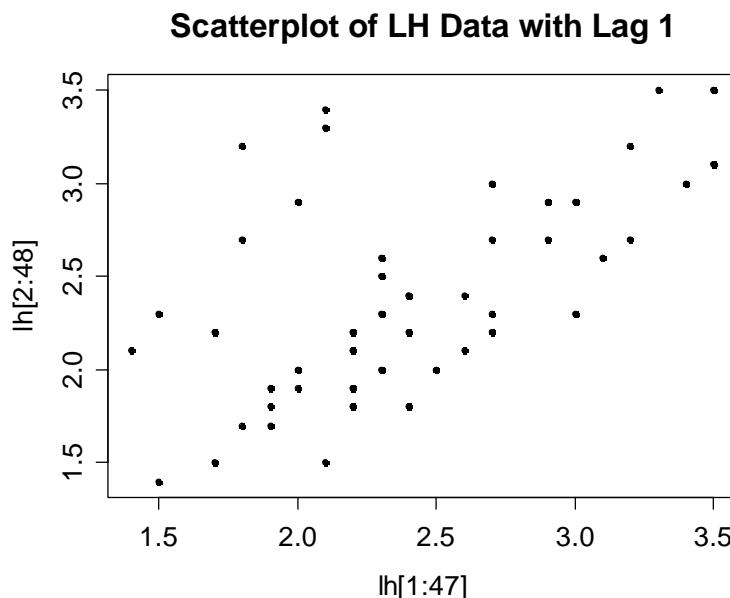
Again, the data themselves are of course needed to perform analyses, but provide little overview. We can improve this by generating a time series plot:

```
> plot(lh, ylab="LH level", main="Luteinizing Hormone")
```



For this series, given the way the measurements were made (i.e. 10 minute intervals), we can almost certainly exclude any deterministic seasonal variation. But is there any stochastic cyclic behavior? This question is more difficult to answer. Normally, one resorts to the simpler question of analyzing the correlation of subsequent records, called *autocorrelations*. The autocorrelation for lag 1 can be visualized by producing a scatterplot of adjacent observations:

```
> plot(lh[1:47], lh[2:48], pch=20)
> title("Scatterplot of LH Data with Lag 1")
```



Besides the (non-standard) observation that there seems to be an inhomogeneity, i.e. two distinct groups of data points, it is apparent that there is a positive correlation between successive measurements. This manifests itself with the clearly visible fact that if the previous observation was above or below the mean, the next one is more likely to be on the same side. We can even compute the value of the Pearson correlation coefficient:

```
> cor(lh[1:47], lh[2:48])
[1] 0.5807322
```

This figure is an estimate for the so-called *autocorrelation coefficient at lag 1*. As we will see in section 4.4, the idea of considering lagged scatterplots and computing Pearson correlation coefficients serves as a good proxy for a mathematically more sound method. We also note that despite the positive correlation of +0.58, the series seems to always have the possibility of “reverting to the other side of the mean”, a property which is common to *stationary series* – an issue that will be discussed in section 2.2.

1.2.4 Swiss Market Index

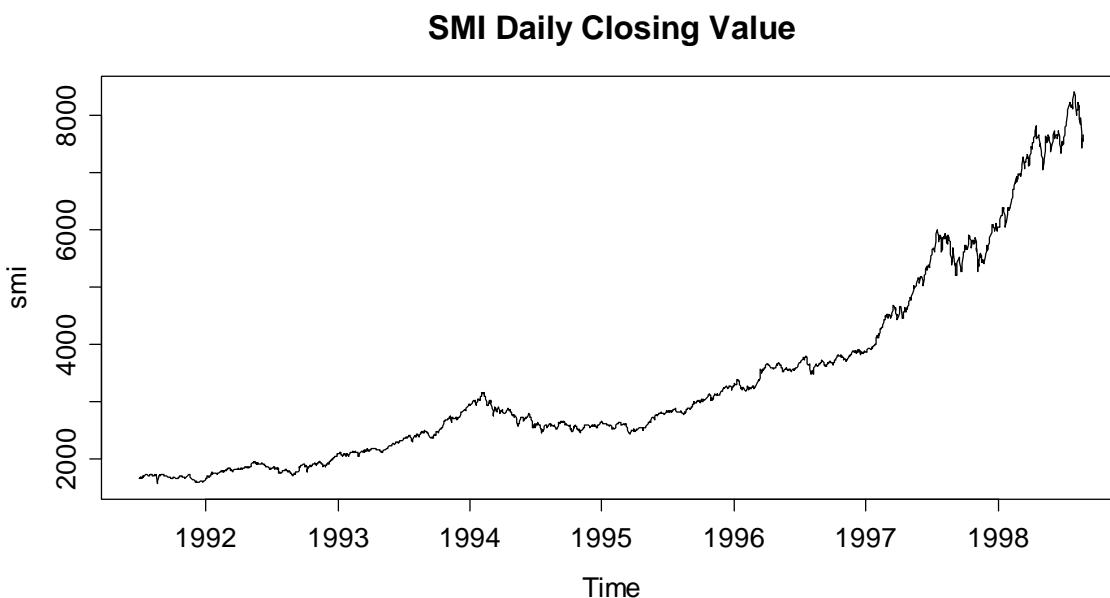
The *SMI* is the blue chip index of the Swiss stock market. It summarizes the value of the shares of the 20 most important companies, and currently contains nearly 90% of the total market capitalization. It was introduced on July 1, 1988 at a basis level of 1500.00 points. Daily closing data for 1860 consecutive trading days from 1991-1998 are available in *R*. We observe a more than 4-fold increase during that period. As a side note, the value in the second half of 2013 is around 8000 points, indicating a sideways movement over the latest 15 years.

```
> data(EuStockMarkets)
> EuStockMarkets
Time Series:
Start = c(1991, 130)
End = c(1998, 169)
Frequency = 260
      DAX     SMI     CAC     FTSE
1991.496 1628.75 1678.1 1772.8 2443.6
1991.500 1613.63 1688.5 1750.5 2460.2
1991.504 1606.51 1678.6 1718.0 2448.2
1991.508 1621.04 1684.1 1708.1 2470.4
1991.512 1618.16 1686.6 1723.1 2484.7
1991.515 1610.61 1671.6 1714.3 2466.8
```

As we can see, `EuStockMarkets` is a multiple time series object, which also contains data from the German DAX, the French CAC and UK's FTSE. We will focus on the SMI and thus extract and plot the series:

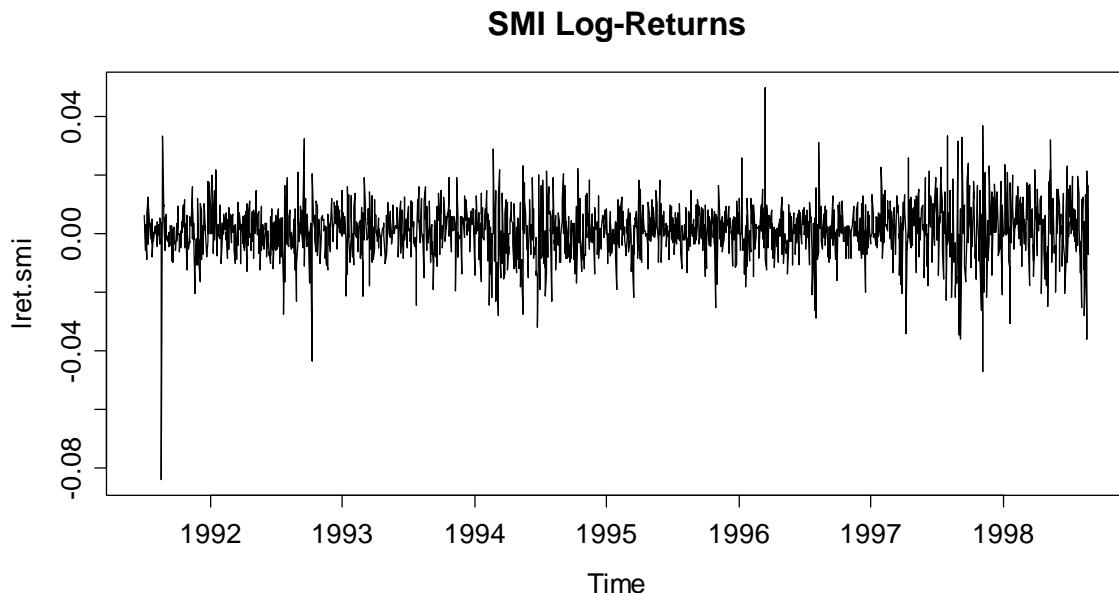
```
esm <- EuStockMarkets
tmp <- EuStockMarkets[, 2]
smi <- ts(tmp, start=start(esm), freq=frequency(esm))
plot(smi, main="SMI Daily Closing Value")
```

Because subsetting from a multiple time series object results in a *vector*, but not a *time series object*, we need to regenerate a latter one, sharing the arguments of the original. In the plot we clearly observe that the series has a trend, i.e. the mean is obviously non-constant over time. This is typical for all financial time series.



Such trends in financial time series are nearly impossible to predict, and difficult to characterize mathematically. We will not embark in this, but analyze the so-called log-returns, i.e. the logged-value of today's value divided by the one of yesterday:

```
> lret.smi <- diff(log(smi))
> plot(lret.smi, main="SMI Log-Returns")
```



The SMI log-returns are a close approximation to the relative change (percent values) with respect to the previous day. As can be seen above, they do not exhibit a trend anymore, but show some of the stylized facts that most log-returns of financial time series share. Using *lagged scatterplots* or the *correlogram* (to be discussed later in section 4.4), you can convince yourself that there is no serial correlation. Thus, there is no direct dependency which could be exploited to predict tomorrow's return based on the one of today and/or previous days.

However, it is visible that large changes, i.e. log-returns with high absolute values, imply that future log-returns tend to be larger than normal, too. This feature is also known as *volatility clustering*, and financial service providers are trying their best to exploit this property to make profit. Again, you can convince yourself of the volatility clustering effect by taking the squared log-returns and analyzing their serial correlation, which is different from zero.

1.3 Goals in Time Series Analysis

A first impression of the purpose and goals in time series analysis could be gained from the previous examples. We conclude this introductory section by explicitly summarizing the most important goals.

1.3.1 Exploratory Analysis

Exploratory analysis for time series mainly involves *visualization* with time series plots, *decomposition* of the series into deterministic and stochastic parts, and studying the *dependency structure* in the data.

1.3.2 Modeling

The formulation of a stochastic model, as it is for example also done in regression, can and does often lead to a deeper understanding of the series. The formulation of a suitable model usually arises from a mixture between background knowledge in the applied field, and insight from exploratory analysis. Once a suitable model is found, a central issue remains, i.e. the *estimation* of the parameters, and subsequent *model diagnostics* and *evaluation*.

1.3.3 Forecasting

An often-heard motivation for time series analysis is the prediction of future observations in the series. This is an ambitious goal, because time series forecasting relies on *extrapolation*, and is generally based on the assumption that past and present characteristics of the series continue. It seems obvious that good forecasting results require a very good comprehension of a series' properties, be it in a more descriptive sense, or in the sense of a fitted model.

1.3.4 Time Series Regression

Rather than just forecasting by extrapolation, we can try to understand the relation between a so-identified *response time series*, and one or more *explanatory series*. If all of these are observed at the same time, we can in principle employ the *ordinary least squares* (OLS) regression framework. However, the all-to-common assumption of (serially) uncorrelated errors in OLS is usually violated in a time series setup. We will illustrate how to properly deal with this situation, in order to generate correct confidence and prediction intervals.

1.3.5 Process Control

Many production or other processes are measured quantitatively for the purpose of *optimal management* and *quality control*. This usually results in time series data, to which a stochastic model is fit. This allows understanding the signal in the data, but also the noise: it becomes feasible to monitor which fluctuations in the production are normal, and which ones require intervention.

2 Mathematical Concepts

For performing anything else than very basic exploratory time series analysis, even from a much applied perspective, it is necessary to introduce the mathematical notion of what a time series is, and to study some basic probabilistic properties, namely the *moments* and the *concept of stationarity*.

2.1 Definition of a Time Series

As we have explained in section 1.2, observations that have been collected over fixed sampling intervals form a time series. Following a statistical approach, we consider such series as realizations of random variables. A sequence of random variables, defined at such fixed sampling intervals, is sometimes referred to as a *discrete-time stochastic process*, though the shorter names *time series model* or *time series process* are more popular and will mostly be used in this scriptum. It is very important to make the distinction between a time series, i.e. observed values, and a process, i.e. a probabilistic construct.

Definition: A *time series process* is a set of random variables $\{X_t, t \in T\}$, where T is the set of times at which the process was, will or can be observed. We assume that each random variable X_t is distributed according some univariate distribution function F_t . Please note that for our entire course and hence scriptum, we exclusively consider time series processes with equidistant time intervals, as well as real-valued random variables X_t . This allows us to enumerate the set of times, so that we can write $T = \{1, 2, 3, \dots\}$.

An observed time series, on the other hand, is seen as a realization of the random vector $X = (X_1, X_2, \dots, X_n)$, and is denoted with small letters $x = (x_1, x_2, \dots, x_n)$. It is important to note that in a multivariate sense, a time series is only *one single* realization of the n -dimensional random variable X , with its multivariate, n -dimensional distribution function F . As we all know, we cannot do statistics with just a single observation. As a way out of this situation, we need to impose some conditions on the joint distribution function F .

2.2 Stationarity

The aforementioned condition on the joint distribution F will be formulated as the concept of *stationarity*. In colloquial language, stationarity means that the probabilistic character of the series must not change over time, i.e. that any section of the time series is “typical” for every other section with the same length. More mathematically, we require that for any indices s, t and k , the observations x_t, \dots, x_{t+k} could have just as easily occurred at times $s, \dots, s+k$. If that is not the case practically, then the series is hardly stationary.

Imposing even more mathematical rigor, we introduce the concept of *strict stationarity*. A time series is said to be *strictly stationary* if and only if the $(k+1)$ -dimensional joint distribution of X_1, \dots, X_{t+k} coincides with the joint distribution of X_s, \dots, X_{s+k} for any combination of indices t, s and k . For the special case of $k=0$ and $t=s$, this means that the univariate distributions F_t of all X_t are equal. For strictly stationary time series, we can thus leave off the index t on the distribution. As the next step, we will define the unconditional moments:

$$\begin{aligned}\text{Expectation } \mu &= E[X_t], \\ \text{Variance } \sigma^2 &= \text{Var}(X_t), \\ \text{Covariance } \gamma(h) &= \text{Cov}(X_t, X_{t+h}).\end{aligned}$$

In other words, strictly stationary series have *constant (unconditional) expectation*, *constant (unconditional) variance*, and the covariance, i.e. the *dependency structure*, depends only on the lag h , which is the time difference between the two observations. However, the covariance terms are generally different from 0, and thus, the X_t are usually dependent. Moreover, the conditional expectation given the past of the series, $E[X_t | X_{t-1}, X_{t-2}, \dots]$ is typically non-constant, denoted as μ_t . In some (rarer, e.g. for financial time series) cases, even the conditional variance $\text{Var}(X_t | X_{t-1}, X_{t-2}, \dots)$ can be non-constant.

In practice however, except for simulation studies, we usually have no explicit knowledge of the latent time series process. Since strict stationarity is defined as a property of the process' joint distributions (all of them), it is impossible to verify from an observed time series, i.e. a single data realization. We can, however, try to verify whether a time series process shows *constant unconditional mean and variance*, and whether the *dependency only depends on the lag h*. This much less rigorous property is known as *weak stationarity*.

In order to do well-founded statistical analyses with time series, weak stationarity is a necessary condition. It is obvious that if a series' observations do not have common properties such as constant mean/variance and a stable dependency structure, it will be impossible to statistically learn from it. On the other hand, it can be shown that weak stationarity, along with the additional property of ergodicity (i.e. the mean of a time series realization converges to the expected value, independent of the starting point), is sufficient for most practical purposes such as model fitting, forecasting, etc.. We will, however, not further embark in this subject.

Remarks:

- From now on, when we speak of *stationarity*, we strictly mean weak stationarity. The motivation is that weak stationarity is sufficient for applied time series analysis, and strict stationarity is a practically useless concept.
- When we analyze time series data, we need to verify whether it might have arisen from a stationary process or not. Be careful with the wording: stationarity is always a property of the process, and never of the data.

- Moreover, bear in mind that stationarity is a hypothesis, which needs to be evaluated for every series. We may be able to reject this hypothesis with quite some certainty if the data strongly speak against it. However, we can never prove stationarity with data. At best, it is plausible that a series originated from a stationary process.
- Some obvious violations of stationarity are trends, non-constant variance, deterministic seasonal variation, as well as apparent breaks in the data, which are indicators for changing dependency structure.

2.3 Testing Stationarity

If, as explained above, stationarity is a hypothesis which is tested on data, students and users keep asking if there are any formal tests. The answer to this question is yes, and there are even quite a number of tests. This includes the *Augmented Dickey-Fuller Test*, the *Phillips-Perron Test*, the *KPSS Test*, which are all available in R's `tseries` package. The `urca` package includes further tests such as the *Elliott-Rothenberg-Stock*, *Schmidt-Phillips* und *Zivot-Andrews*.

However, we will not discuss any of these tests here for a variety of reasons. First and foremost, they all focus on some very specific non-stationarity aspects, but do not test stationarity in a broad sense. While they may reasonably do their job in the narrow field they are aimed for, they have low power to detect general non-stationarity and in practice often fail to do so. Additionally, theory and formalism of these tests is quite complex, and thus beyond the scope of this course. In summary, these tests are to be seen as more of a pastime for the mathematically interested, rather than a useful tool for the practitioner.

Thus, we here recommend assessing stationarity by visual inspection. The primary tool for this is the time series plot, but also the correlogram (see section 4.4) can be helpful as a second check. For long time series, it can also be useful to split up the series into several parts for checking whether mean, variance and dependency are similar over the blocks.

3 Time Series in R

3.1 Time Series Classes

In **R**, there are *objects*, which are organized in a large number of *classes*. These classes e.g. include *vectors*, *data frames*, *model output*, *functions*, and many more. Not surprisingly, there are also several classes for time series. We start by presenting `ts`, the basic class for regularly spaced time series. This class is comparably simple, as it can only represent time series with fixed interval records, and only uses numeric time stamps, i.e. (sophistically) enumerates the index set. However, it will be sufficient for most, if not all, of what we do in this course. Then, we also provide an outlook to more complicated concepts.

3.1.1 The `ts` Class

For defining a time series of class `ts`, we of course need to provide the *data*, but also the *starting time* as argument `start`, and the *frequency* of measurements as argument `frequency`. If no starting time is supplied, **R** uses its default value of 1, i.e. enumerates the times by the index set $1, \dots, n$, where n is the length of the series. The frequency is the number of observations per unit of time, e.g. 1 for yearly, 4 for quarterly, or 12 for monthly recordings. Instead of the start, we could also provide the end of the series, and instead of the frequency, we could supply argument `deltat`, the fraction of the sampling period between successive observations. The following example will illustrate the concept.

Example: We here consider a simple and short series that holds the number of days per year with traffic holdups in front of the Gotthard road tunnel north entrance in Switzerland. The data are available from the Federal Roads Office.

2004	2005	2006	2007	2008	2009	2010
88	76	112	109	91	98	139

The start of this series is in 2004. The time unit is years, and since we have just one record per year, the frequency of this series is 1. This tells us that while there may be a trend, there cannot be a seasonal effect, as the latter can only be present in periodic series, i.e. series with frequency > 1 . We now define a `ts` object in **R**.

```
> rawdat <- c(88, 76, 112, 109, 91, 98, 139)
> ts.dat <- ts(rawdat, start=2004, freq=1)
> ts.dat
Time Series: Start = 2004, End = 2010
Frequency = 1
[1] 88 76 112 109 91 98 139
```

There are a number of simple but useful functions that extract basic information from objects of class `ts`, see the following examples:

```
> start(ts.dat)
[1] 2004    1

> end(ts.dat)
[1] 2010    1

> frequency(ts.dat)
[1] 1

> deltat(ts.dat)
[1] 1
```

Another possibility is to obtain the measurement times from a time series object. As class `ts` only enumerates the times, they are given as fractions. This can still be very useful for specialized plots, etc.

```
> time(ts.dat)
Time Series:
Start = 2004
End = 2010
Frequency = 1
[1] 2004 2005 2006 2007 2008 2009 2010
```

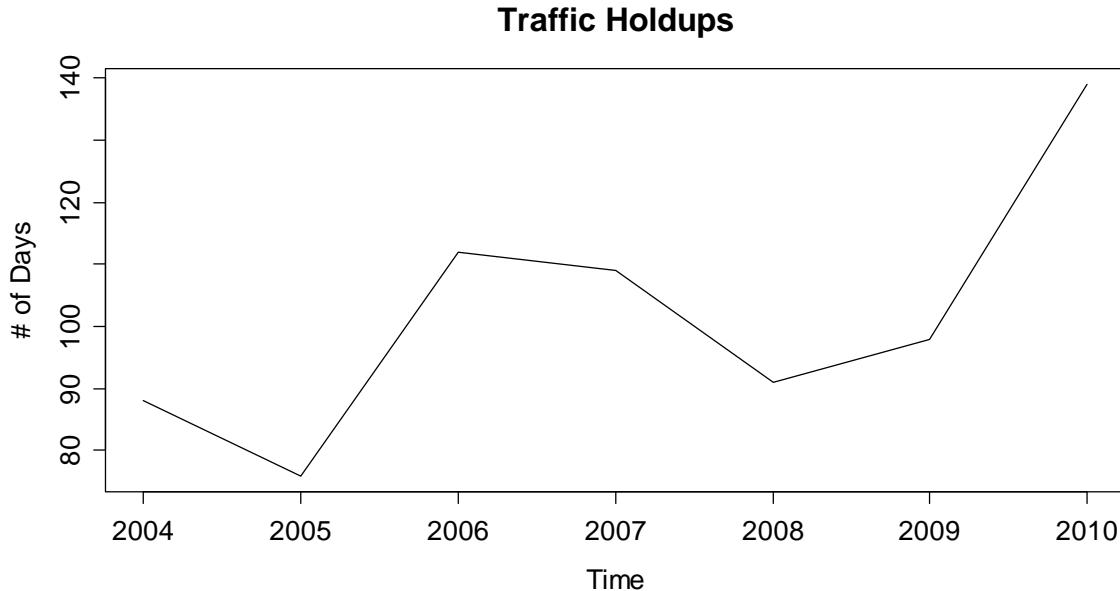
The next basic, but for practical purposes very useful function is `window()`. It is aimed at selecting a subset from a time series. Of course, also regular R-subsetting such as `ts.dat[2:5]` does work with the time series class. However, this results in a vector rather than a time series object, and is thus mostly of less use than the `window()` command.

```
> window(ts.dat, start=2006, end=2008)
Time Series:
Start = 2006
End = 2008
Frequency = 1
[1] 112 109 91
```

While we here presented the most important basic methods/functions for class `ts`, there is a wealth of further ones. This includes the `plot()` function, and many more, e.g. for estimating trends, seasonal effects and dependency structure, for fitting time series models and generating forecasts. We will present them in the forthcoming chapters of this scriptum.

To conclude the previous example, we will not do without showing the time series plot of the Gotthard road tunnel traffic holdup days, see next page. Because there are a limited number of observations, it is difficult to give statements regarding a possible trend and/or stochastic dependency.

```
> plot(ts.dat, ylab="# of Days", main="Traffic Holdups")
```



3.1.2 Other Classes

Besides the basic `ts` class, there are several more which offer a variety of additional options, but will rarely to never be required during our course. Most prominently, this includes the `zoo` package, which provides infrastructure for both regularly and irregularly spaced time series using arbitrary classes for the time stamps. It is designed to be as consistent as possible with the `ts` class. Coercion from and to `zoo` is also readily available.

Some further packages which contain classes and methods for time series include `xts`, `its`, `tseries`, `fts`, `timeSeries` and `tis`. Additional information on their content and philosophy can be found on CRAN.

3.2 Dates and Times in R

While for the `ts` class, the handling of times has been solved very simply and easily by enumerating, doing time series analysis in R may sometimes also require to explicitly working with date and time. There are several options for dealing with date and date/time data. The built-in `as.Date()` function handles dates that come without times. The contributed package `chron` handles dates and times, but does not control for different time zones, whereas the sophisticated but complex `POSIXct` and `POSIXlt` classes allow for dates and times with time zone control.

As a general rule for date/time data in R, we suggest to use the simplest technique possible. Thus, for date only data, `as.Date()` will mostly be the optimal choice. If handling dates and times, but without time-zone information, is required, the `chron` package is the choice. The `POSIX` classes are especially useful in the relatively rare cases when time-zone manipulation is important.

Apart from the `POSIXlt` class, dates/times are internally stored as the number of days or seconds from some reference date. These dates/times thus generally have a numeric mode. The `POSIXlt` class, on the other hand, stores date/time values as a list of components (hour, min, sec, mon, etc.), making it easy to extract these parts. Also the current date is accessible by typing `Sys.Date()` in the console, and returns an object of class `Date`.

3.2.1 The Date Class

As mentioned above, the easiest solution for specifying days in R is with the `as.Date()` function. Using the `format` argument, arbitrary date formats can be read. The default, however, is four-digit year, followed by month and then day, separated by dashes or slashes:

```
> as.Date("2012-02-14")
[1] "2012-02-14"
> as.Date("2012/02/07")
[1] "2012-02-07"
```

If the dates come in non-standard appearance, we require defining their format using some codes. While the most important ones are shown below, we refer to the R help file of function `strptime()` for the full list.

Code	Value
%d	Day of the month (decimal number)
%m	Month (decimal number)
%b	Month (character, abbreviated)
%B	Month (character, full name)
%y	Year (decimal, two digit)
%Y	Year (decimal, four digit)

The following examples illustrate the use of the `format` argument:

```
> as.Date("27.01.12", format="%d.%m.%y")
[1] "2012-01-27"
> as.Date("14. Februar, 2012", format="%d. %B, %Y")
[1] "2012-02-14"
```

Internally, `Date` objects are stored as the number of days passed since the 1st of January in 1970. Earlier dates receive negative numbers. By using the `as.numeric()` function, we can easily find out how many days are past since the reference date. Also back-conversion from a number of past days to a date is straightforward:

```
> mydat <- as.Date("2012-02-14")
> ndays <- as.numeric(mydat)
> ndays
[1] 15384
```

```
> tdays <- 10000
> class(tdays) <- "Date"
> tdays
[1] "1997-05-19"
```

A very useful feature is the possibility of extracting weekdays, months and quarters from Date objects, see the examples below. This information can be converted to factors. In this form, they serve for purposes such as visualization, decomposition, or time series regression.

```
> weekdays(mydat)
[1] "Dienstag"
> months(mydat)
[1] "Februar"
> quarters(mydat)
[1] "Q1"
```

Furthermore, some very useful summary statistics can be generated from Date objects: median, mean, min, max, range, ... are all available. We can even subtract two dates, which results in a difftime object, i.e. the time difference in days.

```
> dat <- as.Date(c("2000-01-01", "2004-04-04", "2007-08-09"))
> dat
[1] "2000-01-01" "2004-04-04" "2007-08-09"

> min(dat)
[1] "2000-01-01"
> max(dat)
[1] "2007-08-09"
> mean(dat)
[1] "2003-12-15"
> median(dat)
[1] "2004-04-04"

> dat[3]-dat[1]
Time difference of 2777 days
```

Another option is generating time sequences. For example, to generate a vector of 12 dates, starting on August 3, 1985, with an interval of one single day between them, we simply type:

```
> seq(as.Date("1985-08-03"), by="days", length=12)
[1] "1985-08-03" "1985-08-04" "1985-08-05" "1985-08-06"
[5] "1985-08-07" "1985-08-08" "1985-08-09" "1985-08-10"
[9] "1985-08-11" "1985-08-12" "1985-08-13" "1985-08-14"
```

The by argument proves to be very useful. We can supply various units of time, and even place an integer in front of it. This allows creating a sequence of dates separated by two weeks:

```
> seq(as.Date("1992-04-17"), by="2 weeks", length=12)
[1] "1992-04-17" "1992-05-01" "1992-05-15" "1992-05-29"
[5] "1992-06-12" "1992-06-26" "1992-07-10" "1992-07-24"
[9] "1992-08-07" "1992-08-21" "1992-09-04" "1992-09-18"
```

3.2.2 The chron Package

The `chron()` function converts dates and times to `chron` objects. The dates and times are provided separately to the `chron()` function, which may well require some initial pre-processing. For such parsing, R-functions such as `substr()` and `strsplit()` can be of great use. In the `chron` package, there is no support for time zones and daylight savings time, and `chron` objects are internally stored as fractional days since the reference date of January 1st, 1970. By using the function `as.numeric()`, these internal values can be accessed. The following example illustrates the use of `chron`:

```
> library(chron)
> dat <- c("2007-06-09 16:43:20", "2007-08-29 07:22:40",
   "2007-10-21 16:48:40", "2007-12-17 11:18:50")
> dts <- substr(dat, 1, 10)
> tme <- substr(dat, 12, 19)
> fmt <- c("y-m-d", "h:m:s")
> cdt <- chron(dates=dts, time=tme, format=fmt)
> cdt
[1] (07-06-09 16:43:20) (07-08-29 07:22:40)
[3] (07-10-21 16:48:40) (07-12-17 11:18:50)
```

As before, we can again use the entire palette of summary statistic functions. Of some special interest are time differences, which can now be obtained as either fraction of days, or in weeks, hours, minutes, seconds, etc.:

```
> cdt[2]-cdt[1]
Time in days:
[1] 80.61065
> difftime(cdt[2], cdt[1], units="secs")
Time difference of 6964760 secs
```

3.2.3 POSIX Classes

The two classes `POSIXct` and `POSIXlt` implement date/time information, and in contrast to the `chron` package, also support time zones and daylight savings time. We recommend utilizing this functionality only when urgently needed, because the handling requires quite some care, and may on top of that be system dependent. Further details on the use of the `POSIX` classes can be found on CRAN.

As explained above, the `POSIXct` class also stores dates/times with respect to the internal reference, whereas the `POSIXlt` class stores them as a list of components (hour, min, sec, mon, etc.), making it easy to extract these parts.

3.3 Data Import

We can safely assume that most time series data are already present in electronic form; however, not necessarily in `R`. Thus, some knowledge on how to import data into `R` is required. It is beyond the scope of this scriptum to present the uncounted options which exist for this task. Hence, we will restrict ourselves to providing a short overview and some useful hints.

The most common form for sharing time series data are certainly spreadsheets, or in particular, Microsoft Excel files. While `library(ROBDC)` offers functionality to directly import data from Excel files, we discourage its use. First of all, this only works on Windows systems. More importantly, it is usually simpler, quicker and more flexible to export comma- or tab-separated text files from Excel, and import them via the ubiquitous `read.table()` function, respectively the tailored version `read.csv()` (for comma separation) and `read.delim()` (for tab separation).

With packages `RODBC` and `RMySQL`, `R` can also communicate with SQL databases, which is the method of choice for large scale problems. Furthermore, after loading `library(foreign)`, it is also possible to read files from Stata, SPSS, Octave and SAS.

4 Descriptive Analysis

As always when working with “a pile of numbers”, also known as *data*, it is important to first gain an overview. In the field of time series analysis, this encompasses several aspects:

- understanding the context of the problem and the data source
- making suitable plots, looking for general structure and outliers
- thinking about data transformations, e.g. to reduce skewness
- judging stationarity and potentially achieve it by decomposition

We start by discussing time series plots, then discuss transformations, focus on the decomposition of time series into trend, seasonal effect and stationary random part and conclude by discussing methods for visualizing the dependency structure.

4.1 Visualization

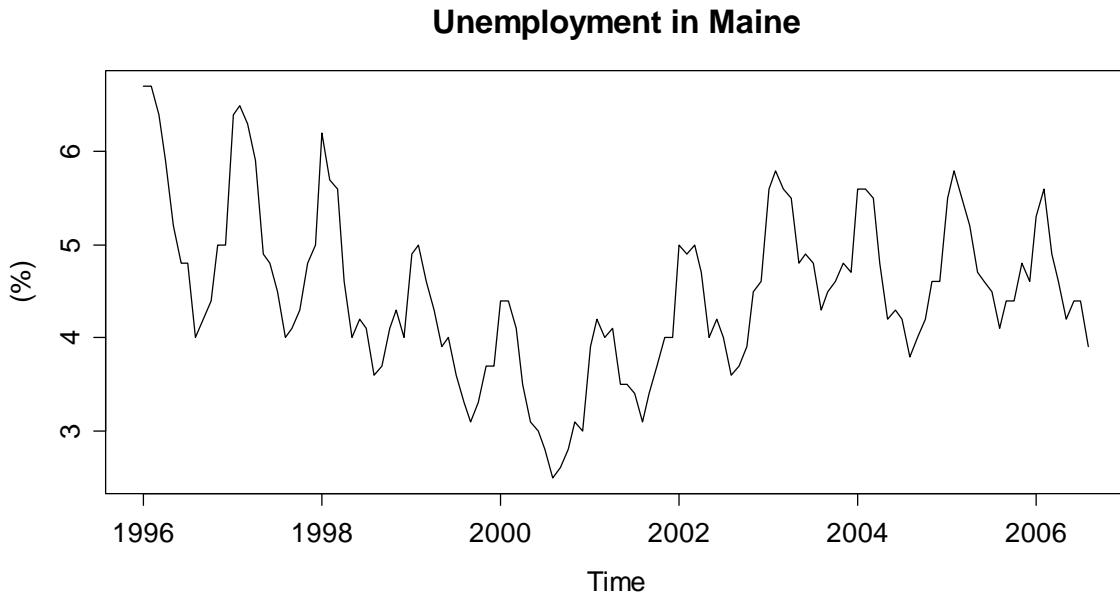
4.1.1 Time Series Plot

The most important means of visualization is the time series plot, where the data are plotted versus time/index. There are several examples in section 1.2, where we also got acquainted with R’s generic `plot()` function. As a general rule, the data points are joined by lines in time series plots. An exception is when there are missing values. Moreover, the reader expects that the axes are well-chosen, labeled and the measurement units are given.

Another issue is the correct aspect ratio for time series plots: if the time axis gets too much compressed, it can become difficult to recognize the behavior of a series. Thus, we recommend choosing the aspect ratio appropriately. However, there are no hard and simple rules on how to do this. As a rule of the thumb, use the “banking to 45 degrees” paradigm: increase and decrease in periodic series should not be displayed at angles much higher or lower than 45 degrees. For very long series, this can become difficult on either A4 paper or a computer screen. In this case, we recommend splitting up the series and display it in different frames.

For illustration, we here show an example, the monthly unemployment rate in the US state of Maine, from January 1996 until August 2006. The data are available from a text file on the web. We can read it directly into R, define the data as an object of class `ts` and then do the time series plot:

```
> www <- "http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/"
> dat <- read.table(paste(www,"Maine.dat",sep=" ", header=T)
> tsd <- ts(dat, start=c(1996,1), freq=12)
> plot(tsd, ylab="(%)", main="Unemployment in Maine")
```



Not surprisingly for monthly economic data, the series shows both seasonal variation and a non-linear trend. Since unemployment rates are one of the main economic indicators used by politicians/decision makers, this series poses a worthwhile forecasting problem.

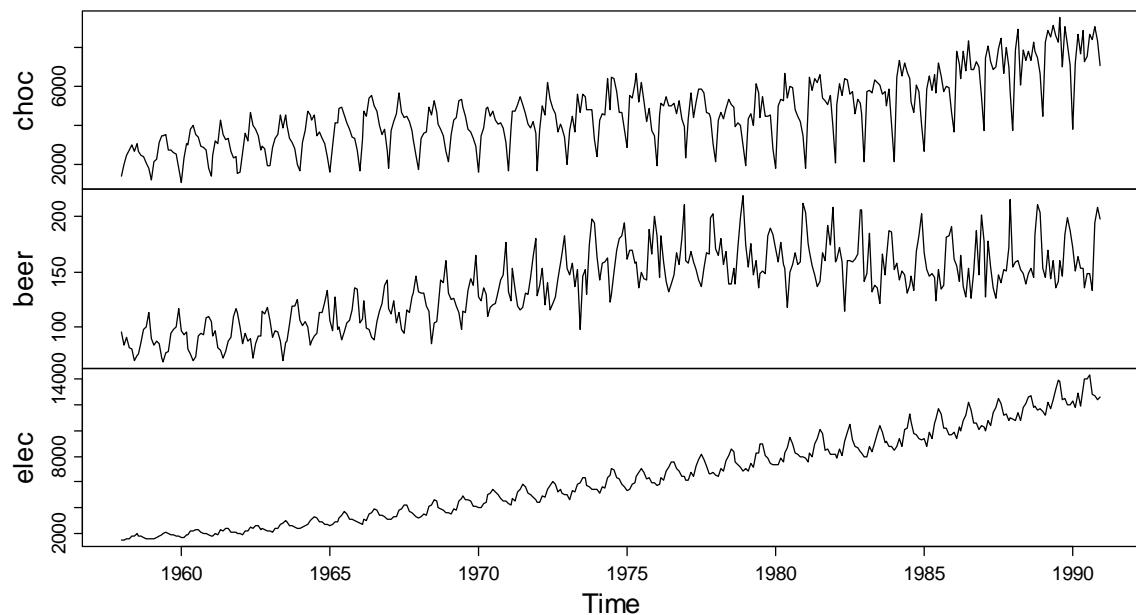
4.1.2 Multiple Time Series Plots

In applied problems, one is often provided with multiple time series. Here, we illustrate some basics on import, definition and plotting. Our example exhibits the monthly supply of electricity (millions of kWh), beer (millions of liters) and chocolate-based production (tonnes) in Australia over the period from January 1958 to December 1990. These data are available from the Bureau of Australian Statistics and are, in pre-processed form, accessible as a text-file online.

```
www <- "http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/"
dat <- read.table(paste(www, "cbe.dat", sep = " ", header=T))
tsd <- ts(dat, start=1958, freq=12)
plot(tsd, main="Chocolate, Beer & Electricity")
```

All three series show a distinct seasonal pattern, along with a trend. It is also instructive to know that the Australian population increased by a factor of 1.8 during the period where these three series were observed. As visible in the bit of code above, plotting multiple series into different panels is straightforward. As a general rule, using different frames for multiple series is the most recommended means of visualization. However, sometimes it can be more instructive to have them in the same frame. Of course, this requires that the series are either on the same scale, or have been indexed, resp. standardized to be so. While R offers function `ts.plot()` to include multiple series in the same frame, that function does not allow color coding. For this reason, we prefer doing some manual work.

Chocolate, Beer & Electricity



```

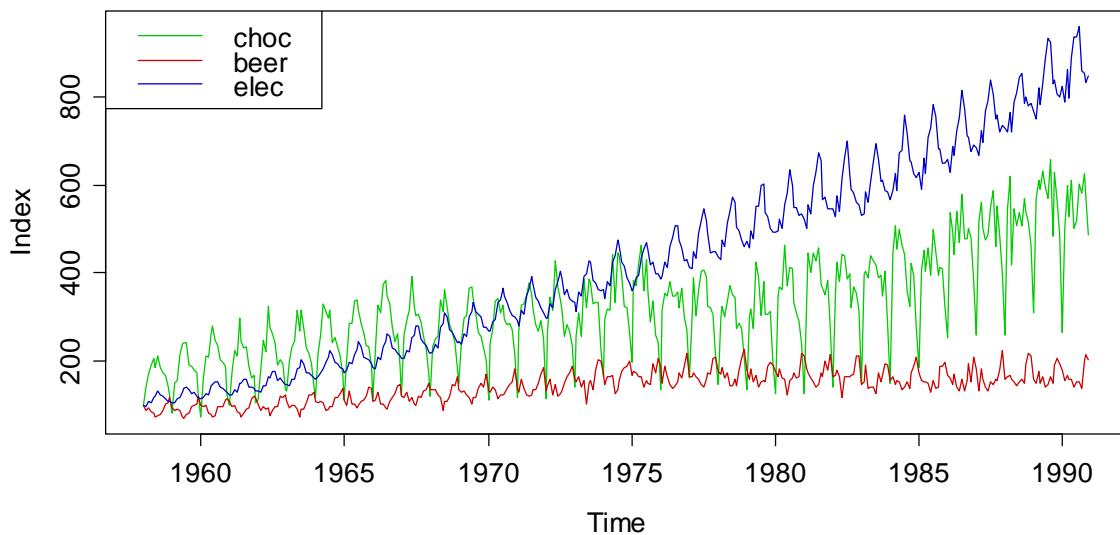
## Indexing the series
tsd[,1] <- tsd[,1]/tsd[1,1]*100
tsd[,2] <- tsd[,2]/tsd[1,2]*100
tsd[,3] <- tsd[,3]/tsd[1,3]*100

## Plotting in one single frame
clr <- c("green3", "red3", "blue3")
plot.ts(tsd[,1], ylim=range(tsd), ylab="Index", col=clr[1])
title("Indexed Chocolate, Beer & Electricity")
lines(tsd[,2], col=clr[2]); lines(tsd[,3], col=clr[3])

## Legend
ltxt <- names(dat)
legend("topleft", lty=1, col=clr, legend=ltxt)

```

Indexed Chocolate, Beer & Electricity

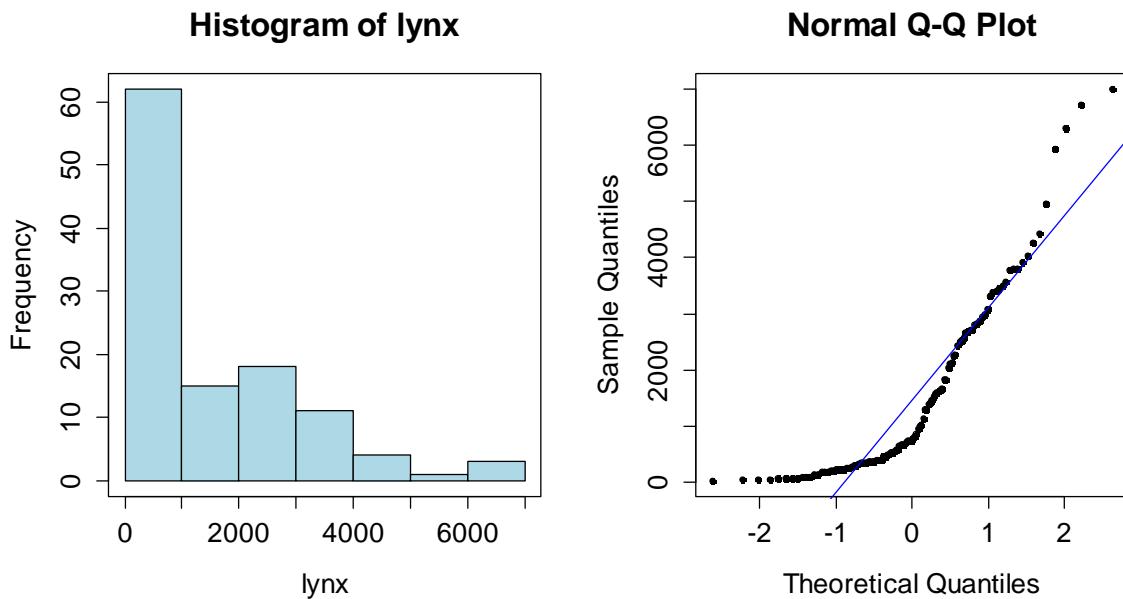


In the indexed single frame plot above, we can very well judge the relative development of the series over time. Due to different scaling, this was nearly impossible with the multiple frames on the previous page. We observe that electricity production increased around 8x during 1958 and 1990, whereas for chocolate the multiplier is around 4x, and for beer less than 2x. Also, the seasonal variation is most pronounced for chocolate, followed by electricity and then beer.

4.2 Transformations

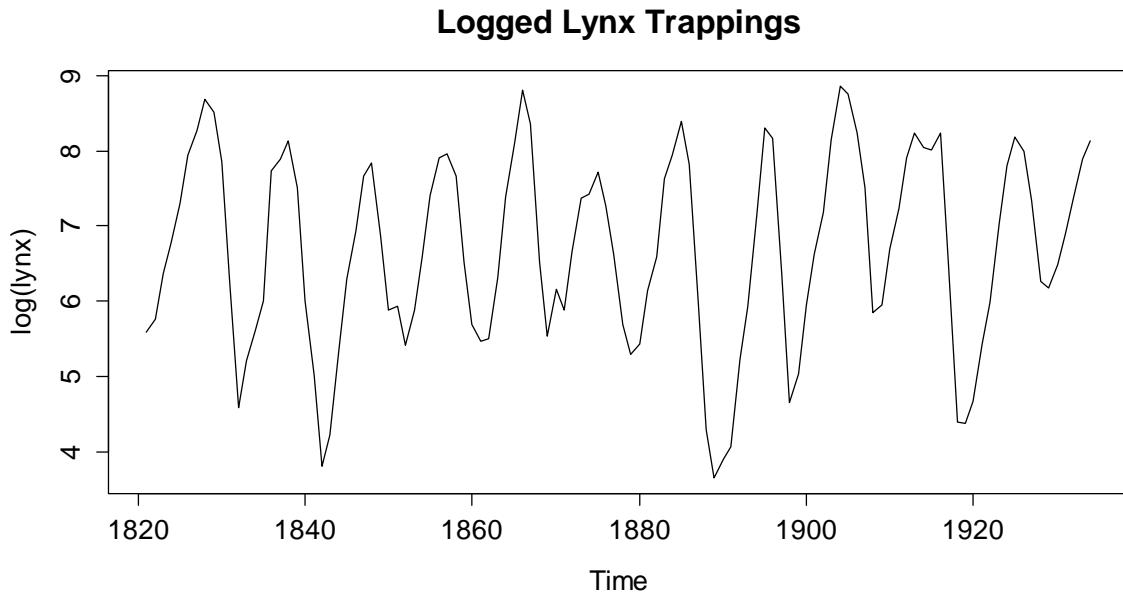
Many popular time series models are based on the Gaussian distribution and linear relations between the variables. However, data may exhibit different behavior. In such cases, we can often improve the fit by not using the original data x_1, \dots, x_n , but a transformed version $g(x_1), \dots, g(x_n)$. The most popular and practically relevant transformation is $g(\cdot) = \log(\cdot)$. It is indicated if either the variation in the series grows with increasing mean, or if the marginal distribution appears to be right-skewed. Both properties often appear in data that can take positive values only, such as the lynx trappings from section 1.2.2. It is easy to spot right-skewness by histograms and QQ-plots:

```
> hist(lynx, col="lightblue")
> qqnorm(lynx, pch=20); qqline(lynx, col="blue")
```



The lynx data show some very strong right-skewness and hence, a log-transformation is indicated. Of course, it was not wrong to use the original scale for a time series plot, but when it comes to estimating autocorrelations or estimating time series models, it is clearly better to log-transform the data first. This is very easy in R:

```
> plot(log(lynx))
> title("Logged Lynx Trappings")
```



The data now follow a more symmetrical pattern; the extreme upward spikes are all gone. We will use these transformed data to determine the autocorrelation and to generate forecasts. However, please be aware of the fact that back-transforming fitted or predicted (model) values to the original scale by just taking $\exp(\cdot)$ usually leads to biased results, unless a correction factor is used. An in-depth discussion of that issue is contained in chapter 8.

4.3 Decomposition

4.3.1 The Basics

We have learned in section 2.2 that stationarity is an important prerequisite for being able to statistically learn from time series data. However, many of the example series exhibit either trend and/or seasonal effect, and thus are non-stationary. In this section, we will learn how to deal with that. It is achieved by using *decomposition models*, the easiest of which is the *simple additive* one:

$$X_t = m_t + s_t + R_t,$$

where X_t is the time series process at time t , m_t is the trend, s_t is the seasonal effect, and R_t is the remainder, i.e. a sequence of usually correlated random variables with mean zero. The goal is to find a decomposition such that R_t is a stationary time series process. Such a model might be suitable for all the monthly-data series we got acquainted with so far: air passenger bookings, unemployment in Maine and Australian production. However, closer inspection of all these series exhibits that the seasonal effect and the random variation increase as the trend increases. In such cases, a multiplicative decomposition model is better:

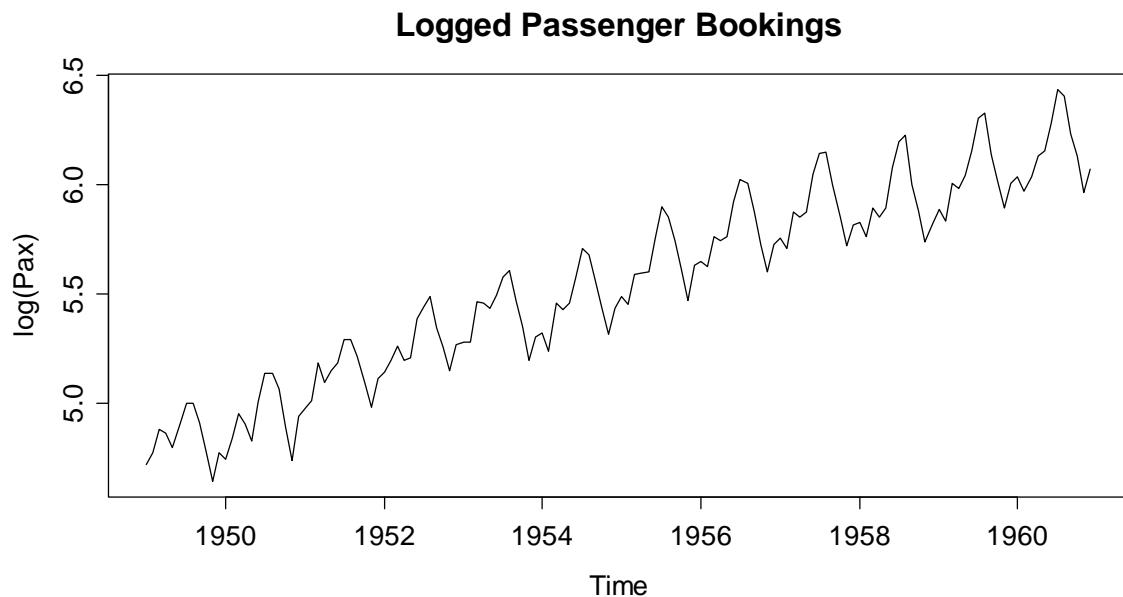
$$X_t = m_t \cdot s_t \cdot R_t$$

Empirical experience says that taking logarithms is beneficial for such data. Also, some basic math shows that this brings us back to the additive case:

$$\log(X_t) = \log(m_t \cdot s_t \cdot R_t) = \log(m_t) + \log(s_t) + \log(R_t) = m'_t + s'_t + R'_t$$

For illustration, we carry out the log-transformation on the air passenger bookings:

```
> plot(log(AirPassengers), ylab="log(Pax)", main=...)
```



Indeed, seasonal effect and random variation now seem to be independent of the level of the series. Thus, the multiplicative model is much more appropriate than the additive one. However, a further snag is that the seasonal effect seems to alter over time rather than being constant. That issue will be addressed later.

4.3.2 Differencing

A simple approach for removing deterministic trends and/or seasonal effects from a time series is by taking differences. A practical interpretation of taking differences is that by doing so, the changes in the data will be monitored, but no longer the series itself. While this is conceptually simple and quick to implement, the main disadvantage is that it does not result in explicit estimates of the trend component m_t and the seasonal component s_t .

We will first turn our attention to series with an additive trend, but without seasonal variation. By taking first-order differences with lag 1, and assuming a trend with little short-term changes, i.e. $m_t \approx m_{t-1}$, we have:

$$\begin{aligned} X_t &= m_t + R_t \\ Y_t &= X_t - X_{t-1} \approx R_t - R_{t-1} \end{aligned}$$

In practice, this kind of differencing approach “mostly works”, i.e. manages to reduce presence of a trend in the series in a satisfactory manner. However, the trend is only fully removed if it is exactly linear, i.e. $m_t = \alpha + \beta t$. Then, we obtain:

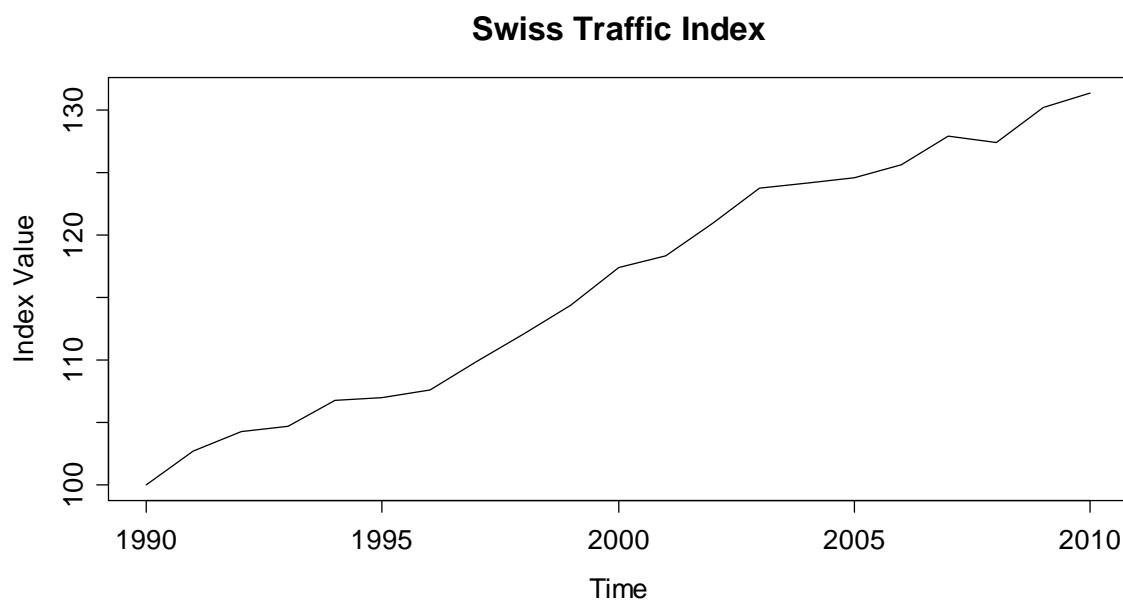
$$Y_t = X_t - X_{t-1} = \beta + R_t - R_{t-1}$$

Another somewhat disturbing property of the differencing approach is that strong, artificial new dependencies are created, meaning that the autocorrelation in Y_t is different from the one in R_t . For illustration, consider a stochastically independent remainder R_t : the differenced process Y_t has autocorrelation!

$$\begin{aligned} \text{Cov}(Y_t, Y_{t-1}) &\approx \text{Cov}(R_t - R_{t-1}, R_{t-1} - R_{t-2}) \\ &= -\text{Cov}(R_{t-1}, R_{t-1}) \\ &\neq 0 \end{aligned}$$

We illustrate how differencing works by using a dataset that shows the traffic development on Swiss roads. The data are available from the federal road office (ASTRA) and show the indexed traffic amount from 1990-2010. We type in the values and plot the original series:

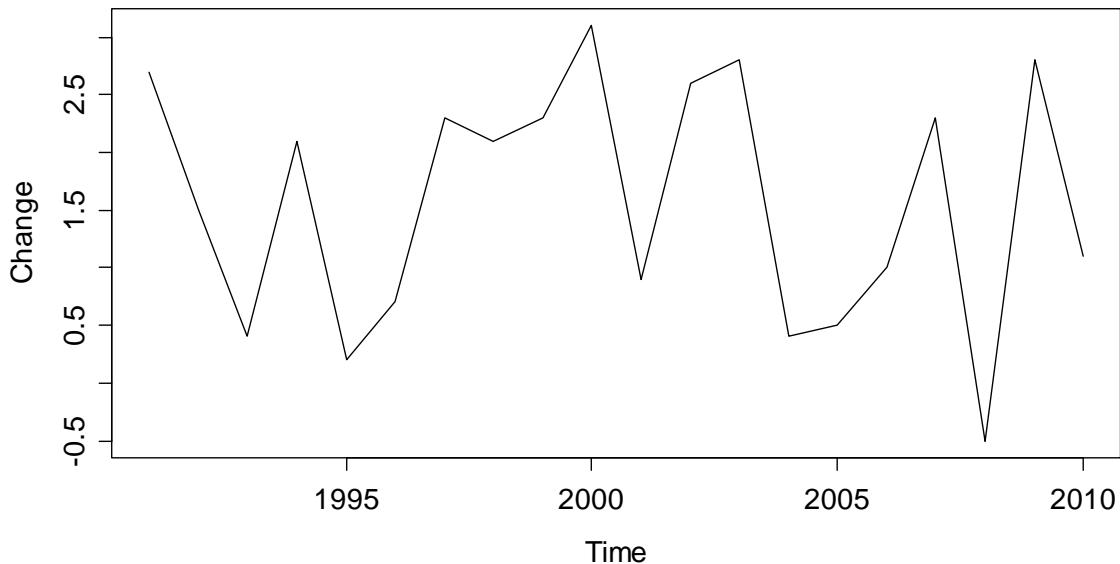
```
> SwissTraffic <- ts(c(100.0, 102.7, 104.2, 104.6, 106.7,
  106.9, 107.6, 109.9, 112.0, 114.3,
  117.4, 118.3, 120.9, 123.7, 124.1,
  124.6, 125.6, 127.9, 127.4, 130.2,
  131.3), start=1990, freq=1)
> plot(SwissTraffic)
```



There is a clear trend, which is close to linear, thus the simple approach should work well here. Taking first-order differences with lag 1 shows the yearly changes in the Swiss Traffic Index, which must now be a stationary series. In R, the job is done with function `diff()`.

```
> diff(SwissTraffic)
Time Series:
Start = 1991
End = 2010
Frequency = 1
[1]  2.7  1.5  0.4  2.1  0.2  0.7  2.3  2.1  2.3  3.1
[11] 0.9  2.6  2.8  0.4  0.5  1.0  2.3 -0.5  2.8  1.1
```

Differenced Swiss Traffic Index



Please note that the time series of differences is now 1 instance shorter than the original series. The reason is that for the first year, 1990, there is no difference to the previous year available. The differenced series now clearly has a constant mean, i.e. the trend was successfully removed.

Log-Transformation and Differencing

On a sidenote, we consider a series that was log-transformed first, before first-order differences with lag 1 were taken. An example is the SMI data that were shown in section 1.2.4. The result is the so-called *log return*, which is an approximation to the relative change, i.e. the percent in- or decrease with respect to the previous instance. In particular:

$$Y_t = \log(X_t) - \log(X_{t-1}) = \log\left(\frac{X_t}{X_{t-1}}\right) = \log\left(\frac{X_t - X_{t-1} + 1}{X_{t-1}} + 1\right) \approx \frac{X_t - X_{t-1}}{X_{t-1}}$$

The approximation of the log return to the relative change is very good for small changes, and becomes a little less precise with larger values. For example, if we have a 0.00% relative change, then $Y_t = 0.00\%$, for 1.00% relative change we obtain $Y_t = 0.995\%$ and for 5.00%, $Y_t = 4.88\%$. We conclude with summarizing that for any non-stationary series which is also due to a log-transformation, the transformation is always carried out first, and then followed by the differencing!

The Backshift Operator

We here introduce the backshift operator B because it allows for convenient notation. When the operator B is applied to X_t , it returns the instance at lag 1, i.e.

$$B(X_t) = X_{t-1}.$$

Less mathematically, we can also say that applying B means “go back one step”, or “increment the time series index t by -1”. The operation of taking first-order differences at lag 1 as above can be written using the backshift operator:

$$Y_t = (1 - B)X_t = X_t - X_{t-1}$$

However, the main aim of the backshift operator is to deal with more complicated forms of differencing, as will be explained below.

Higher-Order Differencing

We have seen that taking first-order differences is able to remove linear trends from time series. What has differencing to offer for polynomial trends, i.e. quadratic or cubic ones? We here demonstrate that it is possible to take higher order differences to remove also these, for example, in the case of a quadratic trend.

$$\begin{aligned} X_t &= \alpha + \beta_1 t + \beta_2 t^2 + R_t, \quad R_t \text{ stationary} \\ Y_t &= (1 - B)^2 X_t \\ &= (X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) \\ &= R_t - 2R_{t-1} + R_{t-2} + 2\beta_2 \end{aligned}$$

We see that the operator $(1 - B)^2$ means that after taking “normal” differences, the resulting series is again differenced “normally”. This is a discretized variant of taking the second derivative, and thus it is not surprising that it manages to remove a quadratic trend from the data. As we can see, Y_t is an additive combination of the stationary R_t ’s terms, and thus itself stationary. Again, if R_t was an independent process, that would clearly not hold for Y_t , thus taking higher-order differences (strongly!) alters the dependency structure.

Moreover, the extension to cubic trends and even higher orders d is straightforward. We just use the $(1 - B)^d$ operator applied to series X_t . In R, we can employ function `diff()`, but have to provide argument `differences=d` for indicating the order of the difference d .

Removing Seasonal Effects by Differencing

For time series with monthly measurements, seasonal effects are very common. Using an appropriate form of differencing, it is possible to remove these, as well as potential trends. We take first-order differences with lag p :

$$Y_t = (1 - B^p)X_t = X_t - X_{t-p},$$

Here, p is the period of the seasonal effect, or in other words, the frequency of series, which is the number of measurements per time unit. The series Y_t then is made up of the changes compared to the previous period's value, e.g. the previous year's value. Also, from the definition, with the same argument as above, it is evident that not only the seasonal variation, but also a strictly linear trend will be removed.

Usually, trends are not exactly linear. We have seen that taking differences at lag 1 removes slowly evolving (non-linear) trends well due to $m_t \approx m_{t-1}$. However, here the relevant quantities are m_t and m_{t-p} , and especially if the period p is long, some trend will usually be remaining in the data. Then, further action is required.

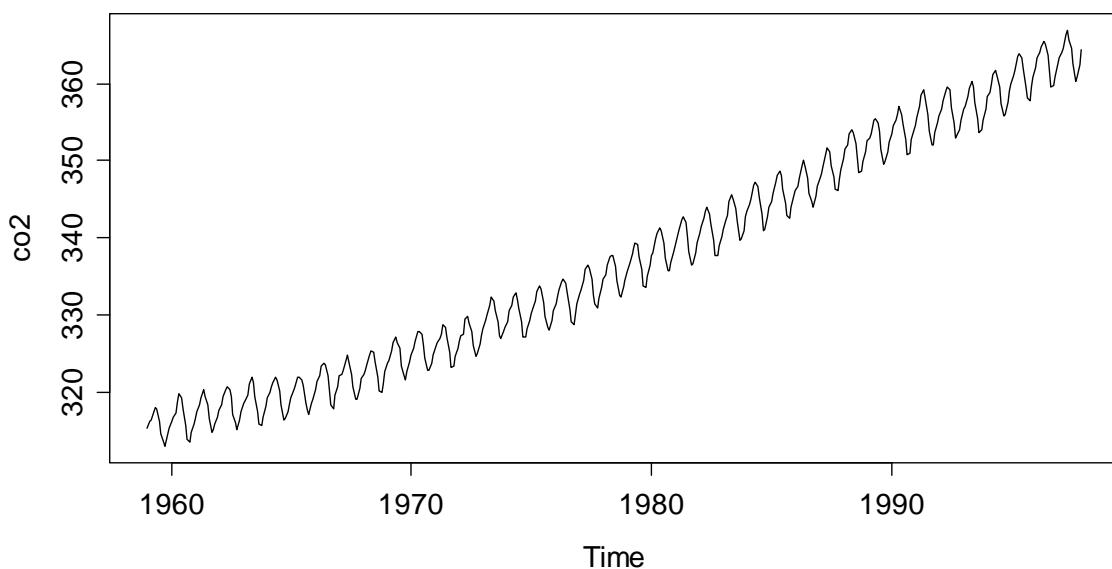
Example

We are illustrating seasonal differencing using the Mauna Loa atmospheric CO_2 concentration data. This is a time series with monthly records from January 1959 to December 1997. It exhibits both a trend and a distinct seasonal pattern. We first load the data and do a time series plot:

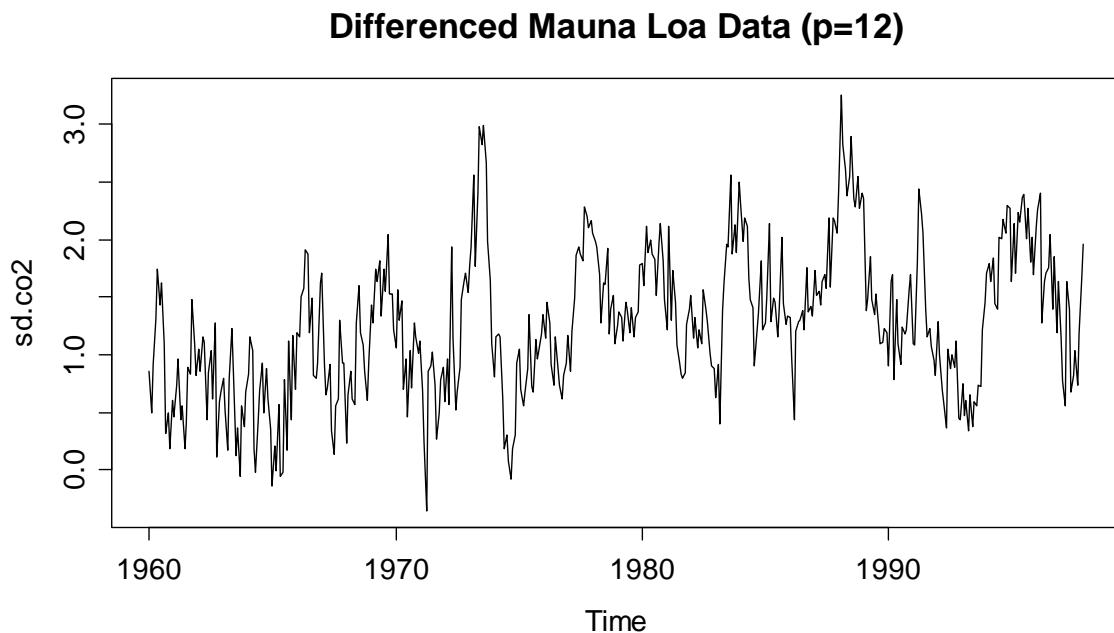
```
> data(co2)
> plot(co2, main="Mauna Loa CO2 Concentrations")
```

Seasonal differencing is very conveniently available in R. We use function `diff()`, but have to set argument `lag=....`. For the Mauna Loa data with monthly measurements, the correct lag is 12. This results in the series shown on the next page. Because we are comparing every record with the one from the previous year, the resulting series is 12 observations shorter than the original one. It is pretty obvious that some trend is remaining and thus, the result from seasonal differencing cannot be considered as stationary. As the seasonal effect is gone, we could try to add some first-order differencing at lag 1.

Mauna Loa CO2 Concentrations



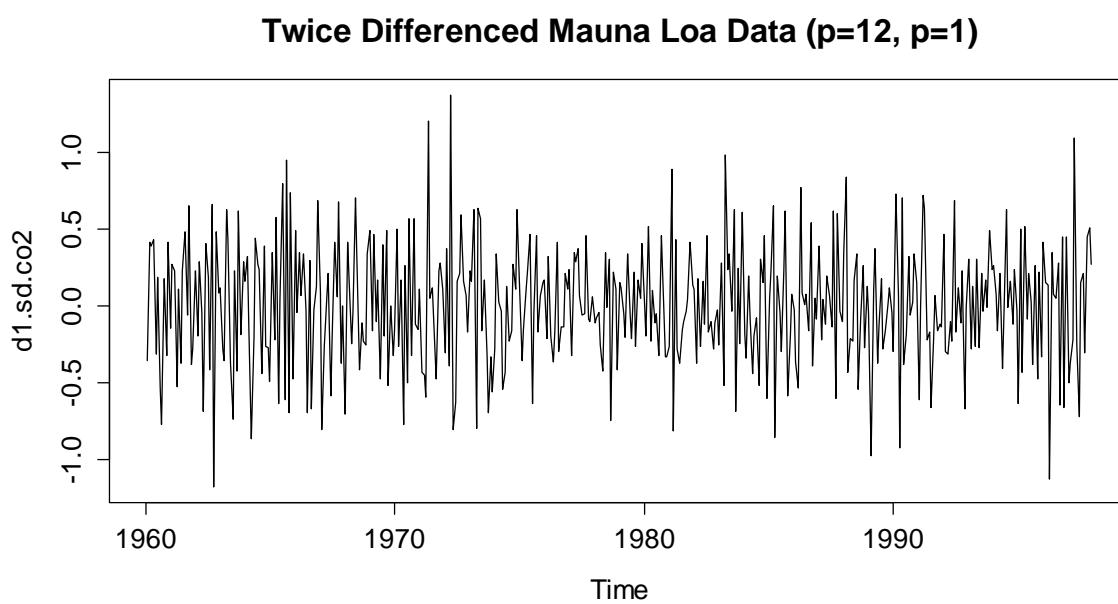
```
> sd.co2 <- diff(co2, lag=12)
> plot(sd.co2, main="Differenced Mauna Loa Data (p=12) " )
```



The second differencing step indeed manages to produce a stationary series, as can be seen below. The equation for the final series is:

$$Z_t = (1 - B)Y_t = (1 - B)(1 - B^{12})X_t .$$

The next step would be to analyze the autocorrelation of the series below and fit an $ARMA(p,q)$ model. Due to the two differencing steps, such constructs are also named *SARIMA* models. They will be discussed in chapter 6.



We conclude this section by emphasizing that while differencing is quick and simple, and (correctly done) manages to remove any trend and/or seasonality, we do not obtain explicit estimates for trend m_t , seasonal effect s_t and remainder R_t which proves problematic in many applications.

4.3.3 Smoothing, Filtering

Our next goal is to define a decomposition procedure that yields explicit trend, seasonality and remainder estimates \hat{m}_t , \hat{s}_t and \hat{R}_t . In the absence of a seasonal effect, the trend of a time series can simply be obtained by applying an *additive linear filter*:

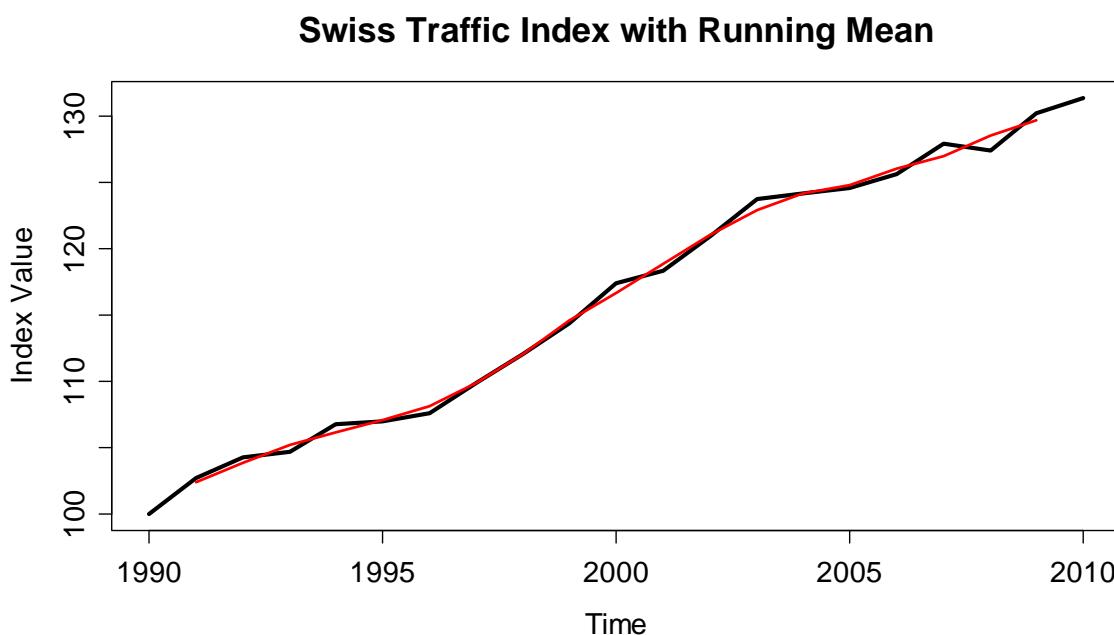
$$\hat{m}_t = \sum_{i=-p}^q a_i X_{t+i}$$

This definition is general, as it allows for arbitrary weights and asymmetric windows. The most popular implementation, however, relies on $p=q$ and $a_i=1/(2p+1)$, i.e. a *running mean estimator* with symmetric window and uniformly distributed weights. The window width is the smoothing parameter.

Example: Trend Estimation with Running Mean

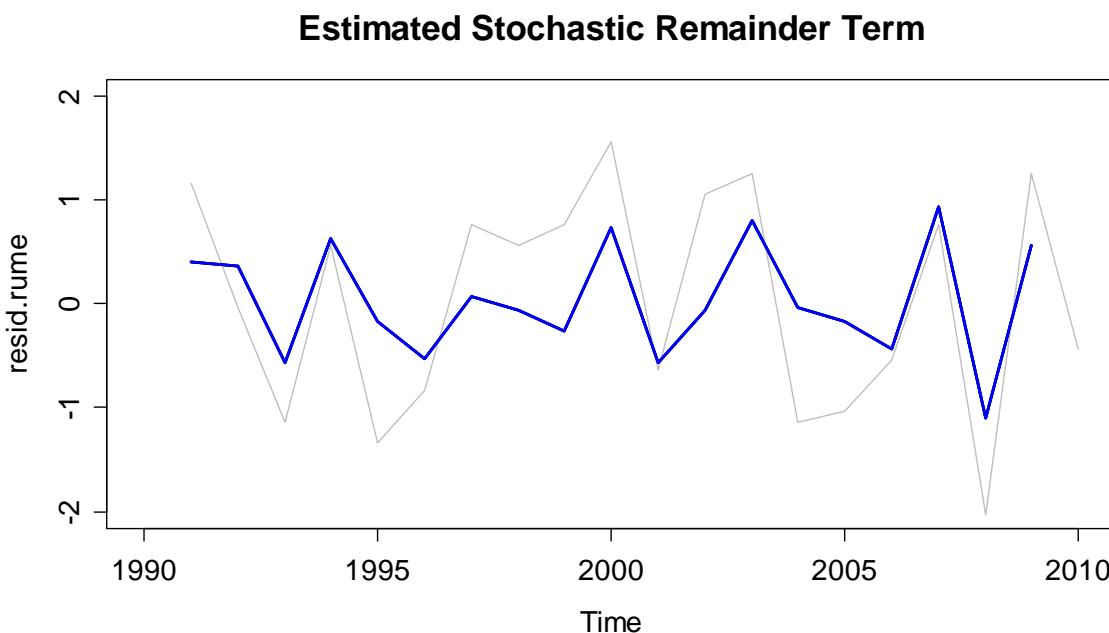
We here again consider the Swiss Traffic data that were already exhibited before. They show the indexed traffic development in Switzerland between 1990 and 2010. Linear filtering is available with function `filter()` in **R**. With the correct settings, this function becomes a running mean estimator.

```
> trend.est <- filter(SwissTraffic, filter=c(1,1,1)/3)
```



```
> trend.est
Time Series: Start = 1990, End = 2010, Frequency = 1
[1]      NA 102.3000 103.8333 105.1667 106.0667 107.0667
[7] 108.1333 109.8333 112.0667 114.5667 116.6667 118.8667
[13] 120.9667 122.9000 124.1333 124.7667 126.0333 126.9667
[19] 128.5000 129.6333      NA
```

In our example, we chose the trend estimate to be the mean over three consecutive observations. This has the consequence that for both the first and the last instance of the time series, no trend estimate is available. Also, it is apparent that the Swiss Traffic series has a very strong trend signal, whereas the remaining stochastic term is comparably small in magnitude. We can now compare the estimated remainder term from the running mean trend estimation to the result from differencing:



The blue line is the remainder estimate from running mean approach, while the grey one resulted from differencing with lag 1. We observe that the latter has bigger variance; and, while there are some similarities between the two series, there are also some prominent differences – please note that while both seem stationary, they are different.

Trend Estimation for Seasonal Data

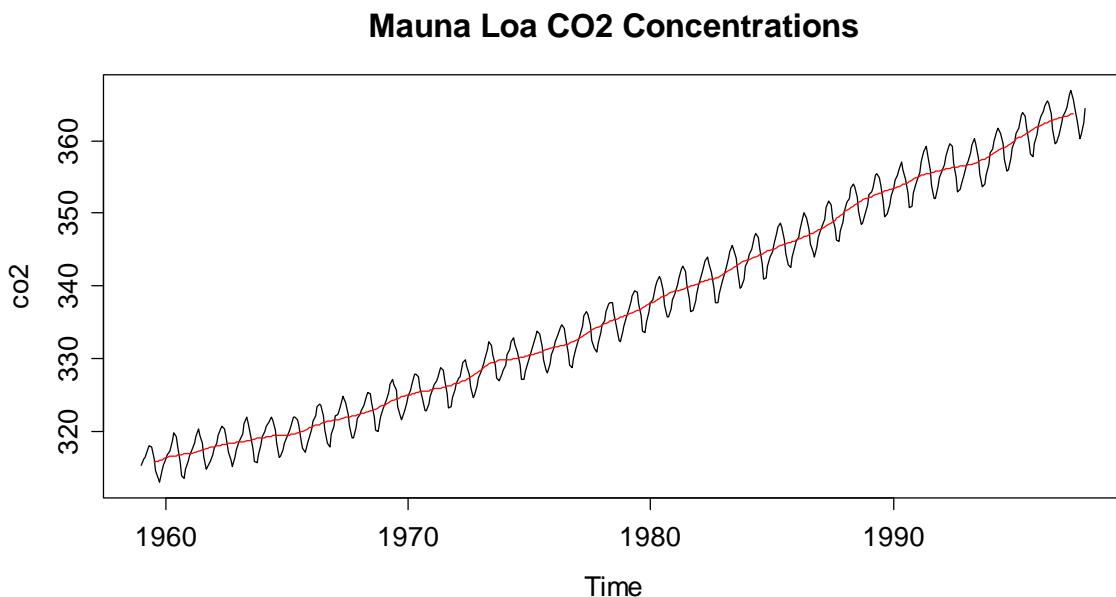
We now turn our attention to time series that show both trend *and* seasonal effect. The goal is to specify a filtering approach that allows trend estimation for periodic data. We still base this on the running mean idea, but have to make sure that we average over a full period. For monthly data, the formula is:

$$\hat{m}_t = \frac{1}{12} \left(\frac{1}{2} X_{t-6} + X_{t-5} + \dots + X_{t+5} + \frac{1}{2} X_{t+6} \right), \text{ for } t = 7, \dots, n-6$$

Be careful, as there is a slight snag if the frequency is even: if we estimate the trend for December, we use data from July to May, and then also add half of the value of the previous June, as well as half of the next June. This is required for having a window that is centered at the time we wish to estimate the trend. Using R's function `filter()`, with appropriate choice of weights, we can compute the seasonal running mean. We illustrate this with the Mauna Loa CO_2 data.

```
> wghts      <- c(.5,rep(1,11),.5)/12
> trend.est <- filter(co2, filter=wghts, sides=2)
> plot(co2, main="Mauna Loa CO2 Concentrations")
> lines(trend.est, col="red")
```

We obtain a trend which fits well to the data. It is not a linear trend, rather it seems to be slightly progressively increasing, and it has a few kinks, too.



We finish this section about trend estimation using linear filters by stating that other smoothing approaches, e.g. *running median estimation*, the *loess smoother* and many more are valid choices for trend estimation, too.

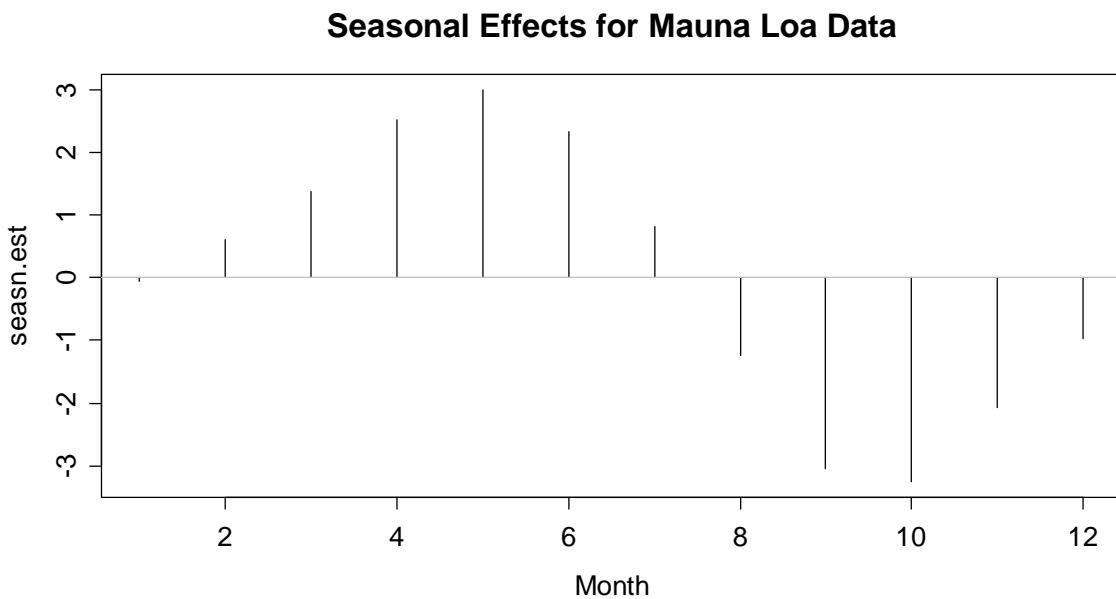
Estimation of the Seasonal Effect

For fully decomposing periodic series such as the Mauna Loa data, we also need to estimate the seasonal effect. This is done on the basis of the trend adjusted data: simple averages over all observations from the same seasonal entity are taken. The following formula shows the January effect estimation for the Mauna Loa data, a monthly series which starts in January and has 39 years of data.

$$\hat{s}_{Jan} = \hat{s}_1 = \hat{s}_{13} = \dots = \frac{1}{39} \cdot \sum_{j=0}^{38} (x_{12j+1} - \hat{m}_{12j+1})$$

In R, a convenient way of estimating such seasonal effects is by generating a factor for the months, and then using the `tapply()` function. Please note that the seasonal running mean naturally generates NA values at the start and end of the series, which we need to remove in the seasonal averaging process.

```
> trend.adj <- co2-trend.est
> month      <- factor(rep(1:12,39))
> seasn.est <- tapply(trend.adj, month, mean, na.rm=TRUE)
> plot(seasn.est, type="h", xlab="Month")
> title("Seasonal Effects for Mauna Loa Data")
> abline(h=0, col="grey")
```

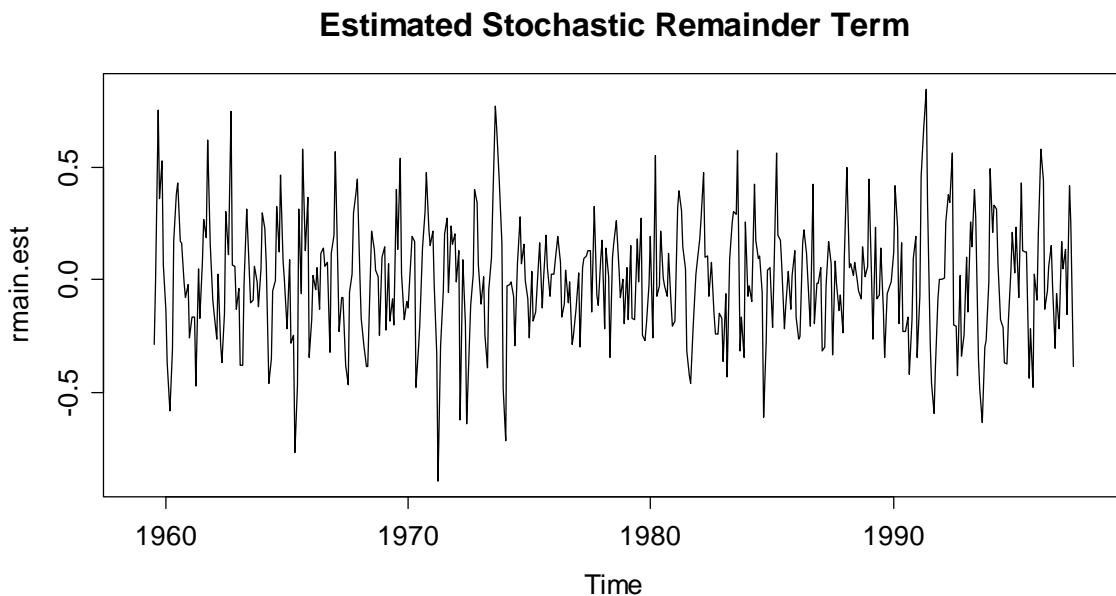


In the plot above, we observe that during a period, the highest values are usually observed in May, whereas the seasonal low is in October. The estimate for the remainder at time t is simply obtained by subtracting estimated trend and seasonality from the observed value.

$$\hat{R}_t = x_t - \hat{m}_t - \hat{s}_t$$

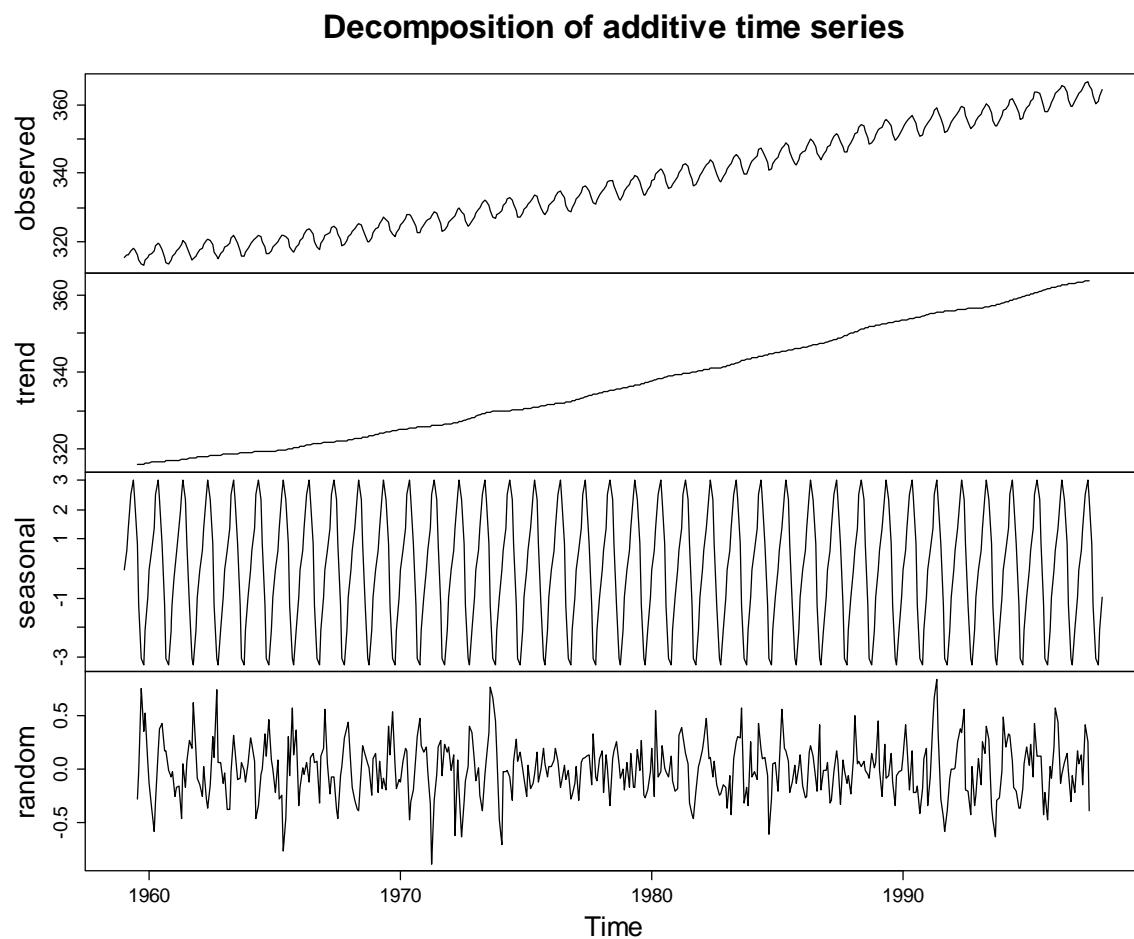
From the plot on the next page, it seems as if the estimated remainder still has some periodicity and thus it is questionable whether it is stationary. The periodicity is due to the fact that the seasonal effect is not constant but slowly evolving over time. In the beginning, we tend to overestimate it for most months, whereas in the end, we underestimate. We will address the issue on how to visualize evolving seasonality below in section 4.3.4 about STL-decomposition. A further option for dealing with non-constant seasonality is given by the exponential smoothing approach which is covered in chapter 8.3.

```
> rmain.est <- co2-trend.est-rep(seasn.est,39)
> plot(rmain.est, main="Estimated Stochastic Remainder Term")
```



Moreover, we would like to emphasize that R offers the convenient `decompose()` function for running mean estimation and seasonal averaging.

```
> co2.dec <- decompose(co2)
> plot(co2.dec)
```



Please note that `decompose()` only works with periodic series where at least two full periods were observed; else it is not mathematically feasible to estimate trend and seasonality from a series. The `decompose()` function also offers the neat plotting method shown above that generates the four frames above with the series, and the estimated trend, seasonality and remainder. Except for the different visualization, the results are exactly the same as what we had produced with our do-it-yourself approach.

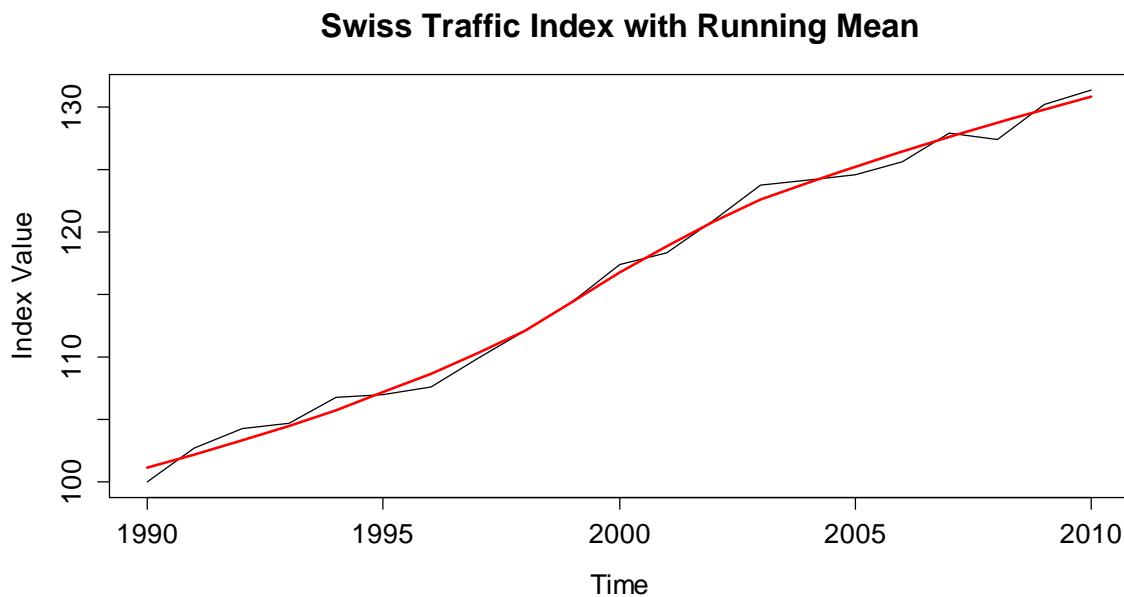
4.3.4 Seasonal-Trend Decomposition with LOESS

It is well known that the running mean is not the best smoother around. Thus, potential for improvement exists. While there is a dedicated R procedure for decomposing periodic series into trend, seasonal effect and remainder, we have to do some handwork in non-periodic cases.

Trend Estimation with LOESS

We here consider again the Swiss Traffic dataset, for which the trend had already been estimated above. Our goal is to re-estimate the trend with *LOESS*, a smoothing procedure that is based on local, weighted regression. The aim of the weighting scheme is to reduce potentially disturbing influence of outliers. Applying the LOESS smoother with (the often optimal) default settings is straightforward:

```
> fit    <- loess(SwissTraffic~time(SwissTraffic))
> trend <- predict(fit)
```

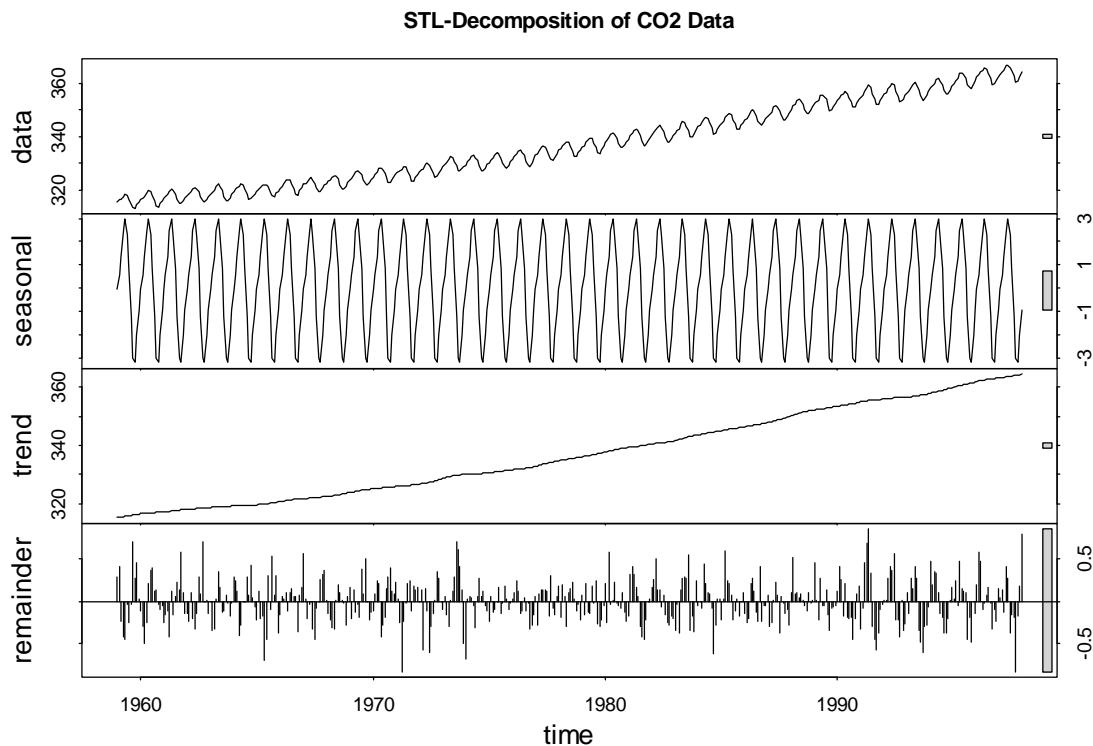


We observe that the estimated trend, in contrast to the running mean result, is now smooth and allows for interpolation within the observed time. Also, the `loess()` algorithm returns trend estimates which extend to the boundaries of the dataset. In summary, we recommend to always perform trend estimation with LOESS.

Using the `stl()` Procedure for Periodic Series

R's `stl()` procedure offers a decomposition of a periodic time series into trend, seasonality and remainder. All estimates are based on the LOESS smoother. While the output is (nearly) equivalent to what we had obtained above with `decompose()`, we recommend to use this procedure only, because the results are more trustworthy. We do here without giving the full details on the procedure, but only state that it works iteratively. The commands are as follows:

```
> co2.stl <- stl(co2, s.window="periodic")
> plot(co2.stl, main="STL-Decomposition of CO2 Data")
```

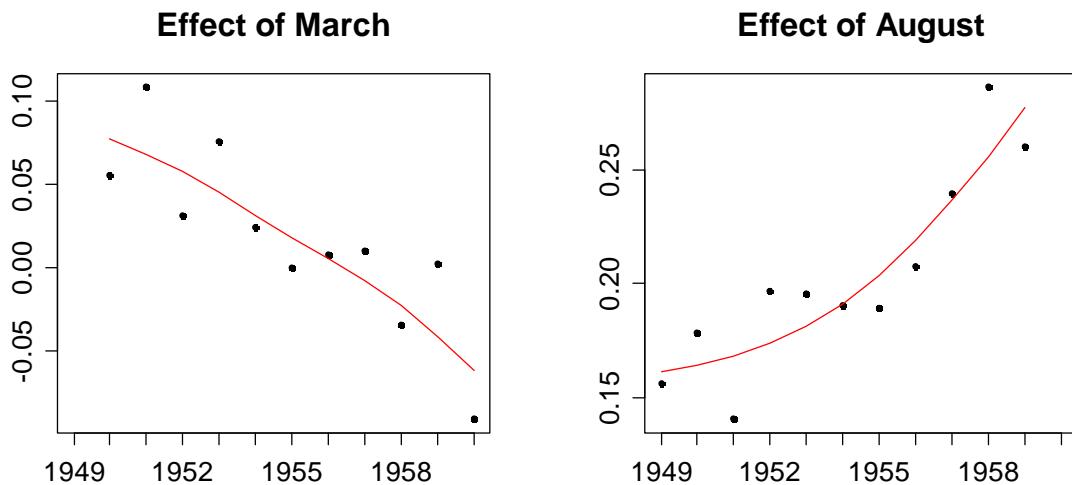


The graphical output is similar to the one from `decompose()`. The grey bars on the right hand side facilitate interpretation of the decomposition: they show the relative magnitude of the effects, i.e. cover the same span on the y-scale in all of the frames. The two principal arguments in function `stl()` are `t.window` and `s.window`: `t.window` controls the amount of smoothing for the trend, and has a default value which often yields good results. The value used can be inferred with:

```
> co2.stl$win[2]
t
19
```

The result is the number of lags used as a window for trend extraction in LOESS. Increasing it means the trend becomes smoother; lowering it makes the trend rougher, but more adapted to the data. The second argument, `s.window`, controls the smoothing for the seasonal effect. When set to "periodic" as above, the seasonality is obtained as a constant value from simple (monthly) averaging.

However, `stl()` offers better functionality. If `s.window` is set to a numeric value, the procedure can accommodate for evolving seasonality. The assumption behind is that the change in the seasonal effect happens slowly and smoothly. We visualize what is meant with the logged air passenger data. For quick illustration, we estimate the trend with a running mean filter; subtract it from the observed series and display all March and all August values of the trend adjusted series:



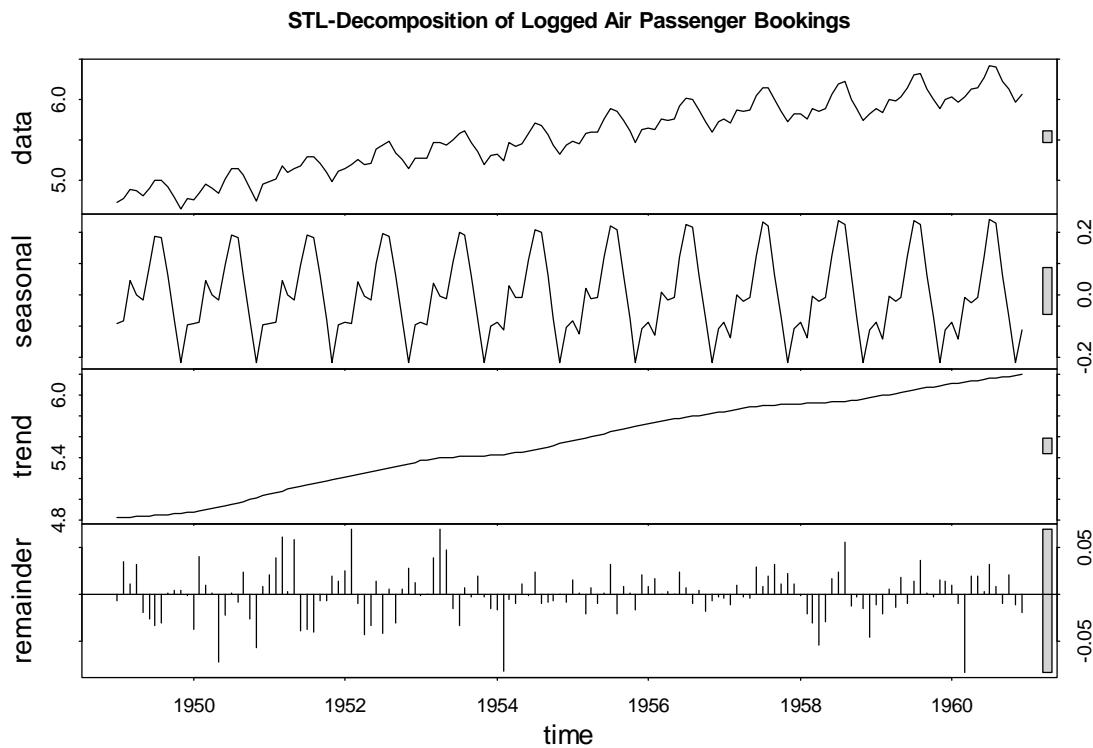
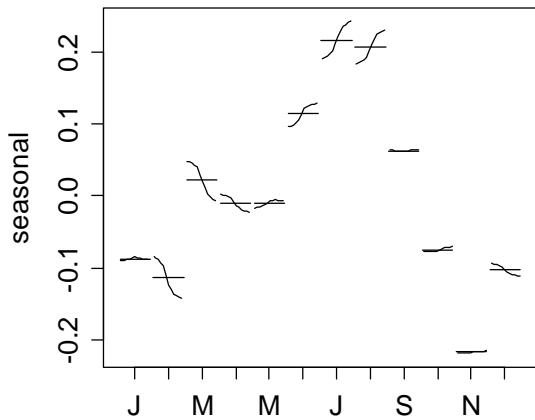
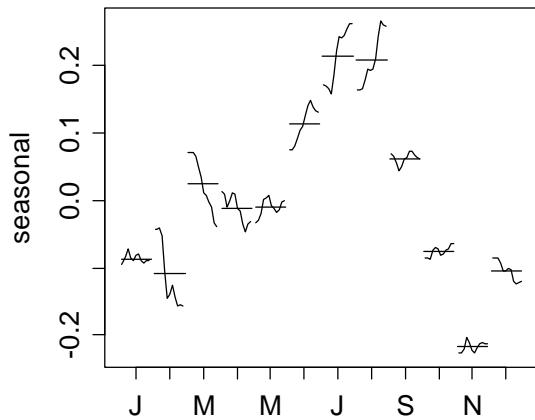
When assuming a non-changing seasonal effect, the standard procedure would be to take the mean of the data points in the above scatterplots and declare that as the seasonal effect for March and August, respectively. This is a rather crude way of data analysis, and can of course be improved. We achieve a better decomposition of the logged air passenger bookings by employing the `stl()` function and setting `s.window=13`. The resulting output is displayed below.

```
> fit.05 <- stl(lap, s.window= 5)
> fit.13 <- stl(lap, s.window=13)
> plot(fit.13, main="STL-Decomposition ...")
```

Please note that there is no default value for the seasonal span, and the optimal choice is left to the user upon visual inspection. An excellent means for doing so is the `monthplot()` function which shows the seasonal effects that were estimated by `stl()`.

```
> monthplot(fit.13, main="Monthplot, s.window=13")
> monthplot(fit.05, main="Monthplot, s.window=5")
```

The output, which can be found on the next page shows an appropriate amount of smoothing in the left panel, with `s.window=13`. However on the right, with smaller span, i.e. `s.window=5`, we observe overfitting – the seasonal effects do not evolve in a smooth way, and it means that this is not a good decomposition estimate.

**Monthplot, s.window=13****Monthplot, s.window=5**

4.3.5 Parametric Modeling

A powerful approach for decomposing time series is parametric modeling. It is based on the assumption of a functional form for the trend, usually a polynomial. For the seasonal effect, we can either use the dummy variable approach that amounts to averaging. Or, in some special cases, a sine/cosine seasonality may be appropriate. We illustrate the parametric modeling approach by two examples and use them for discussing some specifics.

We consider the Maine unemployment data from section 4.1.1. Our goal is to fit a polynomial trend, along with a seasonal effect that is obtained from averaging. We write down this model for a polynomial of grade 4.

$$X_t = \beta_0 + \beta_1 \cdot t + \beta_2 \cdot t^2 + \beta_3 \cdot t^3 + \beta_4 \cdot t^4 + \alpha_{i(t)} + E_t,$$

where $t = 1, \dots, 128$ and $i(t) \in \{1, \dots, 12\}$, i.e. $\alpha_{i(t)}$ is a factor variable encoding for the month the observation was made in, see the R code below. Two questions immediately pop up, namely what polynomial order is appropriate, and how this model can be fit.

As for the fitting, this will usually be done with the ordinary least squares (OLS) algorithm. This requires some prudence, because when working with time series, the remainder term E_t may not necessarily be stochastically independent. Thus, we might have some violated assumption for the OLS estimation. Since the estimated coefficients are still unbiased, OLS is a valid approach. However, be careful with the standard errors, as well as tests and confidence intervals derived from them, because they can be grossly misleading.

For the grade of the polynomial, we determine by eyeballing from the time series plot that the hypothesized trend in the unemployment series has at least 3 extrema. This means that a polynomial with grade below 4 will not result in a sensible trend estimate. Thus, we try orders 4, 5 and 6, and discuss how an appropriate choice can be made from residual analysis. However, we first focus on the R code for fitting such models. In the first step below, we lay the basics. From time series `maine`, we extract the times of observation, i.e. the predictor. As always when fitting polynomial regression models, it is crucial to center the x-values to mitigate potential collinearity among the terms. Furthermore, we define a factor variable for modeling the seasonality.

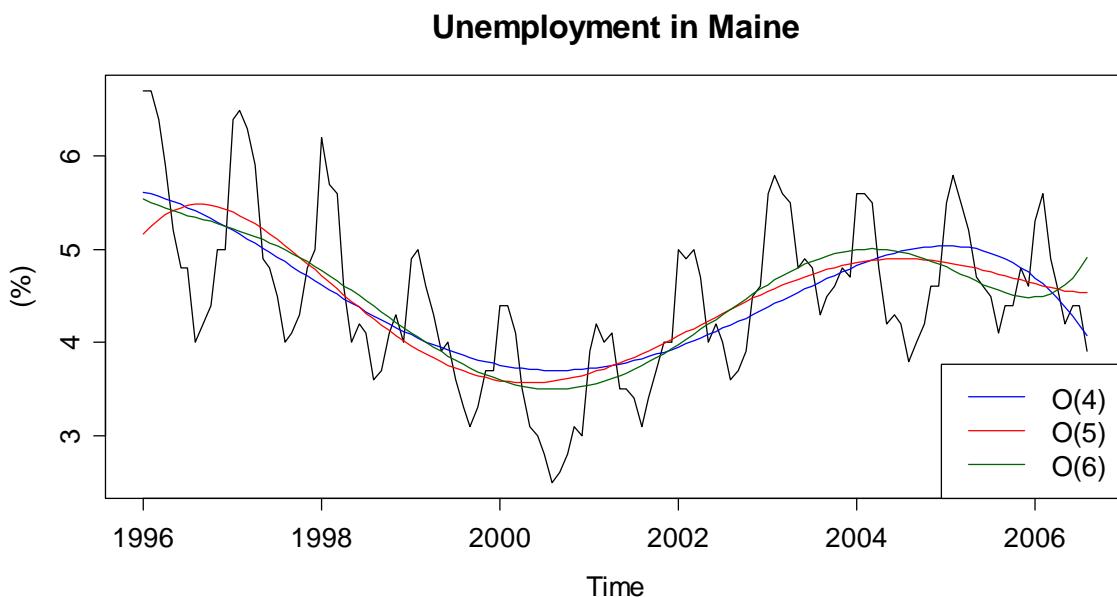
```
> maine <- ts(dat, start=c(1996,1), freq=12)
> tr    <- as.numeric(time(maine))
> tc    <- tr-mean(tr)
> mm    <- rep(c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
+                 "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))
> mm    <- factor(rep(mm,11),levels=mm)[1:128]
```

We can then obtain an OLS-fit of the decomposition model with R's `lm()` procedure, as shown below. The `I()` notation in the formula assures that the "`^`" are interpreted as arithmetical operators, i.e. powers of the predictor, rather than as formula operators. Thereafter, we can use the estimated coefficients for determining the trend estimate `t.est.04`. Because the seasonal factor uses the month of January as a reference, and thus generally has a mean different from zero, we need to shift the trend to make run through "the middle of the data".

```
> fit04    <- lm(maine~tc+I(tc^2)+I(tc^3)+I(tc^4)+mm)
> cf      <- coef(fit04)
> t.est.04 <- cf[1]+cf[2]*tc+cf[3]*tc^2+cf[4]*tc^3+cf[5]*tc^4
> t04.adj  <- t.est.04-mean(t.est.04)+mean(maine)
```

```
> plot(maine, ylab=" (%)", main="Unemployment in Maine")
> lines(tr, t.04.adj)
```

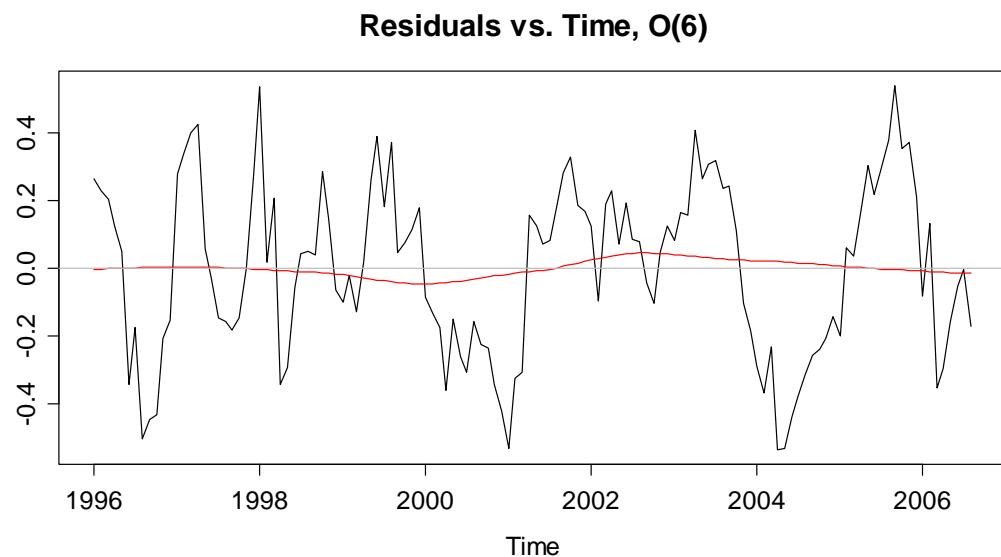
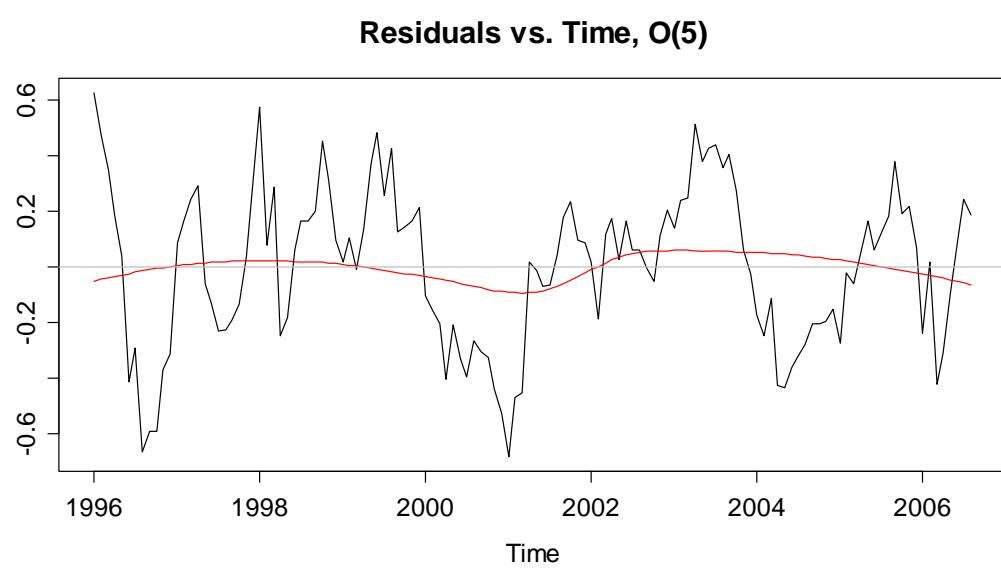
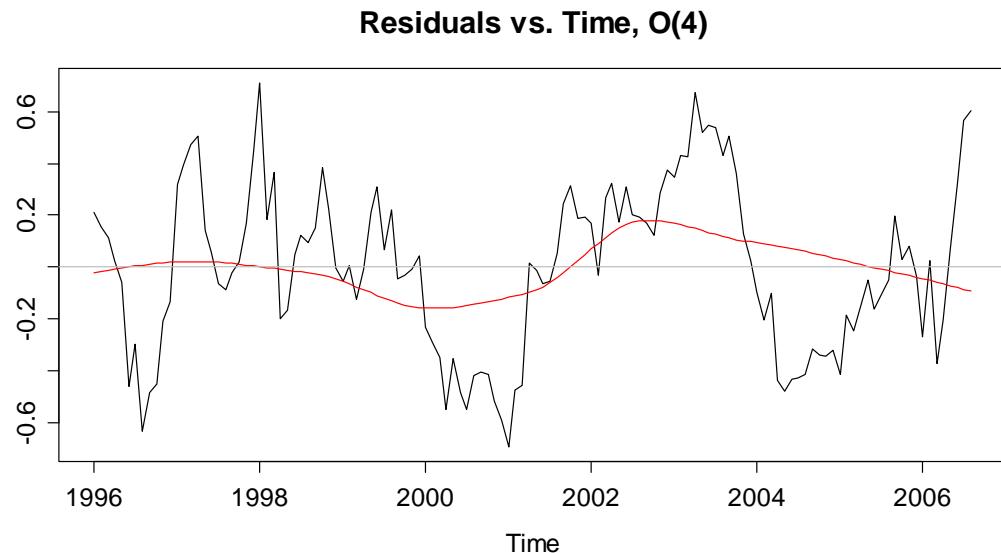
The time series plot below is enhanced with polynomial trend lines of order 4 (blue), 5 (red) and 6 (green). From this visualization, it is hard to decide which of the polynomials is most appropriate as a trend estimate. Because there are some boundary effects for orders 5 and 6, we might guess that their additional flexibility is not required. As we will see below, this is treacherous.



A better way for judging the fit of a parametric model is by residual analysis. We plot the remainder term \hat{E}_t versus time and add a LOESS smoother. The following bit of code shows this for the grade 4 polynomial.

```
> re.est <- maine-fitted(fit04)
> plot(re.est, ylab="", main="Residuals vs. Time, O(4) ")
> fit <- loess(re.est~tr)
> lines(tr, fitted(fit), col="red")
> abline(h=0, col="grey")
```

We repeat the procedure for polynomial orders 5 and 6 and display all results on the next page. In the top plot, the remainder term still features a slight trend, owing to a too low polynomial grade. The middle panel looks better, but there is another clear improvement with order 6, which is the best choice for this series. It is also striking that the remainder is not an i.i.d. series, the serial correlation is clearly standing out. Its estimation and visualization will be discussed in the next section. We conclude this chapter on parametric modeling by issuing a warning: while the explicit form of the trend can be useful, it shall never be interpreted as causal for the evolution of the series. Also, much care needs to be taken if forecasting is the goal. Extrapolating high-order polynomials beyond the range of observed times can yield very poor results. We will discuss some simple methods for trend extrapolation later in section 8 about forecasting.



4.4 Autocorrelation

An important feature of time series is their (potential) serial correlation. This section aims at analyzing and visualizing these correlations. We first display the autocorrelation between two random variables X_{t+k} and X_t , which is defined as:

$$\text{Cor}(X_{t+k}, X_t) = \frac{\text{Cov}(X_{t+k}, X_t)}{\sqrt{\text{Var}(X_{t+k})\text{Var}(X_t)}}$$

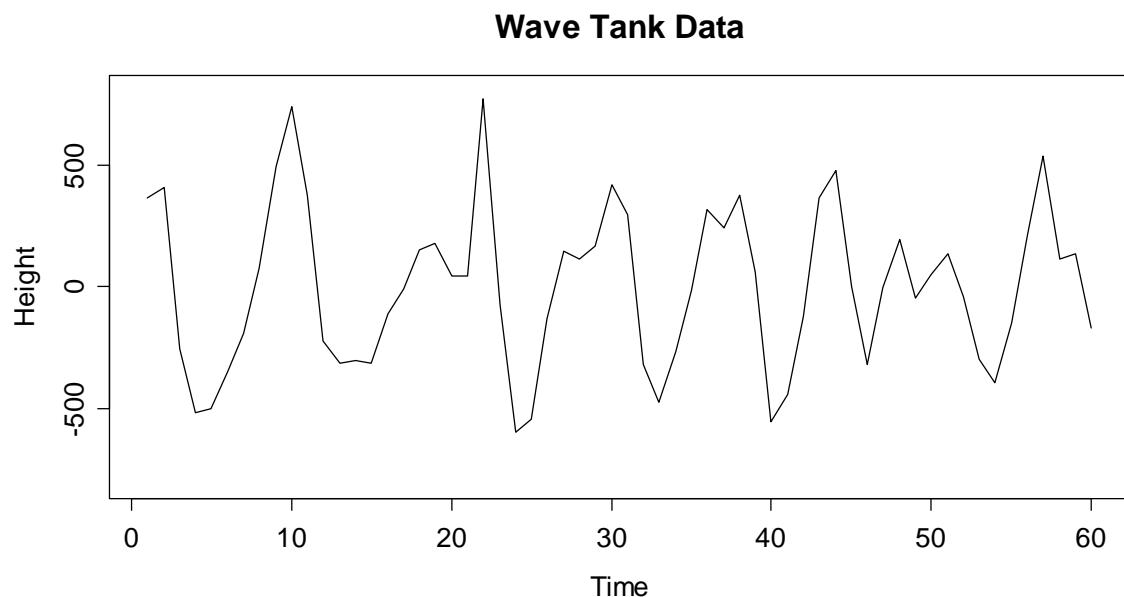
This is a dimensionless measure for the linear association between the two random variables. Since for stationary series, we require the moments to be non-changing over time, we can drop the index t for these, and write the autocorrelation as a function of the lag k :

$$\rho(k) = \text{Cor}(X_{t+k}, X_t)$$

The goals in the forthcoming sections are estimating these autocorrelations from observed time series data, and to study the estimates' properties. The latter will prove useful whenever we try to interpret sample autocorrelations in practice.

The example we consider in this chapter is the wave tank data. The values are wave heights in millimeters relative to still water level measured at the center of the tank. The sampling interval is 0.1 seconds and there are 396 observations. For better visualization, we here display the first 60 observations only:

```
> www <- "http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/"
> dat <- read.table(paste(www, "wave.dat", sep = "", header = T)
> wave <- ts(dat$waveht)
> plot(window(wave, 1, 60), ylim = c(-800, 800), ylab = "Height")
> title("Wave Tank Data")
```

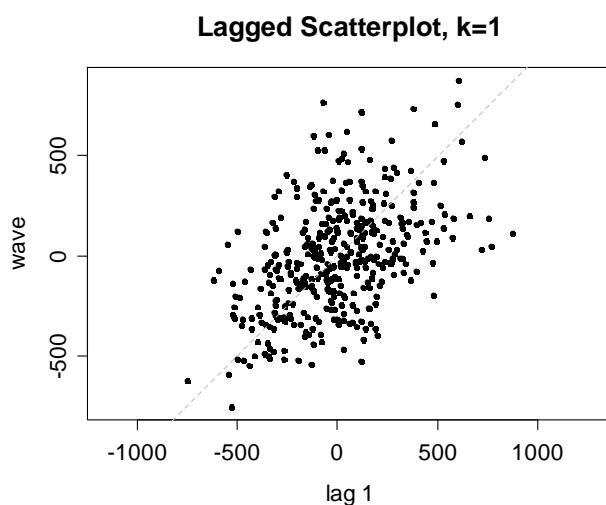


These data show some pronounced cyclic behavior. This does not come as a surprise, as we all know from personal experience that waves do appear in cycles. The series shows some very clear serial dependence, because the current value is quite closely linked to the previous and following ones. But very clearly, it is also a stationary series.

4.4.1 Lagged Scatterplot

An appealing idea for analyzing the correlation among consecutive observations in the above series is to produce a scatterplot of (x_t, x_{t+1}) for all $t = 1, \dots, n-1$. There is a designated function `lag.plot()` in **R**. The result is as follows:

```
> lag.plot(wave, do.lines=FALSE, pch=20)
> title("Lagged Scatterplot, k=1")
```



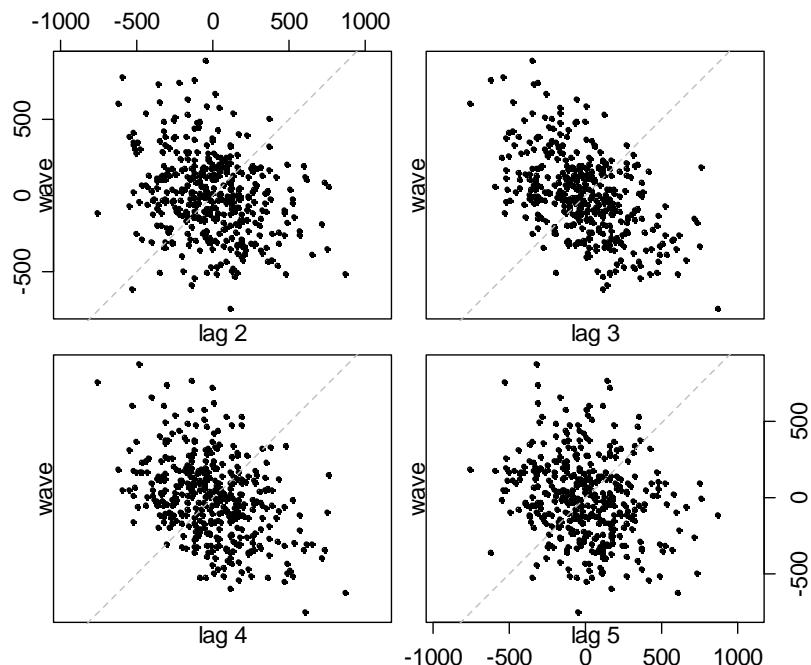
The association seems linear and is positive. The Pearson correlation coefficient turns out to be 0.47, thus moderately strong. How to interpret this value from a practical viewpoint? Well, the square of the correlation coefficient, $0.47^2 = 0.22$, is the percentage of variability explained by the linear association between x_t and its respective predecessor. Here in this case, x_{t-1} explains roughly 22% of the variability observed in x_t .

We can of course extend the very same idea to higher lags. We here analyze the lagged scatterplot correlations for lags $k = 2, \dots, 5$, see below. When computed, the estimated Pearson correlations turn out to be -0.27, -0.50, -0.39 and -0.22, respectively. The formula for computing them is:

$$\tilde{\rho}(k) = \frac{\sum_{s=1}^{n-k} (x_{s+k} - \bar{x}_{(k)})(x_s - \bar{x}_{(1)})}{\sqrt{\sum_{s=k+1}^n (x_s - \bar{x}_{(k)})^2 \cdot \sum_{t=1}^{n-k} (x_t - \bar{x}_{(1)})^2}} \text{ for } k = 1, \dots, n-2,$$

where $\bar{x}_{(1)} = \frac{1}{n-k} \sum_{i=1}^{n-k} x_i$ and $\bar{x}_{(k)} = \frac{1}{n-k} \sum_{i=k+1}^n x_i$

It is important to notice that while there are $n-1$ data pairs for computing $\tilde{\rho}(1)$, there are only $n-2$ for $\tilde{\rho}(2)$, and then less and less, i.e. $n-k$ pairs for $\tilde{\rho}(k)$. Thus for the last autocorrelation coefficient which can be estimated, $\tilde{\rho}(n-2)$, there is only one single data pair which is left. Of course, they can always be interconnected by a straight line, and the correlation in this case is always ± 1 . Of course, this is an estimation snag, rather than perfect linear association for the two random variables.



Intuitively, it is clear that because there are less and less data pairs at higher lags, the respective estimated correlations are less and less precise. Indeed, by digging deeper in mathematical statistics, one can prove that the variance of $\tilde{\rho}(k)$ increases with k . This is undesired, as it will lead to unstable results and spurious effects. The remedy is discussed in the next section.

4.4.2 Plug-In Estimation

For mitigating the above mentioned problem with the lagged scatterplot method, autocorrelation estimation is commonly done using the so-called plug-in approach, using estimated autocovariances as the basis. The formula is as follows:

$$\hat{\rho}(k) = \frac{\hat{\gamma}(k)}{\hat{\gamma}(0)}, \text{ for } k = 1, \dots, n-1,$$

where $\hat{\gamma}(k) = \frac{1}{n} \sum_{s=1}^{n-k} (x_{s+k} - \bar{x})(x_s - \bar{x})$, with $\bar{x} = \frac{1}{n} \sum_{t=1}^n x_t$.

Note that here, n is used as a denominator irrespective of the lag and thus the number of summands. This has the consequence that $\hat{\gamma}(0)$ is not an unbiased estimator for $\gamma(0) = \sigma_x^2$, but as explained above, there are good reasons to do so. When plugging in the above terms, the estimate for the k th autocorrelation coefficient turns out to be:

$$\hat{\rho}(k) = \frac{\sum_{s=1}^{n-k} (x_{s+k} - \bar{x})(x_s - \bar{x})}{\sum_{t=1}^n (x_t - \bar{x})^2}, \text{ for } k = 1, \dots, n-1.$$

It is straightforward to compute these in R, function `acf()` does the job, and we below do so for the wave tank data. As for the moment, we are interested in the numerical results, we set argument `plot=FALSE`. However, as we will see below, it is usually better to visualize the estimated autocorrelation coefficients graphically, as it will be explained below in section 4.4.3. Also note that R by default does not return all autocorrelations which are estimable in this series with 396 observations, but only the first 25.

```
> acf(wave, plot=FALSE)
```

Autocorrelations of series 'wave', by lag

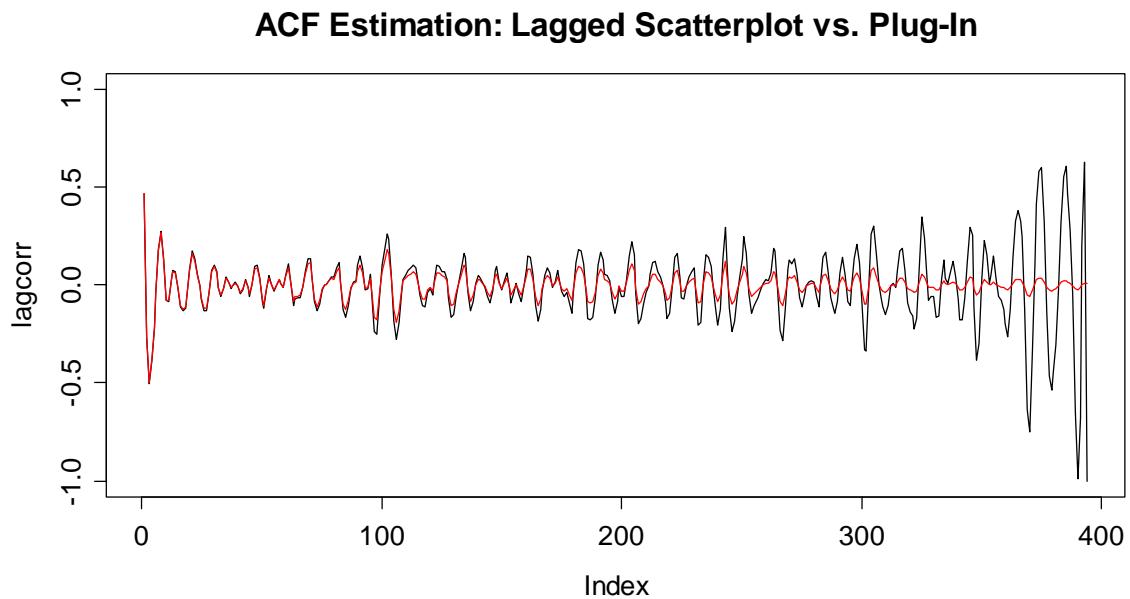
0	1	2	3	4	5	6	7
1.000	0.470	-0.263	-0.499	-0.379	-0.215	-0.038	0.178
8	9	10	11	12	13	14	15
0.269	0.130	-0.074	-0.079	0.029	0.070	0.063	-0.010
16	17	18	19	20	21	22	23
-0.102	-0.125	-0.109	-0.048	0.077	0.165	0.124	0.049
24	25						
-0.005	-0.066						

Next, we compare the autocorrelations from lagged scatterplot estimation vs. the ones from the plug-in approach. These are displayed on the next page. While for the first 50 lags, there is not much of a difference, the plug-in estimates are much more damped for higher lags. As claimed above, the lagged scatterplot estimate shows a value of -1 for lag 394, and some generally very erratic behavior in the few lags before.

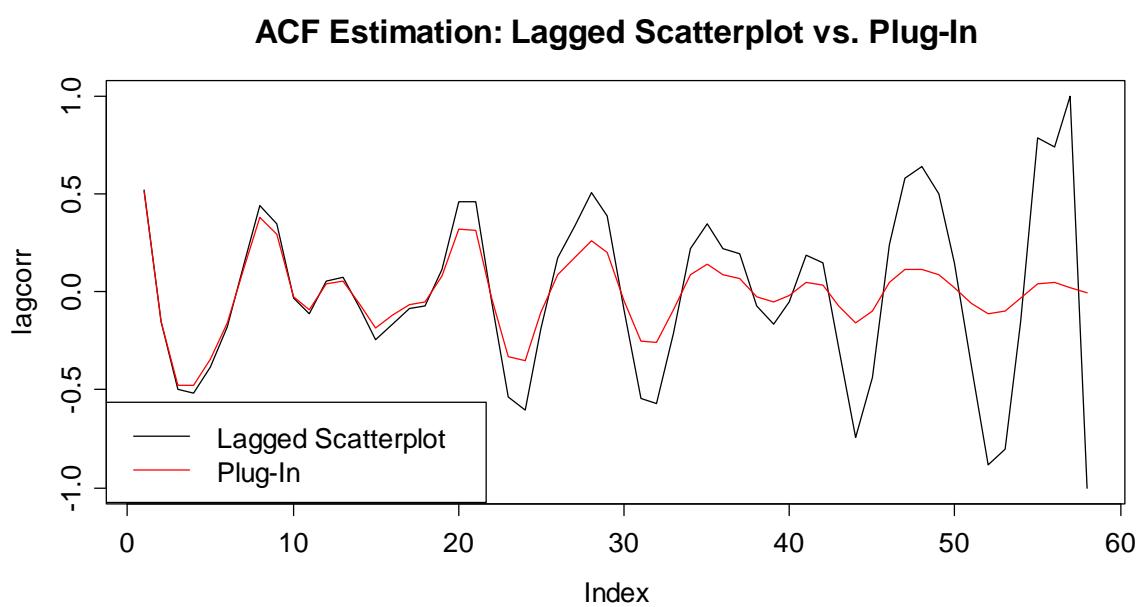
We can “prove”, or rather, provide evidence that this is an estimation artifact only if we restrict the series to the first 60 observations and then repeat the estimation of autocorrelations. Again, for the highest few lags which are estimable, the lagged scatterplot approach shows erratic behavior – and this was not present at the same lags, when the series was still longer. We do not observe this kind of effect with the plug-in based autocorrelations, thus this is clearly the method of choice.

We finish this chapter by repeating that the bigger the lag, the fewer data pairs remain for estimating the autocorrelation coefficient. We discourage of the use of the lagged scatterplot approach. While the preferred plug-in approach is biased

due to the built-in damping mechanism, i.e. the estimates for high lags are shrunken towards zero; it can be shown that it has lower mean squared error. This is because it produces results with much less (random) variability. It can also be shown that the plug-in estimates are consistent, i.e. the bias disappears asymptotically.



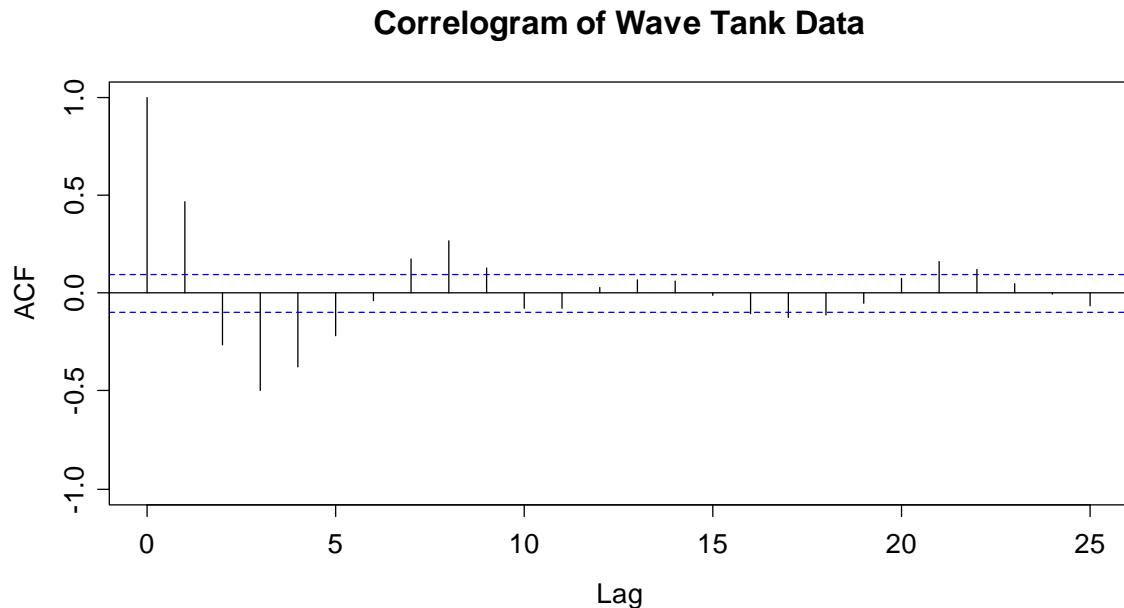
Nevertheless, all our findings still suggest that it is a good idea to consider only a first portion of the estimated autocorrelations. A rule of the thumb suggests that $10 \cdot \log_{10}(n)$ is a good threshold. For a series with 100 observations, the threshold becomes lag 20. A second rule operates with $n/4$ as the maximum lag to which the autocorrelations are shown.



4.4.3 Correlogram

Now, we know how to estimate the autocorrelation function (ACF) for any lag k . Here, we introduce the *correlogram*, the standard means of visualization for the ACF. We will then also study the properties of the ACF estimator. We employ **R** and obtain:

```
> acf(wave, ylim=c(-1,1))
```



It has become a widely accepted standard to use vertical spikes for displaying the estimated autocorrelations. Also note that the ACF starts with lag 0, at which it always takes the value 1. For better judgment, we also recommend setting the y -range to the interval $[-1,1]$. Apart from these technicalities, the ACF reflects the properties of the series. We also observe a cyclic behavior with a period of 8, as it is apparent in the time series plot of the original data. Moreover, the absolute value of the correlations attenuates with increasing lag. Next, we will discuss the interpretation of the correlogram.

Confidence Bands

It is obvious that even for an iid series without any serial correlation, and thus $\rho(k)=0$ for all k , the estimated autocorrelations $\hat{\rho}(k)$ will generally not be zero. Hopefully, they will be close, but the question is how close. An answer is indicated by the confidence bands, i.e. the blue dashed lines in the plot above.

These so-called confidence bands are obtained from an asymptotic result: for long iid time series it can be shown that the $\hat{\rho}(k)$ approximately follow a $N(0,1/n)$ distribution. Thus, each $\rho(k)$ lies within the interval of $\pm 1.96/\sqrt{n}$ with a probability of approximately 95%. This leads us to the following statement that facilitates interpretation of the correlogram: “*for any stationary time series, sample autocorrelation coefficients $\hat{\rho}(k)$ that fall within the confidence band $\pm 2/\sqrt{n}$ are*

considered to be different from 0 only by chance, while those outside the confidence band are considered to be truly different from 0 .”

On the other hand, the above statement means that even for iid series, we expect 5% of the estimated ACF coefficients to exceed the confidence bounds; these correspond to type 1 errors. Please note again that the indicated bounds are asymptotic and derived from iid series. The properties of serially dependent series are much harder to derive.

Ljung-Box Test

The Ljung-Box approach tests the null hypothesis that a number of autocorrelation coefficients are simultaneously equal to zero. Or, more colloquially, it evaluates whether there is any significant autocorrelation in a series. The test statistic is:

$$Q(h) = n \cdot (n+2) \cdot \sum_{k=1}^h \frac{\hat{\rho}_k^2}{n-k}$$

Here, n is the length of the time series, $\hat{\rho}_k$ are the sample autocorrelation coefficients at lag k and h is the lag up to which the test is performed. It is typical to use $h=1, 3, 5, 10$ or 20 . The test statistic asymptotically follows a χ^2 distribution with h degrees of freedom. As an example, we compute the test statistic and the respective p-value for the wave tank data with $h=10$.

```
> nn <- length(wave)
> qq <- nn*(nn+2)*sum((acf(wave)$acf[2:11]^2)/(nn-(1:10)))
> qq
[1] 344.0155
> 1-pchisq(qq, 10)
[1] 0
```

We observe that $Q(10) = 344.0155$ which is far in excess of what we would expect by chance on independent data. The critical value, i.e. the 95%-quantile of the χ^2_{10} is at 18.3 and thus, the p-value is close to (but not exactly) zero. There is also a dedicated R function which can be used to perform Ljung-Box testing:

```
> Box.test(wave, lag=10, type="Ljung-Box")

Box-Ljung test

data: wave

X-squared = 344.0155, df = 10, p-value < 2.2e-16
```

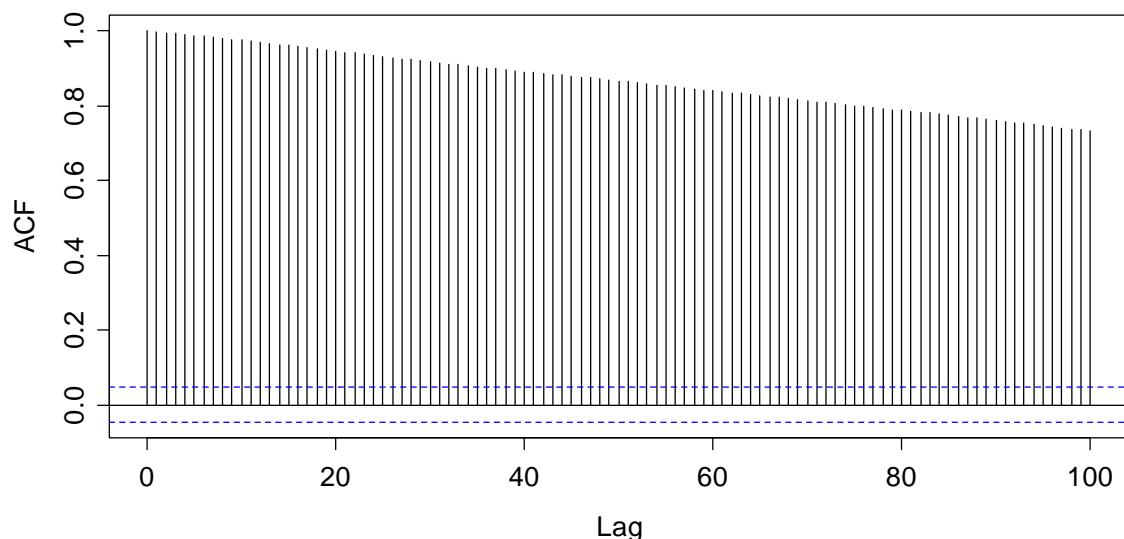
The result is, of course, identical. Please be aware that the test is sometimes also referred to as Box-Ljung test. Also R is not very consistent in its nomenclature. However, the two are one and the same. Moreover, with a bit of experience the results of the Ljung-Box test can usually be guessed quite well from the correlogram by eyeballing.

ACF of Non-Stationary Series

Estimation of the ACF from an observed time series assumes that the underlying process is stationary. Only then we can treat pairs of observations at lag k as being probabilistically “equal” and compute sample covariance coefficients. Hence, while stationarity is at the root of ACF estimation, we can of course still apply the formulae given above to non-stationary series. The ACF then usually exhibits some typical patterns. This can serve as a second check for non-stationarity, i.e. helps to identify it, should it have gone unnoticed in the time series plot. We start by showing the correlogram for the SMI daily closing values from section 1.2.4. This series does not have seasonality, but a very clear trend.

```
> acf(smi, lag.max=100)
```

Correlogram of SMI Daily Closing Values

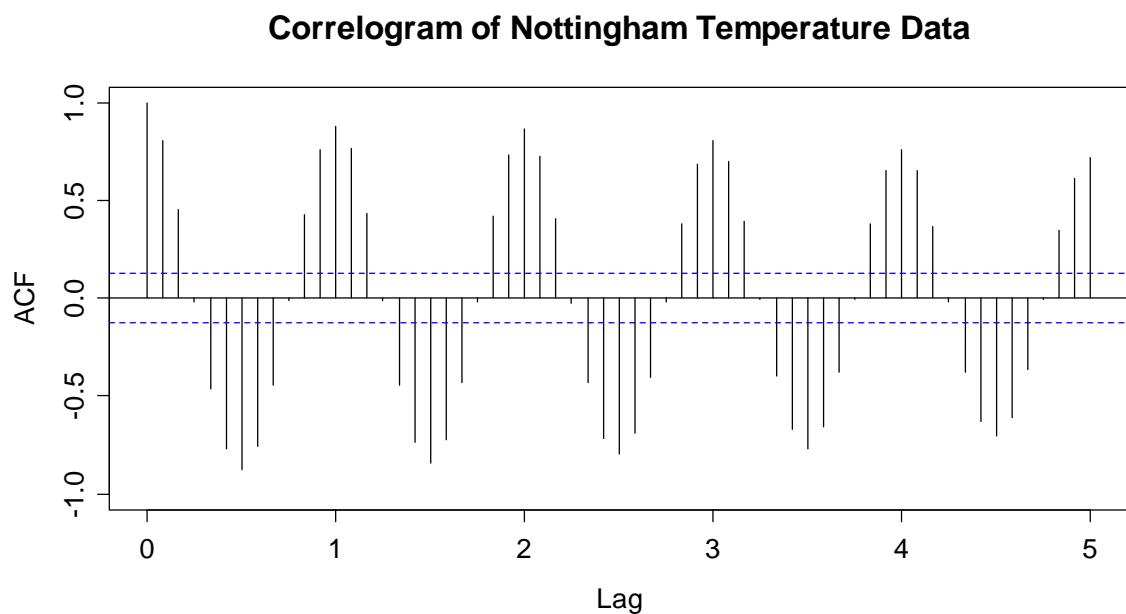
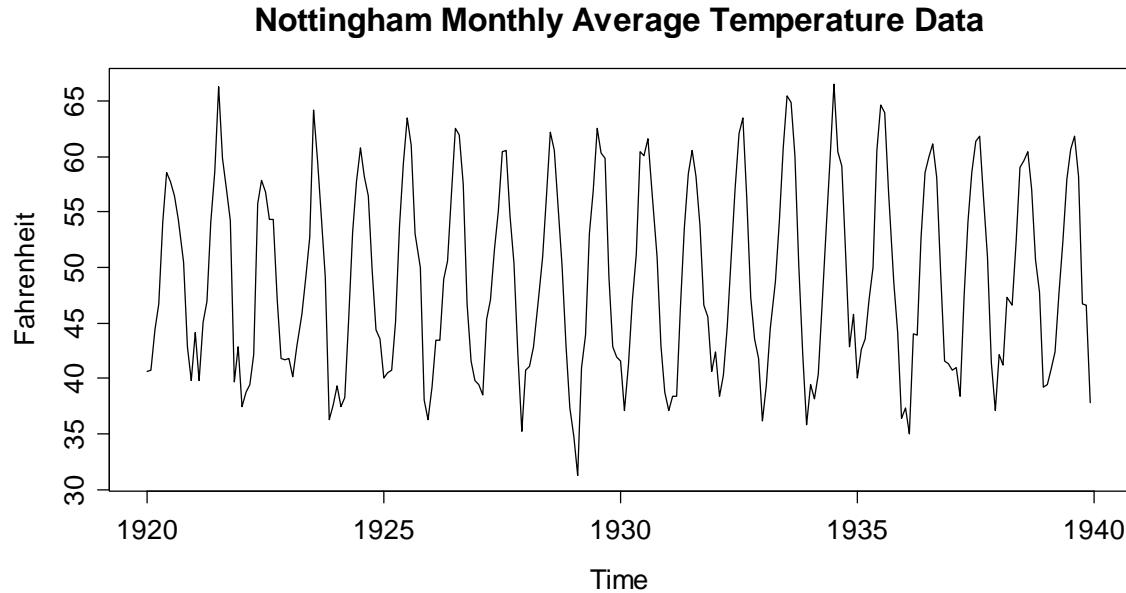


We observe that the ACF decays very slowly. The reason is that if a time series features a trend, the observations at consecutive observations will usually be on the same side of the series’ global mean \bar{x} . This is why that for small to moderate lags k , most of the terms

$$(x_{s+k} - \bar{x})(x_s - \bar{x})$$

are positive. For this reason, the sample autocorrelation coefficient will be positive as well, and is most often also close to 1. Thus, a very slowly decaying ACF is an indicator for non-stationarity, i.e. a trend which was not removed before autocorrelations were estimated.

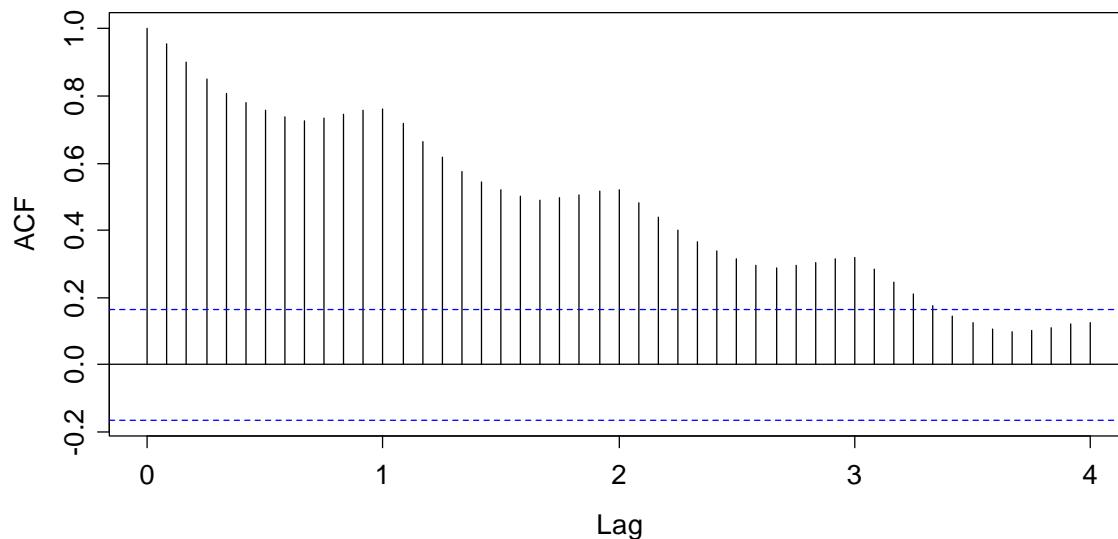
Next, we show an example of a series that has no trend, but a strongly recurring seasonal effect. We use R’s `data(nottem)`, a time series containing monthly average air temperatures at Nottingham Castle in England from 1920-1939. Time series plot and correlogram are as follows:



The ACF is cyclic, and owing to the recurring seasonality, the envelope again decays very slowly. Also note that for periodic series, R has periods rather than lags on the x-axis – often a matter of confusion. We conclude that a hardly, or very slowly decaying periodicity in the correlogram is an indication of a seasonal effect which was forgotten to be removed. Finally, we also show the correlogram for the logged air passenger bookings. This series exhibits both an increasing trend and a seasonal effect. The result is as follows:

```
> data(AirPassengers)
> txt <- "Correlogram of Logged Air Passenger Bookings"
> acf(log(AirPassengers), lag.max=48, main=txt)
```

Correlogram of Logged Air Passenger Bookings

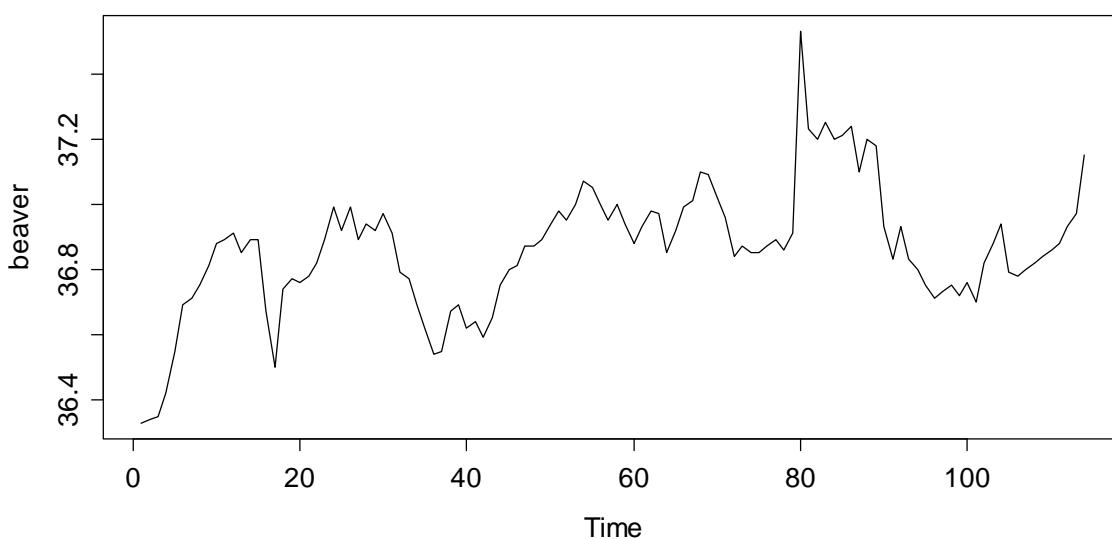


Here, the two effects described above are interspersed. We have a (here dominating) slow decay in the general level of the ACF, plus some periodicity. Again, this is an indication for a non-stationary series. It needs to be decomposed, before the serial correlation in the stationary remainder term can be studied.

The ACF and Outliers

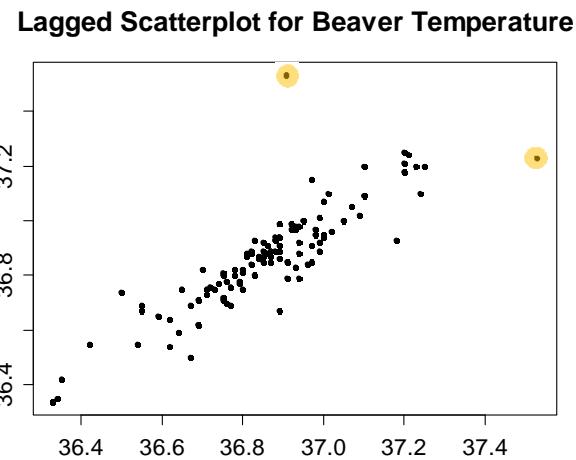
If a time series has an outlier, it will appear twice in any lagged scatterplot, and will thus potentially have “double” negative influence on the $\hat{\rho}(k)$. As an example, we consider variable `temp` from data frame `beaver1`, which can be found in R’s `data(beavers)`. This is the body temperature of a female beaver, measured by telemetry in 10 minute intervals. We first visualize the data with a time series plot.

Beaver Body Temperature Data



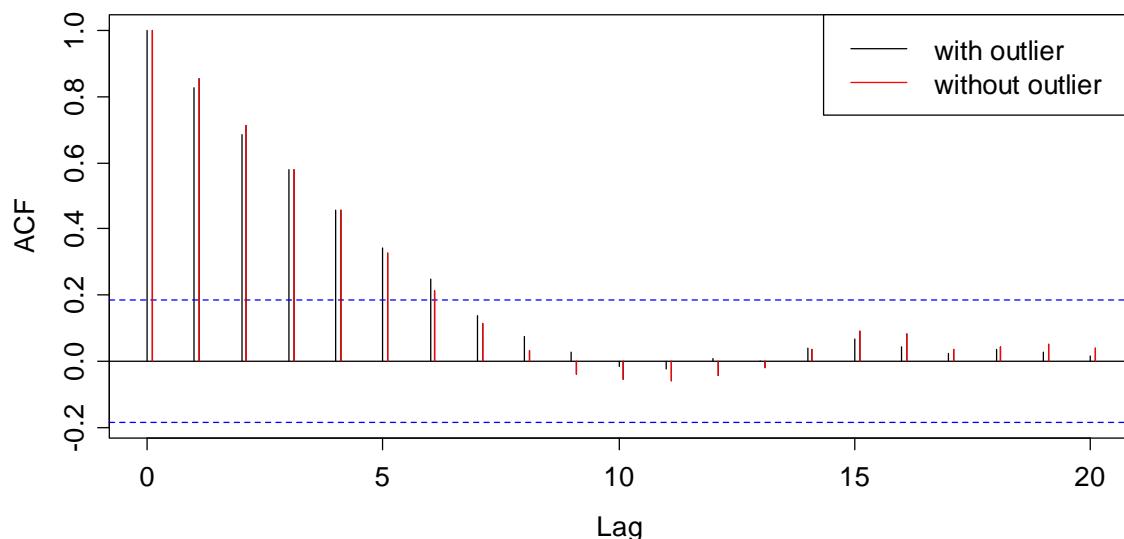
Observation 80 is a moderate, but distinct outlier. It is unclear to the author whether this actually is an error, or whether the reported value is correct. But because the purpose of this section is showing the potential influence of erroneous values, that is not important. Neither the Pearson correlation coefficient, nor the plug-in autocorrelation estimator is robust, thus the appearance of the correlogram can be altered quite strongly due to the presence of just one single outlier.

```
> plot(beaver[1:113], beaver[2:114], pch=20, )
> title("Lagged Scatterplot for Beaver Temperature")
```



The two data points where the outlier is involved are highlighted. The Pearson correlation coefficients with and without these two observations are 0.86 and 0.91. Depending on the outliers severity, the difference can be much bigger. The next plot shows the entire correlogram for the beaver data, computed with (black) and without (red) the outlier. Also here, the difference may seem small and rather academic, but it could easily be severe if the outlier was just pronounced enough.

Correlogram of Beaver Temperature Data



The question is, how do we handle missing values in time series? In principle, we cannot just omit them without breaking the time structure. And breaking it means going away from our paradigm of equally spaced points in time. A popular choice is thus replacing the missing value. This can be done with various degrees of sophistication:

- a) replacing the value with the global mean
- b) using a local mean, i.e. +/- 3 observations
- c) model based imputation by forecasting

The best strategy depends upon the case at hand. And in fact, there is a fourth alternative: while R's `acf()` function by default does not allow for missing values, it still offers the option to proceed without imputation. If argument is set as `na.action=na.pass`, the covariances are computed from the complete cases, and the correlogram is shown as usual. However, having missed values in the series has the consequence that the estimates produced may well not be a valid (i.e. positive definite) autocorrelation sequence, and may contain missing values. From a practical viewpoint, these drawbacks can often be neglected, though.

4.4.4 Quality of ACF Estimates

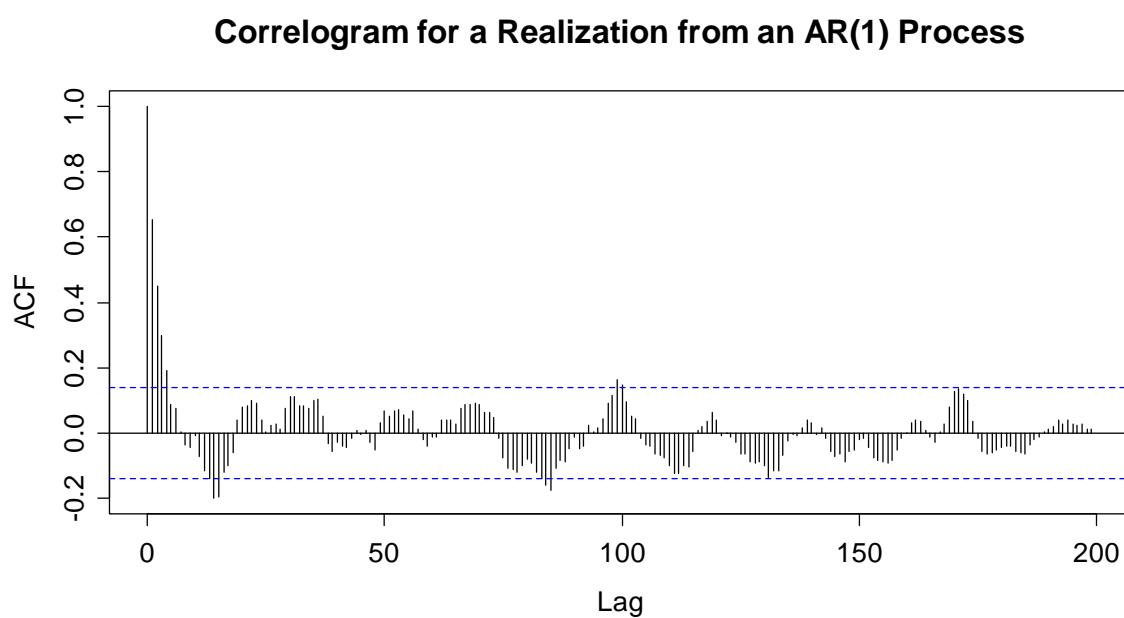
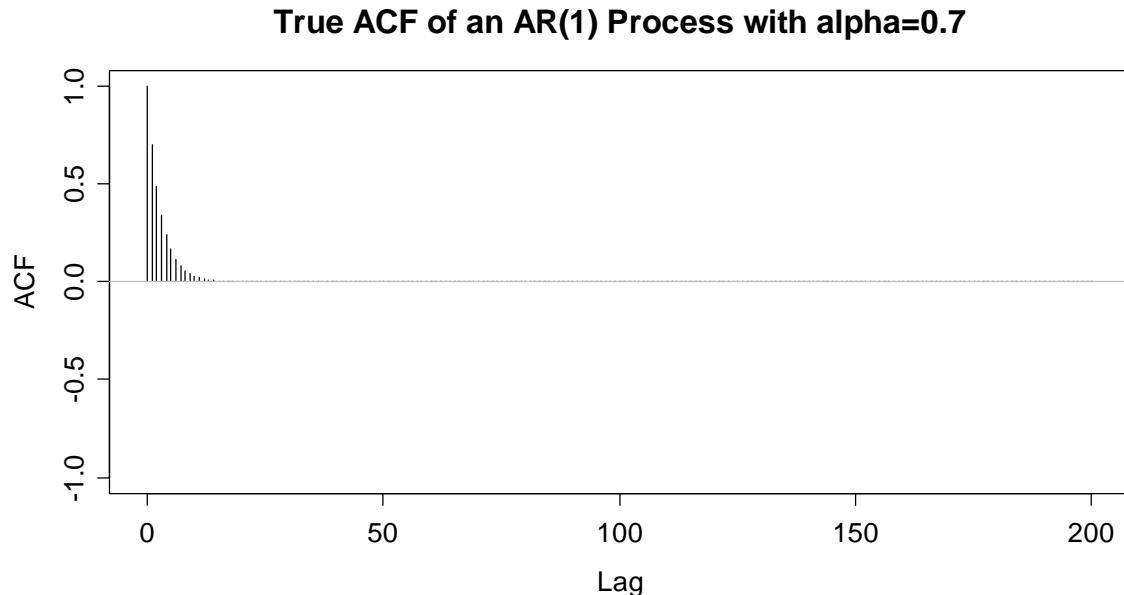
In this section we will deal with the quality of the information that is contained in the correlogram. We will not do this from a very theoretical viewpoint, but rather focus on the practical aspects. We have already learned that the ACF estimates are generally biased, i.e. shrunken towards zero for higher lags. This means that it is better to cut off the correlogram at a certain lag. Furthermore, non-stationarities in the series can hamper the interpretation of the correlogram and we have also seen that outliers can have a quite strong impact. But there are even more aspects in ACF estimation that are problematic...

The Compensation Issue

One can show that the sum of all autocorrelations which can be estimated from a time series realization, the sum over all $\hat{\rho}(k)$ for lags $k = 1, \dots, n - 1$, adds up to $-1/2$. Or, written as a formula:

$$\sum_{k=1}^{n-1} \hat{\rho}(k) = -\frac{1}{2}$$

We omit the proof here. It is clear that the above condition will lead to quite severe artifacts, especially when a time series process has only positive correlations. We here show both the true, theoretical ACF of an $AR(1)$ process with $\alpha_1 = 0.7$, which, as we will see in section 5, has $\rho(k) > 0$ for all k , and the sample correlogram for a realization of that process with a length 200 observations.



The respective R-commands for producing these plots are as follows:

```

## True ACF
true.acf <- ARMAacf(ar=0.7, lag.max=200)
plot(0:200, true.acf, type="h", xlab="Lag", ylim=c(-1,1))
title("True ACF of an AR(1) Process with alpha=0.7")
abline(h=0, col="grey")

## Simulation and Generating the ACF
set.seed(25)
ts.simul <- arima.sim(list(ar=0.7), 200)
acf(ts.simul, lag=200, main="Correlogram ... ")

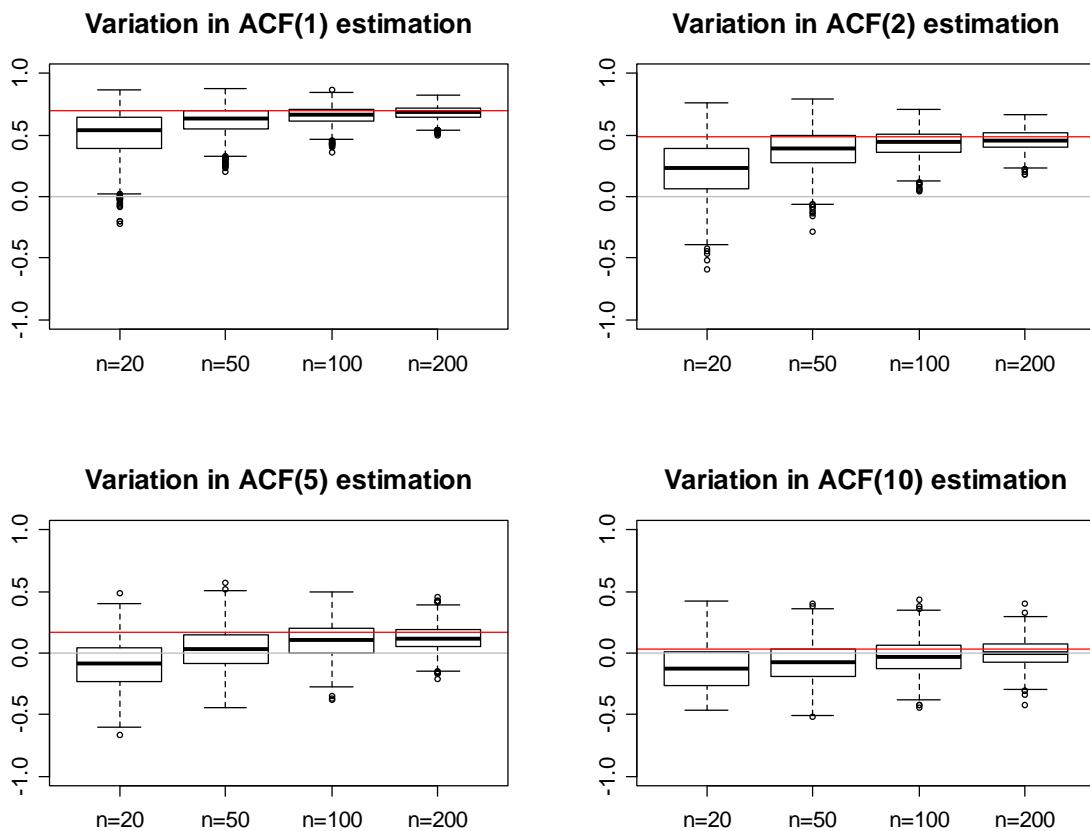
```

What we observe is quite striking: only for the very first few lags, the sample ACF does match with its theoretical counterpart. As soon as we are beyond lag $k = 6$, the sample ACF turns negative. This is an artifact, because the sum of the estimated autocorrelation coefficients needs to add up to $-1/2$. Some of these spurious, negative correlation estimates are so big that they even exceed the confidence bounds – an observation that has to be well kept in mind if one analyzes and tries to interpret the correlogram.

Simulation Study

Last but not least, we will run a small simulation study that visualizes bias and variance in the sample autocorrelation coefficients. We will again base this on the simple $AR(1)$ process with coefficient $\alpha_1 = 0.7$. For further discussion of the process' properties, we refer to section 5. There, it will turn out that the k^{th} autocorrelation coefficient of such a process takes the value $(0.7)^k$, as visualized on the previous page.

For understanding the variability in $\hat{\rho}(1)$, $\hat{\rho}(2)$, $\hat{\rho}(5)$ and $\hat{\rho}(10)$, we simulate from the aforementioned $AR(1)$ process. We generate series of length $n = 20$, $n = 50$, $n = 100$ and $n = 200$. We then obtain the correlogram, record the estimated autocorrelation coefficients and repeat this process 1000 times. This serves as a basis for displaying the variability in $\hat{\rho}(1)$, $\hat{\rho}(2)$, $\hat{\rho}(5)$ and $\hat{\rho}(10)$ with boxplots. They can be found below.



We observe that for “short” series with less than 100 observations, estimating the ACF is difficult: the $\hat{\rho}(k)$ are strongly biased, and there is huge variability. Only for longer series, the consistency of the estimator “kicks in”, and yields estimates which are reasonably precise. For lag $k = 10$, on the other hand, we observe less bias, but the variability in the estimate remains large, even for “long” series.

We conclude this situation by summarizing: by now, we have provided quite a bit of evidence that the correlogram can be tricky to interpret at best, sometimes even misleading, or plain wrong. However, it is the best means we have for understanding the dependency in a time series. And we will base many if not most of our decisions in the modeling process on the correlogram. However, please be aware of the bias and the estimation variability there is.

4.5 Partial Autocorrelation

For the above $AR(1)$ process, with its strong positive correlation at lag 1, it is somehow “evident” that the autocorrelation for lags 2 and higher will be positive as well – just by propagation: if A is highly correlated to B, and B is highly correlated to C, then A is usually highly correlated to C as well. It would now be very instructive to understand the direct relation between A and C, i.e. exploring what dependency there is in excess to the one associated to B. In a time series context, this is exactly what the partial autocorrelations do. The mathematical definition is the one of a conditional correlation:

$$\pi(k) = \text{Cor}(X_{t+k}, X_t | X_{t+1} = x_{t+1}, \dots, X_{t+k-1} = x_{t+k-1})$$

In other words, we can also say that the partial autocorrelation is the association between X_t and X_{t+k} with the linear dependence of X_{t+1} through X_{t+k-1} removed. Another instructive analogy can be drawn to linear regression. The autocorrelation coefficient $\rho(k)$ measures the simple dependence between X_t and X_{t+k} , whereas the partial autocorrelation $\pi(k)$ measures the contribution to the multiple dependence, with the involvement of all intermediate instances $X_{t+1}, \dots, X_{t+k-1}$ as explanatory variables. There is a (theoretical) relation between the partial autocorrelations $\pi(k)$ and the plain autocorrelations $\rho(1), \dots, \rho(k)$, i.e. they can be derived from each other, e.g.:

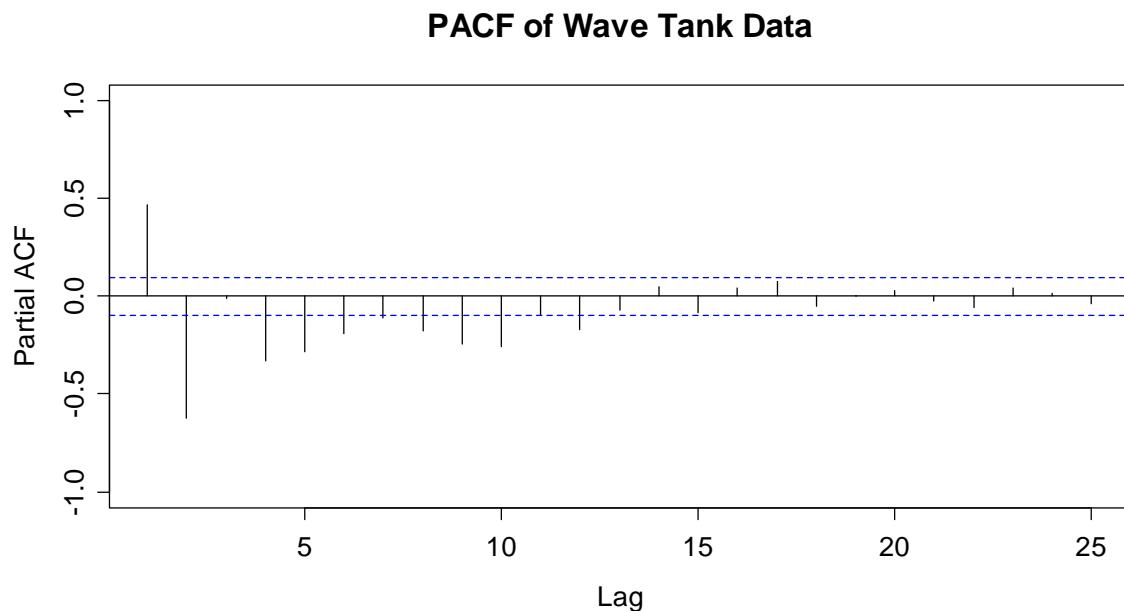
$$\pi(1) = \rho(1) \text{ and } \pi(2) = (\rho(2) - \rho(1)^2) / (1 - \rho(1)^2)$$

The formula for higher lags k exists, but get complicated rather quickly, so we do without displaying them. However, another absolutely central property of the partial autocorrelations $\pi(p)$ is that the k^{th} coefficient of the $AR(p)$ model, denoted as α_p , is equal to $\pi(p)$. While there is an in depth discussion of $AR(p)$ models in section 5, we here briefly sketch the idea, because it makes the above property seem rather logical. An autoregressive model of order p , i.e. an $AR(p)$ is:

$$X_t = \alpha_1 X_{t-1} + \dots + \alpha_k X_{t-p} + E_t,$$

where E_t is a sequence of iid random variables. Making the above statement concrete, this means that in an AR(3) process, we have $\pi(3)=\alpha_3$, but generally $\pi(2)\neq\alpha_2$ and $\pi(1)\neq\alpha_1$. Moreover, we have $\pi(k)=0$ for all $k>p$. These properties are used in R for estimating partial autocorrelation coefficients. Estimates $\hat{\pi}(p)$ are generated by fitting autoregressive models of successively higher orders. The job is done with function `pacf()`: input/output are equal/similar to ACF estimation. In particular, the confidence bounds are also presented for the PACF. We conclude this section by showing the result for the wave tank data.

```
> pacf(wave, ylim=c(-1,1), main="PACF of Wave Tank Data")
```



We observe that $\hat{\pi}(1)\approx 0.5$ and $\hat{\pi}(2)\approx -0.6$. Some further PACF coefficients up to lag 10 seem significantly different from zero, but are smaller. From what we see here, we could try to describe the wave tank data with an AR(2) model. The next section will explain why.

5 Stationary Time Series Models

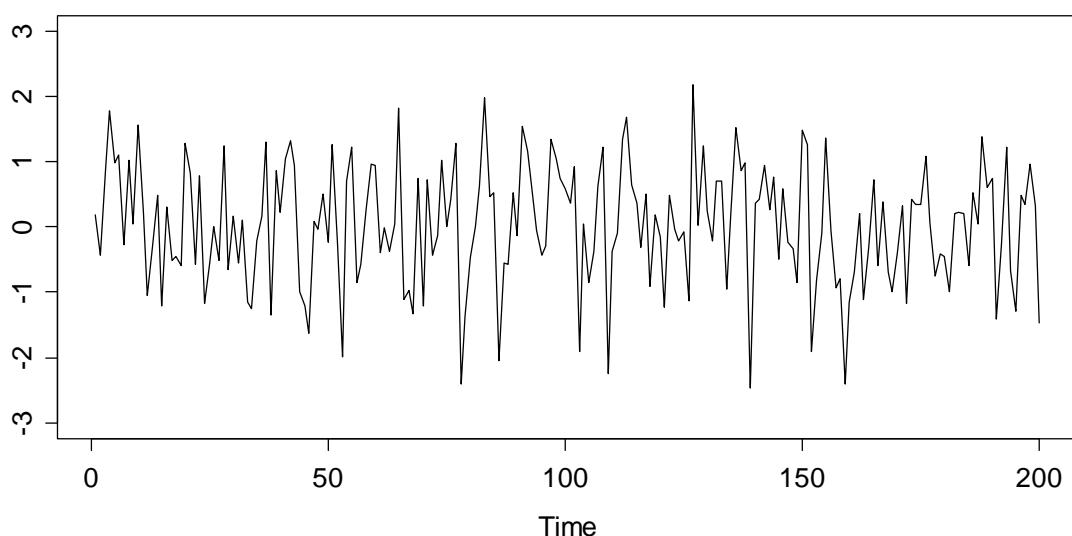
Rather than simply describing observed time series data, we now aim for fitting models. This will prove useful for a deeper understanding of the data, but is especially beneficial when forecasting is the main goal. We here focus on parametric models for stationary time series, namely the broad class of *autoregressive moving average (ARMA) processes* – these have shown great importance in modeling real-world data.

5.1 White Noise

As the most basic stochastic process, we introduce discrete White Noise. A time series (W_1, W_2, \dots, W_n) is called *White Noise* if the random variables W_1, W_2, \dots are independent and identically distributed with mean zero. This also implies that all random variables W_t have identical variance, and there are no autocorrelations and partial autocorrelations either: $\rho(k) = 0$ and $\pi(k) = 0$ for all lags k . If in addition, the variables also follow a Gaussian distribution, i.e. $W_t \sim N(0, \sigma_w^2)$, the series is called *Gaussian White Noise*.

Before we show a realization of a White Noise process, we state that the term “White Noise” was coined in an article on heat radiation published in Nature in April 1922. There, it was used to refer to series time series that contained all frequencies in equal proportions, analogous to white light. It is possible to show that iid sequences of random variables do contain all frequencies in equal proportions, and hence, here we are.

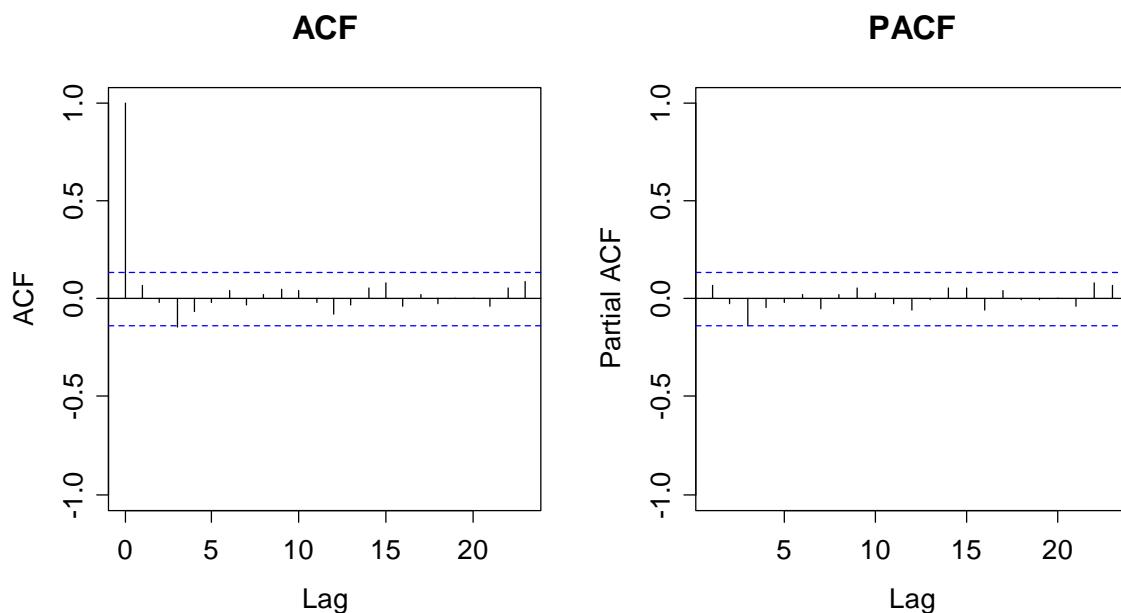
Gaussian White Noise



In R, it is easy to generate Gaussian White Noise, we just type:

```
> ts(rnorm(200, mean=0, sd=1))
```

Well, by giving more thought on how computers work, i.e. by relying on deterministic algorithms, it may seem implausible that they can really generate independent data. We do not embark into these discussions here, but treat the result of `rnorm()` as being “good enough” for a realization of a White Noise process. Here, we show ACF and PACF of the above series. As expected, there are no (strongly) significant estimates.



White Noise series are important, because they usually arise as residual series when fitting time series models. The correlogram generally provides enough evidence for attributing a series as White Noise, provided the series is of reasonable length – our studies in section 4.4 suggests that 100 or 200 is such a value. Please also note that while there is not much structure in Gaussian White Noise, it still has a parameter. It is the variance σ_w^2 .

5.2 Estimating the Conditional Mean

Before we present some time series models, it is important to build some understanding of what we are actually doing. All the $AR(p)$, $MA(q)$ and $ARMA(p,q)$ models that will be presented below are based on the assumption that the time series can be written as:

$$X_t = \mu_t + E_t.$$

Hereby, μ_t is the conditional mean of the series, i.e. $\mu_t = E[X_t | X_{t-1}, X_{t-2}, \dots]$ and E_t is a disturbance term. For all models in section 5, the disturbance term is assumed to be a White Noise innovation.

It is very important to notice that while stationary series feature a constant marginal expectation μ , the conditional mean μ_t is can be and often is non-constant and time-dependent. Or in other words, there is some short-term memory in the series. The $ARMA(p,q)$ processes that will be discussed here in this section are built on the following notion:

$$\mu_t = f(X_{t-1}, X_{t-2}, \dots, X_{t-p}, E_{t-1}, E_{t-2}, \dots, E_{t-q}).$$

In words, the conditional mean is a function of past instances of the series as well as past innovations. We will see that usually, a selection of the involved terms is made, and that the function $f(\cdot)$ is linear.

5.3 Autoregressive Models

5.3.1 Definition and Properties

The most natural formulation of a time series model is a linear regression approach on the past instances, i.e. a regression on the series itself. This coined the term *autoregressive*. In practice, such models prove to be very important; they are the most popular way of describing time series.

Model and Terms

An *autoregressive model of order p*, abbreviated as $AR(p)$, is based on a linear combination of past observations according to the following equation:

$$X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + \dots + \alpha_p X_{t-p} + E_t.$$

Hereby, the disturbance term E_t comes from a White Noise process, i.e. is iid. Moreover, we require that it is an *innovation*, i.e. that it is stochastically independent of X_{t-1}, X_{t-2}, \dots . The term innovation is illustrative, because (under suitable conditions), it has the power to drive the series into a new direction, meaning that it is strong enough so that it can overplay the dependence of the series from its own past. An alternative notation for $AR(p)$ models is possible with the backshift operator:

$$(1 - \alpha_1 B - \alpha_2 B^2 - \dots - \alpha_p B^p) X_t = E_t, \text{ or short } \Phi(B) X_t = E_t$$

Hereby, $\Phi(B)$ is called the *characteristic polynomial*. It determines all the relevant properties of the process. The most important questions that we will deal with in this chapter are of course the choice of the order p and the estimation of the coefficients $\alpha_1, \dots, \alpha_p$. But first, a very important point:

- $AR(p)$ models must only be fitted to stationary time series. Any potential trends and/or seasonal effects need to be removed first. We will also make sure that the $AR(p)$ processes are stationary.

When is an $AR(p)$ stationary? Not always, but under some mild conditions. First of all, we study the unconditional expectation of an $AR(p)$ process X_t , which we assume to be stationary, hence $E[X_t] = \mu$ for all t . When we take expectations on both sides of the model equation, we have:

$$\mu = E[X_t] = E[\alpha_1 X_{t-1} + \dots + \alpha_p X_{t-p} + E_t] = (\alpha_1 + \dots + \alpha_p) \cdot \mu + 0, \text{ hence } \mu = 0.$$

Thus, any stationary $AR(p)$ process has a global mean of zero. But please be aware of the fact that the conditional mean is time dependent and generally different from zero.

$$\mu_t = E[X_t | X_{t-1}, \dots, X_{t-p}] = \alpha_1 x_{t-1} + \dots + \alpha_p x_{t-p}$$

The question remains if $AR(p)$ processes are practically useful, because most of the real-word time series have a global mean μ that is different from zero. However, that generally poses little difficulties if we add an additional parameter m to the model definition:

$$Y_t = m + X_t$$

In that case, Y_t is a *shifted AR(p) process*, i.e. it has all dependency properties from an $AR(p)$, but its mean is different from zero. In fact, all R methodology that exists for fitting $AR(p)$'s assumes the process Y_t and thus estimates a global mean m unless this is explicitly excluded. In practice, if one colloquially speaks of an $AR(p)$, mostly one thinks of Y_t rather than X_t .

However, for the stationarity of an $AR(p)$, some further conditions on the model coefficients $\alpha_1, \dots, \alpha_p$ are required. The general derivation is quite complicated and will be omitted here. But for illustrative purpose, we assume a stationary $AR(1)$ which has $Var(X_t) = \sigma_X^2$ for all t . If we determine the centralized second moment on both sides of the model equation, we obtain:

$$\sigma_X^2 = Var(X_t) = Var(\alpha_1 X_{t-1} + E_t) = \alpha_1^2 \sigma_X^2 + \sigma_E^2, \text{ hence } \sigma_X^2 = \frac{\sigma_E^2}{1 - \alpha_1^2}.$$

From this we derive that an $AR(1)$ can only be stationary if $|\alpha_1| < 1$. That limitation means that the dependence from the series' past must not be too strong, so that the memory fades out. If $|\alpha_1| > 1$, the process diverges. The general condition for $AR(p)$ models is (as mentioned above) more difficult to derive. We require that:

The (potentially complex) roots of the characteristic polynomial $\Phi(B)$ must all exceed 1 in absolute value for an $AR(p)$ process to be stationary.

In R, there is function `polyroot()` for finding a polynomials roots. If we want to verify whether an $AR(3)$ with $\alpha_1 = 0.4$, $\alpha_2 = -0.2$, $\alpha_3 = 0.3$ is stationary, we type:

```
> abs(polyroot(c(1, 0.4, -0.2, 0.3)))
[1] 1.776075 1.056710 1.776075
```

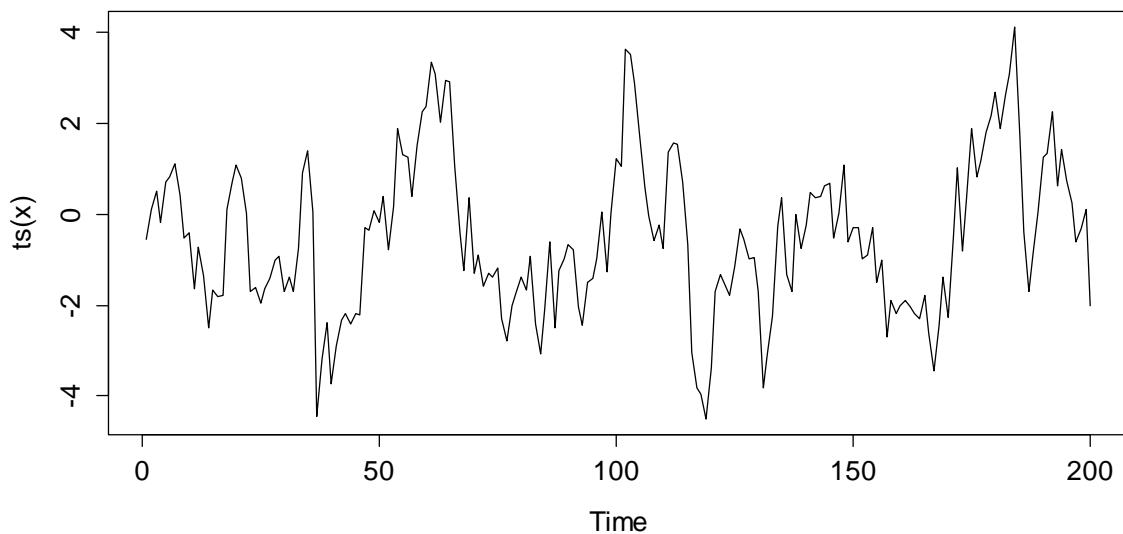
Thus, the $AR(3)$ we specified is stationary. We will proceed by studying the dependency in $AR(p)$ processes. For illustration, we first simulate from an $AR(1)$ with $\alpha_1 = 0.8$. The model equation is:

$$X_t = 0.8 \cdot X_{t-1} + E_t$$

So far, we had only required that E_t is a White Noise innovation, but not a distribution. We use the Gaussian in this example and set $x_1 = E_1$ as the starting value.

```
> set.seed(24)
> E      <- rnorm(200, 0, 1)
> x      <- numeric()
> x[1]   <- E[1]
> for(i in 2:200) x[i] <- 0.8*x[i-1] + E[i]
> plot(ts(x), main= "AR(1) with...")
```

AR(1) with $\alpha_1=0.8$



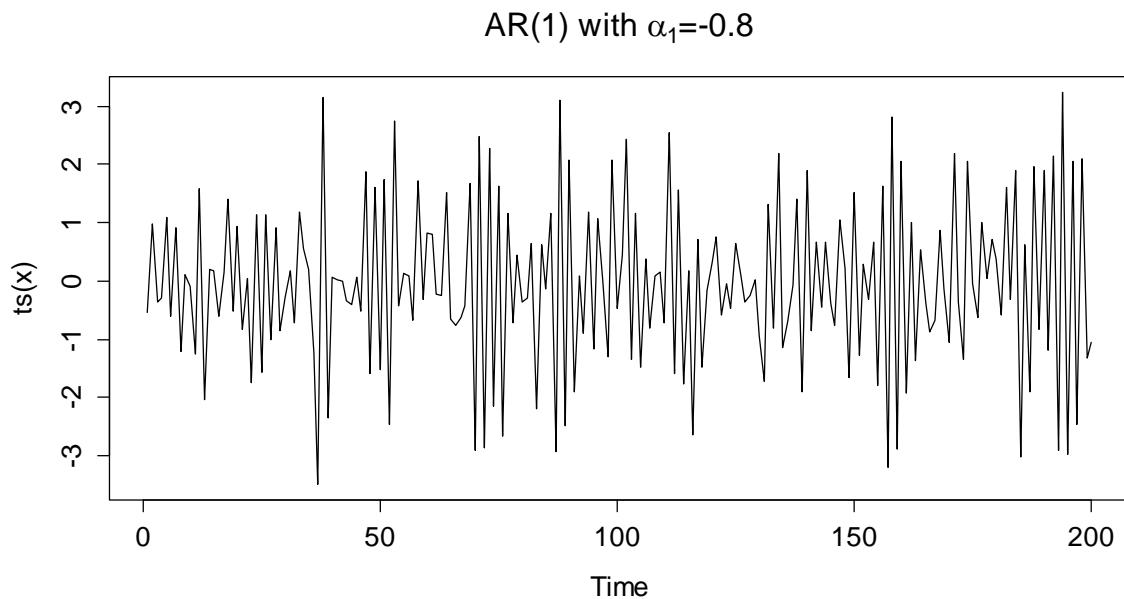
We observe some cycles with exclusively positive and others with only negative values. That is not surprising: if the series takes a large value, then the next one is determined as 0.8 times that large value plus the innovation. Thus, it is more likely that the following value has the same sign as its predecessor. On the other hand, the innovation is powerful enough so that jumping to the other side of the global mean zero is always a possibility. Given that behavior, it is evident that the autocorrelation at lag 1 is positive. We can compute it explicitly from the model equation:

$$\text{Cor}(X_t, X_{t-1}) = \text{Cor}(\alpha_1 X_{t-1} + E_t, X_{t-1}) = \alpha_1$$

Thus we have $\rho(1) = 0.8$ here, or in general $\rho(1) = \alpha_1$. The correlation for higher lags can be determined similarly by repeated plug-in of the model equation. It is:

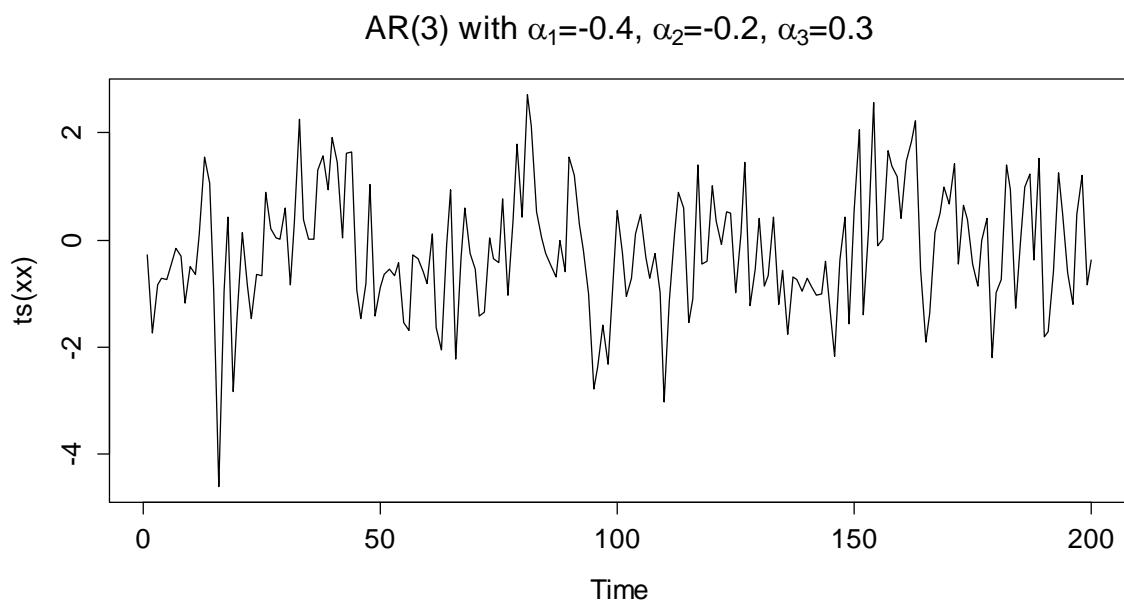
$$\rho(k) = \alpha_1^k.$$

Thus, for stationary $AR(1)$ series, we have an exponential decay of the autocorrelation coefficients. Of course, it is also allowed to have a negative value for α_1 , as long as $|\alpha_1| < 1$. A realization of length 200 with $\alpha_1 = -0.8$ is as follows:



The series shows an alternating behavior: the next value is more likely to lie on the opposite side of the global mean zero, but there are exceptions when the innovation takes a large value. The autocorrelation still follows $\rho(k) = \alpha_1^k$. It is also alternating between positive and negative values with an envelope that is exponentially decaying.

We will now focus on appearance and dependency of an $AR(3)$ (with the coefficients from above). While we could still program the simulation code by ourselves, it is more convenient to use function `arima.sim()`.



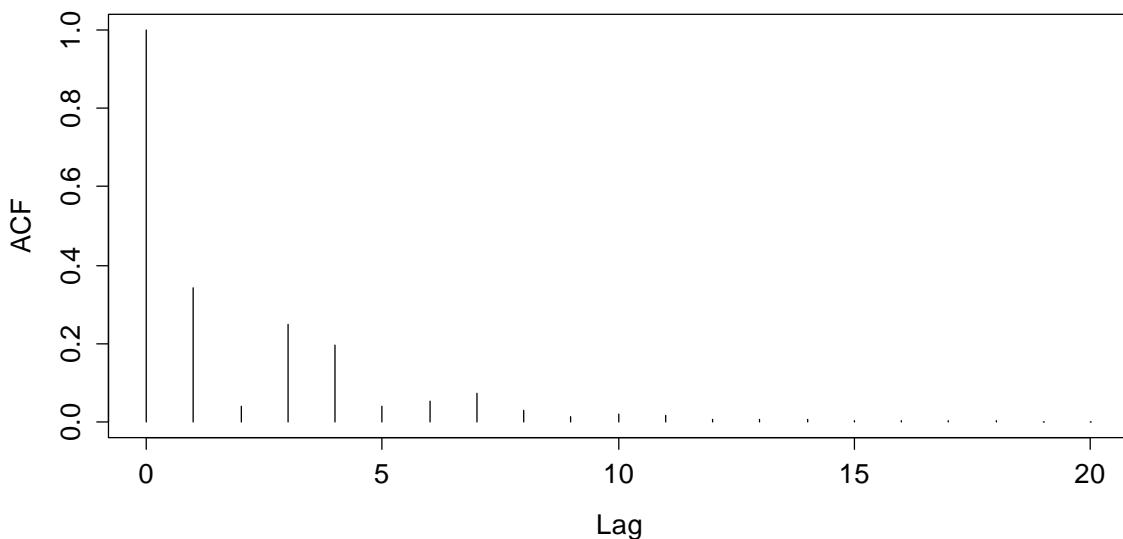
What is now the (theoretical) correlation in this $AR(3)$? We apply the standard trick of plugging-in the model equation. This yields:

$$\begin{aligned}\rho(k) &= Cor(X_{t+k}, X_t) \\ &= Cor(\alpha_1 X_{t+k-1} + \dots + \alpha_p X_{t+k-p}, X_t) \\ &= \alpha_1 \rho(k-1) + \dots + \alpha_p \rho(k-p)\end{aligned}$$

with $\rho(0)=1$ and $\rho(-k)=\rho(k)$. For $k=1,\dots,p$ this results in a $p \times p$ linear equation system called the *Yule-Walker equations*. It can be solved to obtain the autocorrelation coefficients which can finally be propagated for $k=p+1, p+2, \dots$. In R, there is function `armaACF()` that allows to determine the autocorrelation from autoregressive model coefficients.

```
> autocorr <- ARMAacf(ar=c(0.4, -0.2, 0.3), lag.max=20)
> plot(0:20, autocorr, type="h", xlab="Lag")
```

Theoretical Autocorrelation for an AR(3)

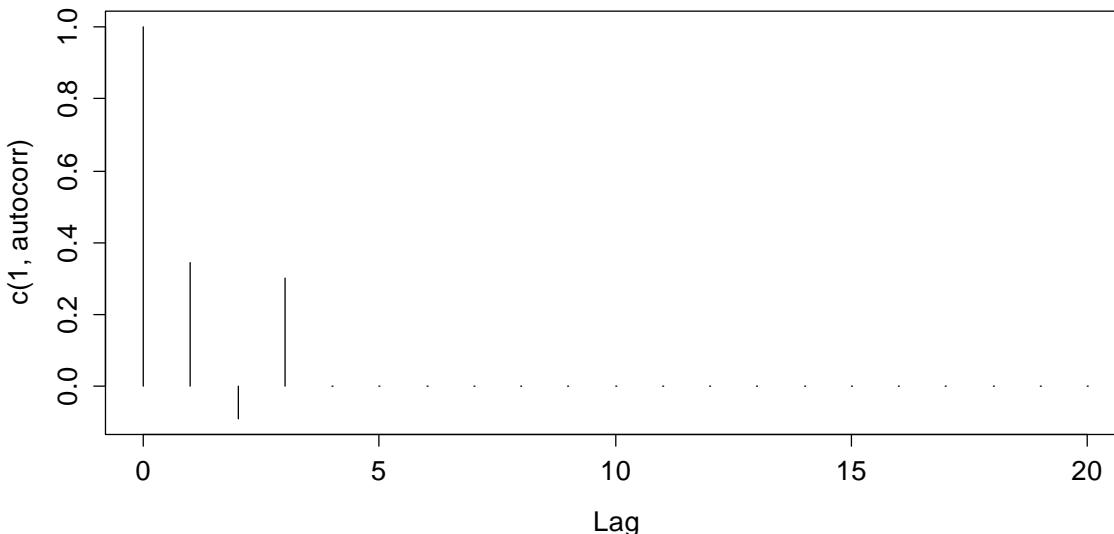


We observe that the theoretical correlogram shows a more complex structure than what could be achieved with an $AR(1)$. Nevertheless, one can still find an exponentially decaying envelope for the magnitude of the autocorrelations. That is a property which is common to all $AR(p)$ models.

From the above, we can conclude that the autocorrelations are generally non-zero for all lags, even though in the underlying model, X_t only depends on the p previous values X_{t-1}, \dots, X_{t-p} . In section 4.5 we learned that the partial autocorrelation at lag k illustrates the dependence between X_t and X_{t+k} when the linear dependence on the intermittent terms was already taken into account. It is evident by definition that for any $AR(p)$ process, we have $\pi(k)=0$ for all $k > p$. This can and will serve as a useful indicator for deciding on the model order p if we are trying to identify the suitable model order when fitting real world data. In this section, we focus on the PACF for the above $AR(3)$.

```
> autocorr <- ARMAacf(ar=..., pacf=TRUE, lag.max=20)
> plot(0:20, autocorr, type="h", xlab="Lag")
```

Theoretical Partial Autocorrelation for an AR(3)



As claimed previously, we indeed observe $\rho(1) = \pi(1) = 0.343$ and $\pi(3) = \alpha_3 = 0.3$. All partial autocorrelations from $\pi(4)$ on are exactly zero.

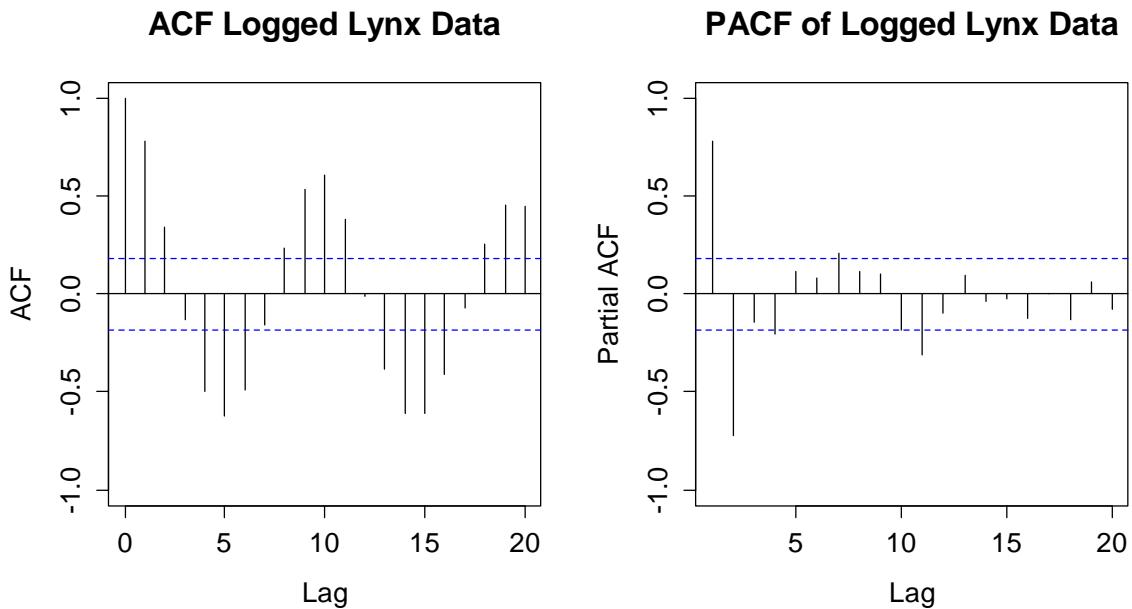
5.3.2 Fitting

Fitting an $AR(p)$ model to data involves three main steps. First, the model and its order need to be identified. Second, the model parameters need to be estimated and third, the quality of the fitted model needs to be verified by residual analysis.

Model Identification

The model identification step first requires verifying that the data show properties which make it plausible that they were generated from an $AR(p)$ process. In particular, the time series we are aiming to model needs to be stationary, show an ACF with approximately exponentially decaying envelope and a PACF with a recognizable cut-off at some lag p smaller than about 5–10. If any of these three properties is strongly violated, it is unlikely that an $AR(p)$ will yield a satisfactory fit, and there might be models which are better suited for the problem at hand.

The choice of the model order p then relies on the analysis of the sample PACF. Following the paradigm of parameter parsimony, we would first try the simplest model that seems plausible. This means choosing the smallest p at which we suspect a cut-off, i.e. the smallest after which none, or only few and weakly significant partial autocorrelations follow. We illustrate the concept with the logged Lynx data that were already discussed in section 1.2.2. We need to generate both ACF and PACF, which can be found on the next page.



There is no reason to doubt the stationarity of the Lynx series. Moreover, the ACF shows a cyclic behavior that has an exponentially decaying envelope. Now does the PACF show a cut-off? That is a bit less clear, and several orders p ($= 2, 4, 7, 11$) come into question. However in summary, we conjecture that there are no strong objections against fitting an $AR(p)$. The choice of the order is debatable, but the parsimony paradigm tells us that we should try with the smallest candidate first, and that is $p = 2$.

Parameter Estimation

Observed time series are rarely centered and thus, it is usually inappropriate to fit a pure $AR(p)$ process. In fact, all R routines for fitting autoregressive models by default assume the shifted process $Y_t = m + X_t$. Hence, we have a regression-type equation with observations:

$$(Y_t - m) = \alpha_1(Y_{t-1} - m) + \dots + \alpha_p(Y_{t-p} - m) + E_t \text{ for } t = p+1, \dots, n.$$

The goal here is to estimate the parameters $m, \alpha_1, \dots, \alpha_p$ such that the data are *fitted well*. There are several concepts that define *well fitting*. These include ordinary least squares estimation (OLS), Burg's algorithm (Burg), the Yule-Walker approach (YW) and maximum likelihood estimation (MLE). Already at this point we note that while the four methods have fundamental individuality, they are asymptotically equivalent (under some mild assumptions) and yield results that mostly only differ slightly in practice. Still, it is worthwhile to study all the concepts.

OLS

The OLS approach is based on the notion with the centering; the above equation defines a multiple linear regression problem without intercept. The goodness-of-fit criterion is $(x_t - \hat{x}_t)^2$ resp. $(y_t - \hat{y}_t)^2$, the two quantities are equal. The first step with this approach is to center the data, which is based on subtracting the global mean:

Estimate $\hat{m} = \bar{y} = \sum_{t=1}^n y_t$ and then compute $x_t = y_t - \hat{m}$ for all $t = 1, \dots, n$.

On the x_t , an OLS (auto)regression without intercept is performed. Note that this regression is (technically) conditional on the first p observations x_1, \dots, x_p , which are only used as predictors, but not as response terms. In other words, the goodness-of-fit of the model is only evaluated for the last $n-p$ observations. The following code chunk implements the procedure for the logged lynx data:

```
> llc      <- log(lynx)-mean(log(lynx))
> resp     <- llc[3:114]
> pred1    <- llc[2:113]
> pred2    <- llc[1:112]
> fit.ols  <- lm(resp ~ -1 + pred1 + pred2)
> summary(fit.ols)

Coefficients:
            Estimate Std. Error t value Pr(>|t| )
pred1    1.38435   0.06359  21.77   <2e-16 ***
pred2   -0.74793   0.06364 -11.75   <2e-16 ***
---
Residual standard error: 0.528 on 110 degrees of freedom
Multiple R-squared: 0.8341, Adjusted R-squared: 0.8311
F-statistic: 276.5 on 2 and 110 DF, p-value: < 2.2e-16
```

We can extract $\hat{m} = 6.686$, $\hat{\alpha}_1 = 1.384$, $\hat{\alpha}_2 = -0.748$ and $\hat{\sigma}_E = 0.528$. But while this is an instructive way of estimating $AR(p)$ models, it is a bit cumbersome and time consuming. Not surprisingly, there are procedures that are dedicated to fitting such models in R. We here display the use of function `ar.ols()`. To replicate the hand-produced result, we type:

```
> f.ar.ols <- ar.ols(log(lynx), aic=F, intercept=F, order=2)
> f.ar.ols

Coefficients:
      1          2
1.3844 -0.7479

Order selected 2  sigma^2 estimated as  0.2738
```

Note that for producing the result, we need to avoid AIC-based model fitting with `aic=FALSE`. The shift m is automatically estimated, and thus we need to exclude an intercept term in the regression model using `intercept=FALSE`. We observe that the estimated AR -coefficients $\hat{\alpha}_1, \hat{\alpha}_2$ take exactly the same values as with the hand-woven procedure above. The estimated shift \hat{m} can be extracted via

```
> fit.ar.ols$x.mean
[1] 6.685933
```

and corresponds to the global mean of the series. Finally, the estimate for the innovation variance requires some prudence. The `lm()` summary output yields an

estimate of σ_E that was computed as $RSS / (n - p)$, whereas the value in the `ar.ols()` output is an estimate of σ_E^2 that was computed as RSS / n . The former is intended to be an unbiased estimate (though it should use the denominator $n - p - 1$ due to the estimation of the shift m), and the latter is the MLE-estimator for the innovation variance. In practice, the numerical difference between the two is neglectable for any series that has reasonable length for fitting an *AR* model.

```
> sum(na.omit(fit.ar.ols$resid)^2)/112
[1] 0.2737594
```

Burg's Algorithm

While the OLS approach works, its downside is the asymmetry: the first p terms are never evaluated as responses. That is cured by Burg's Algorithm, an alternative approach for estimating *AR*(p) models. It is based on the notion that any *AR*(p) process is also an *AR*(p) if the time is run in reverse order. Under this property, minimizing the forward and backward 1-step squared prediction errors makes sense:

$$\sum_{t=p+1}^n \left\{ \left(X_t - \sum_{k=1}^p \alpha_k X_{t-k} \right)^2 + \left(X_{t-p} - \sum_{k=1}^p \alpha_k X_{t-p+k} \right)^2 \right\}$$

In contrast to OLS, there is no explicit solution and numerical optimization is required. This is done with a recursive method called the Durbin-Levinson algorithm. We do not explain its details here, but refer to the R implementation `ar.burg()`.

```
> f.ar.burg <- ar.burg(log(lynx), aic=FALSE, order.max=2)
> f.ar.burg

Call:
ar.burg.default(x = log(lynx), aic = FALSE, order.max = 2)

Coefficients:
      1          2
1.3831   -0.7461

Order selected 2  sigma^2 estimated as  0.2707

> f.ar.burg$x.mean
[1] 6.685933
> sum(na.omit(f.ar.burg$resid)^2)/112
[1] 0.2737614
```

There are a few interesting points which require commenting. First and foremost, Burg's algorithm also uses the arithmetic mean to estimate the global mean \hat{m} . The fitting procedure is then done on the centered observations x_t . On a side remark, note that assuming centered observations is possible. If argument `demean=FALSE` is set, the global mean is assumed to be zero and not estimated.

The two coefficients $\hat{\alpha}_1, \hat{\alpha}_2$ take some slightly different values than with OLS estimation. While often, the difference between the two methods is practically neglectable, it is nowadays generally accepted that the Burg solution is better for finite samples. Asymptotically, the two methods are equivalent. Finally, we observe that the `ar.burg()` output specifies $\hat{\sigma}_E^2 = 0.2707$. This is different from the MLE estimate of 0.27376 on the residuals. The explanation is that for Burg's Algorithm, the innovation variance is estimated from the Durbin-Levinson updates; see the R help file for further reference.

Yule-Walker Equations

A third option for estimating $AR(p)$ models is to plugging-in the sample ACF into the Yule-Walker equations. In section 5.3.1 we had learned that there is a $p \times p$ linear equation system $|\rho(k) = \alpha_1\rho(k-1) + \dots + \alpha_p\rho(k-p)|$ for $k = 1, \dots, p$. Hence we can and will explicitly determine $\hat{\rho}(0), \dots, \hat{\rho}(k)$ and then solve the linear equation system for the coefficients $\alpha_1, \dots, \alpha_p$. The procedure is implemented in R function `ar.yw()`.

```
> f.ar.yw <- ar.yw(log(lynx), aic=FALSE, order.max=2)
> f.ar.yw
```

```
Call: ar.yw.default(x=log(lynx), aic=FALSE, order.max=2)
```

Coefficients:

1	2
1.3504	-0.7200

```
Order selected 2 sigma^2 estimated as 0.3109
```

Again, the two coefficients $\hat{\alpha}_1, \hat{\alpha}_2$ take some slightly different values than compared to the two methods before. Mostly this difference is practically neglectable and Yule-Walker is asymptotically equivalent to OLS and Burg. Nevertheless, for finite samples, the estimates from the Yule-Walker method are often worse in the sense that their (Gaussian) likelihood is lower. Thus, we recommend to prefer Burg's algorithm. We conclude this section by noting that the Yule-Walker method also involves estimating the global mean m with the arithmetic mean as the first step. The innovation variance is estimated from the fitted coefficients and the autocovariance of the series and thus again takes a different value than before.

Maximum-Likelihood Estimation (MLE)

The MLE is based on determining the model coefficients such that the likelihood given the data is maximized, i.e. the density function takes its maximal value under the present observations. This requires assuming a distribution for the $AR(p)$ process, which comes quite naturally if one assumes that for the innovations, we have $E_t \sim N(0, \sigma_E^2)$, i.e. they are *iid* Gaussian random variables. With some theory (which we omit), one can then show that an $AR(p)$ process X_1, \dots, X_n is a random vector with a multivariate Gaussian distribution.

MLE then provides a simultaneous estimation of the shift m , the innovation variance σ_E^2 and the model coefficients $\alpha_1, \dots, \alpha_p$. The criterion that is optimized can, in a simplified version, be written as:

$$L(\alpha, m, \sigma_E^2) \propto \exp\left(\sum_{t=1}^n (x_t - \hat{x}_t)^2\right)$$

The details are quite complex and several constants are part of the equation, too. But we here note that the MLE derived from the Gaussian distribution is based on minimizing the sum of squared errors and thus equivalent to the OLS approach. Due to the simultaneous estimation of model parameters and innovation variance, a recursive algorithm is required. There is an implementation in R:

```
> f.ar.mle

Call: arima(x = log(lynx), order = c(2, 0, 0))

Coefficients:
            ar1      ar2  intercept
            1.3776 -0.7399    6.6863
s.e.    0.0614   0.0612    0.1349

sigma^2 = 0.2708:  log likelihood = -88.58,  aic = 185.15
```

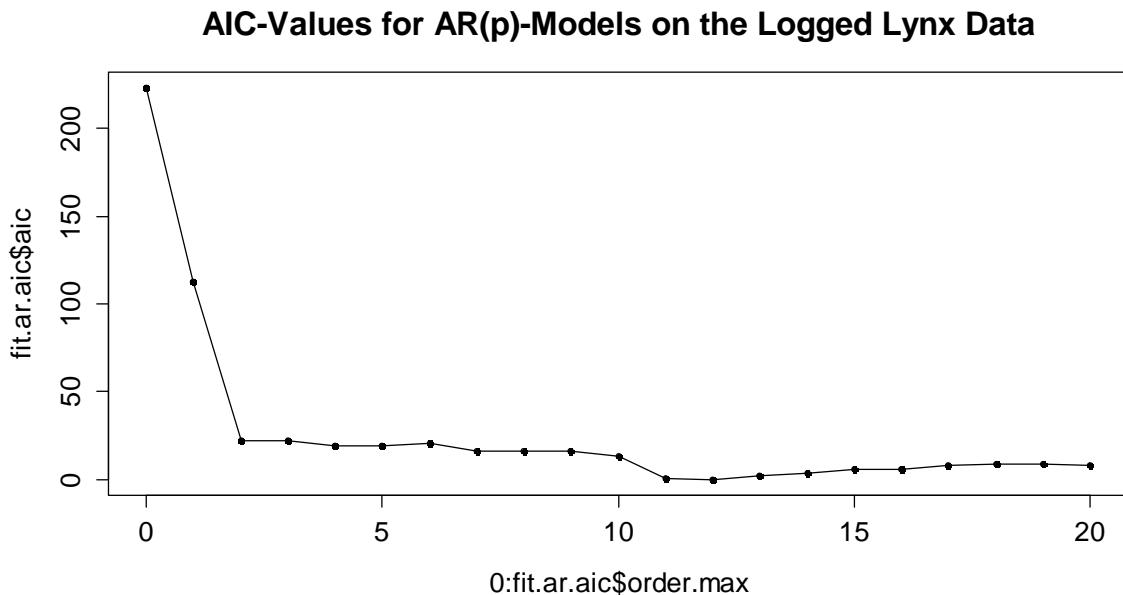
We observe estimates which are again slightly different from the ones computed previously. Again, those differences are mostly neglectable for practical data analysis. What is known from theory is that the MLE is (under mild assumptions) asymptotically normal with minimum variance among all asymptotically normal estimators. Note that the MLE based on Gaussian distribution still works reasonably well if that assumption is not met, as long as we do not have strongly skewed data (apply a transformation in that case) or extreme outliers.

Practical Aspects

We presented four different methods for fitting $AR(p)$ models. How to make a choice in practice? We explained that all methods are asymptotically equivalent and even on finite samples; the differences among them are little. Also, all methods are non-robust with respect to outliers and perform best on data which are approximately Gaussian. There is one practical aspect linked to the fitting routines that are available in R, though. Function `arima()` yields standard errors for m and $\alpha_1, \dots, \alpha_p$. Approximate 95% confidence intervals can be obtained by taking the point estimate +/- twice the standard error. Hence, statements about the significance of the estimates can be made.

On the other hand, `ar.ols()`, `ar.yw()` und `ar.burg()` do not provide standard errors, but allow for convenient determination of the model order p with the AIC statistic. While we still recommend investigating on the suitable order by analyzing ACF and PACF, the parsimony paradigm and inspecting residual plots, using AIC as a second opinion is still recommended. It works as follows:

```
> fit.aic <- ar.burg(log(lynx))
> plot(0:fit.aic$order.max, fit.aic$aic)
```



We observe that already $p = 2$ yields a good AIC value. Then there is little further improvement until $p = 11$, and a just slightly lower value is found at $p = 12$. Hence, we will evaluate $p = 2$ and $p = 11$ as two competing models with some further tools in the next section.

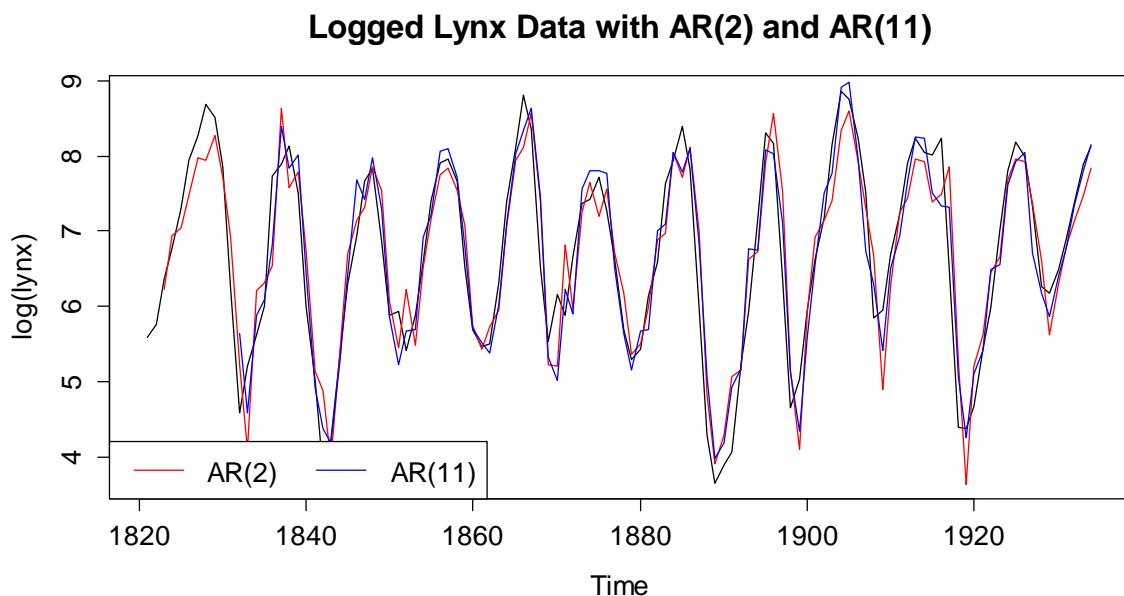
5.3.3 Residual Analysis

When comparing different models, a simple approach is to plot the original series along with the fitted model values. However, one has to keep in mind that this is an insample analysis, i.e. the bigger model has an advantage which does not necessarily persist once analyzes out-of-sample data. Please note that the residuals are estimates of the innovations E_t . Thus, a good model yields residuals that resemble a White Noise process. We require mean zero, constant variance and no autocorrelation. If these properties are not met, the model is not adequate.

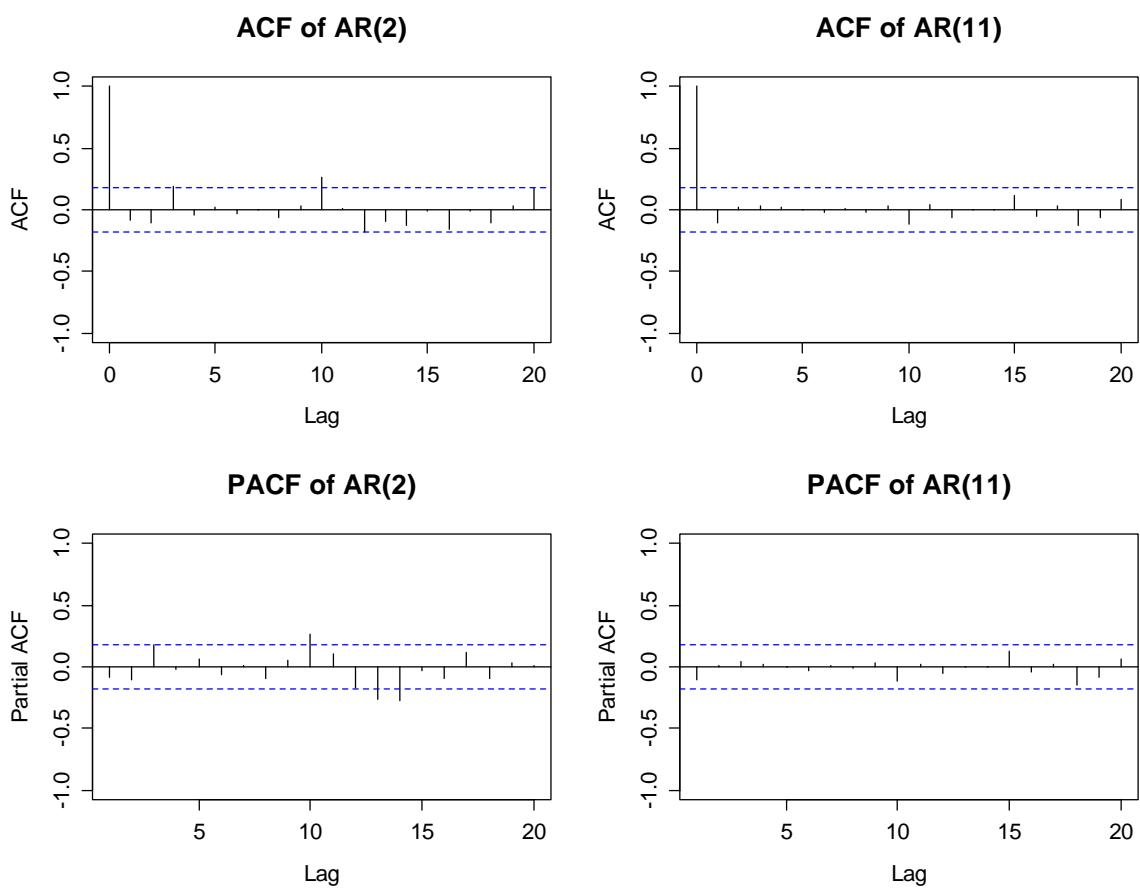
```
> fit.ar02 <- ar.burg(log(lynx), aic=FALSE, order.max=2)
> fit.ar11 <- ar.burg(log(lynx), aic=FALSE, order.max=11)
> plot(log(lynx), main="Logged Lynx Data with ...")
> lines(log(lynx)-fit.ar02$resid, col="red")
> lines(log(lynx)-fit.ar11$resid, col="blue")
```

The output is displayed on the next page. While some differences are visible, it is not easy to judge from the fitted values which of the two models is preferable. A better focus on the quality of the fit is obtained when the residuals and their dependence are inspected with time series plots as well as ACF/PACF correlograms. The graphical output is again displayed on the next page. We observe that the $AR(2)$ residuals are not iid. Hence they do not form a White

Noise process and thus, we conclude that the $AR(11)$ model yields a better description of the logged lynx data.

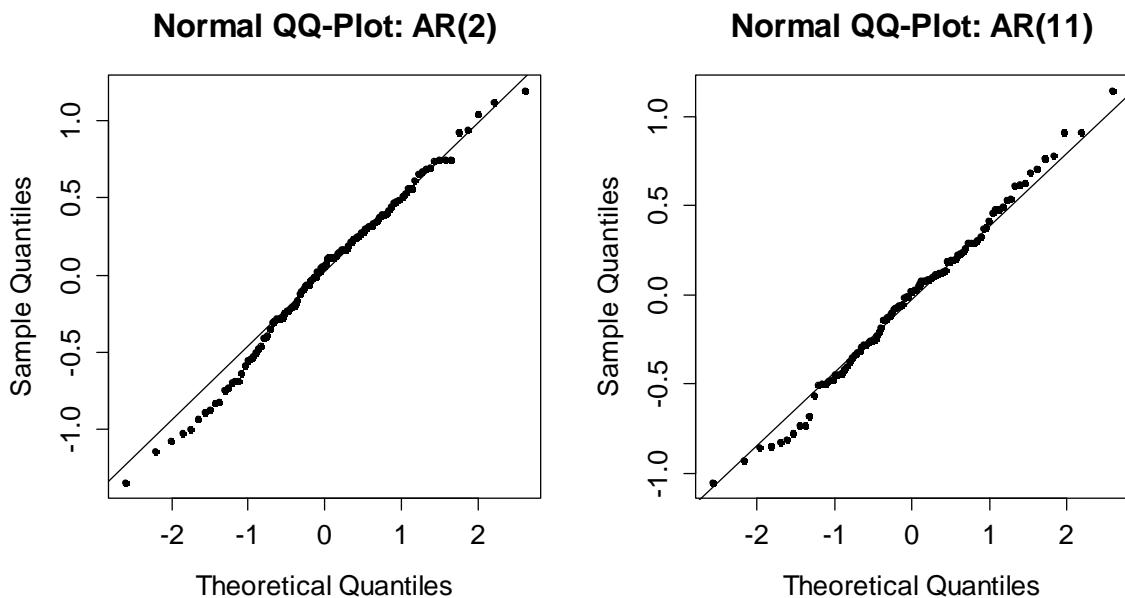


```
> acf(fit.ar02$resid, na.action=na.pass, ylim=c(-1,1))
> pacf(fit.ar02$resid, na.action=na.pass, ylim=c(-1,1))
> acf(fit.ar11$resid, na.action=na.pass, ylim=c(-1,1))
> pacf(fit.ar11$resid, na.action=na.pass, ylim=c(-1,1))
```



Because our estimation routines to some extent rely on the Gaussian distribution, it is always worthwhile to generate a Normal QQ-Plot for verifying this. Here, we obtain:

```
> par(mfrow=c(1, 2))
> qqnorm(as.numeric(fit.ar02$resid))
> qqline(as.numeric(fit.ar02$resid))
> qqnorm(as.numeric(fit.ar11$resid))
> qqline(as.numeric(fit.ar11$resid))
```



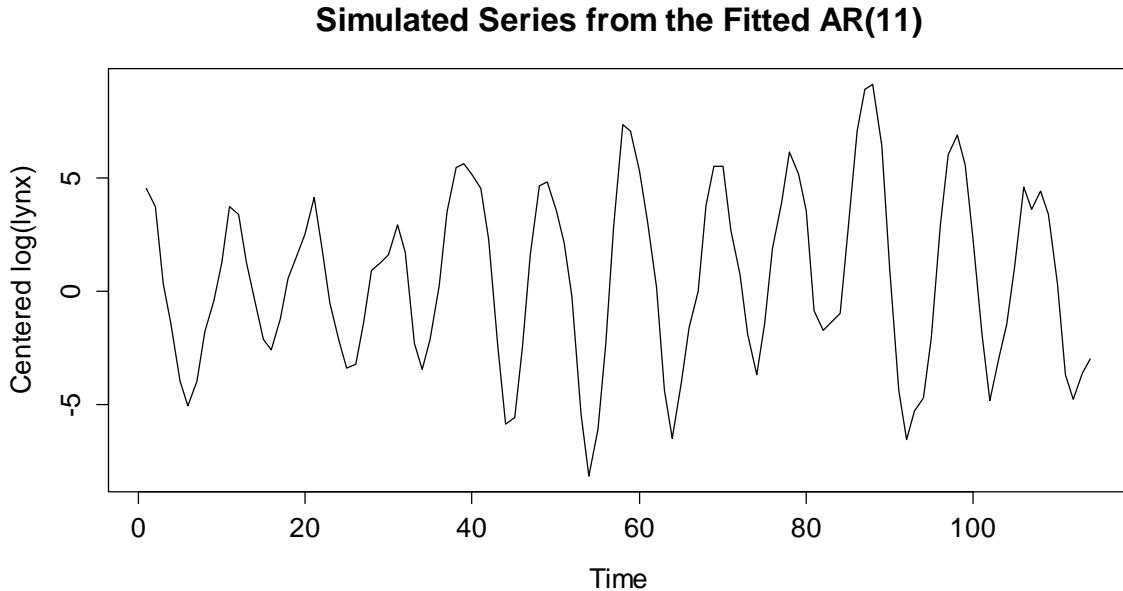
Neither of the two plots (which are very similar) does indicate any problems. If the residuals' distribution looks non-normal, a log-transformation might be worth a consideration, as it often improves the situation.

Simulation from the Fitted Model

If there are competing models and none of the other criterions dictate which one to use, another option is to generate realizations from the fitted process using R's function `arima.sim()`. It usually pays off to generate multiple realizations from each fitted model. By eyeballing, one then tries to judge which model yields data that resemble the true observations best. We here do the following:

```
> ## Repeat these commands a number of times
> plot(arima.sim(n=114, list(ar=fit.ar02$ar)))
> plot(arima.sim(n=114, list(ar=fit.ar11$ar)))
```

A realization from the fitted *AR(11)* can be seen on the next side. In summary, the simulations from this bigger model look more realistic than the ones from the *AR(2)*. The clearest answer about the model which is preferable here comes from the ACF/PACF correlograms, though. We conclude this section about model fitting by saying that the logged lynx data are best modeled with the *AR(11)*.



5.4 Moving Average Models

Here, we discuss moving average models. They are an extension of the White Noise process, i.e. X_t is as a linear combination of the current plus a few of the most recent innovation terms. As we will see, this leads to a time series process that is always stationary, but not iid. Furthermore, we will see that in many respects, moving average models are complementary to autoregressive models.

5.4.1 Definition and Properties

As we had mentioned above, a *moving average process of order q* , or abbreviated, an $MA(q)$ model for a series X_t is a linear combination of the current innovation term E_t , plus the q most recent ones E_{t-1}, \dots, E_{t-q} . The model equation is:

$$X_t = E_t + \beta_1 \cdot E_{t-1} + \dots + \beta_q \cdot E_{t-q}$$

We require that E_t is an innovation, which means independent and identically distributed, and also independent of any X_s where $s < t$. For simple notation, we can make use of the backshift operator and rewrite the model:

$$X_t = (1 + \beta_1 B + \dots + \beta_q B^q) E_t = \Theta(B) E_t$$

We call $\Theta(B)$ the characteristic polynomial of the $MA(q)$ process and obviously, it defines all properties of the series. As a remark, please note that a number of other textbooks define the $MA(q)$ process with negative signs for the β_j . While this is mathematically equivalent, we prefer our notation with the '+' signs, as it matches the way how things are implemented in R. We turn our sights towards the motivation for the moving average process.

What is the rationale for the $MA(q)$ process?

Firstly, they have been applied successfully in many applied fields, particularly in econometrics. Time series such as economic indicators are affected by a variety of random events such as strikes, government decision, referendums, shortages of key commodities, et cetera. Such events will not only have an immediate effect on the indicator, but may also affect its value (to a lesser extent) in several of the consecutive periods. Thus, it is plausible that moving average processes appear in practice. Moreover, some of their theoretical properties are in a nice way complementary to the ones of autoregressive processes. This will become clear if we study the moments and stationarity of the MA process.

Moments and Dependence

A first, straightforward but very important result is that any $MA(q)$ process X_t , as a linear combination of innovation terms, has zero mean and constant variance:

$$E[X_t] = 0 \text{ for all } t, \text{ and } Var(X_t) = \sigma_E^2 \cdot \left(1 + \sum_{j=1}^q \beta_j^2 \right) = const$$

Please note that in practice, we can always enhance $MA(q)$'s by adding a constant m that accounts for non-zero expectation of a time series, i.e. we can consider the shifted $MA(q)$ process

$$Y_t = m + X_t.$$

Hence, the zero mean property does not affect the possible field of practical application. Now, if we could additionally show that the autocovariance in MA processes is independent of the time t , we had already proven their stationarity. This is indeed the case. We start by considering a $MA(1)$ with $X_t = E_t + \beta_1 \cdot E_{t-1}$

$$\gamma(1) = Cov(X_t, X_{t-1}) = Cov(E_t + \beta_1 E_{t-1}, E_{t-1} + \beta_1 E_{t-2}) = \beta_1 \sigma_E^2.$$

For any lag k exceeding the order $q=1$, we use the same trick of plugging-in the model equation and directly obtain a perhaps somewhat surprising result:

$$\gamma(k) = Cov(X_t, X_{t-k}) = 0 \text{ for all } k > q = 1.$$

Thus, there is no more unconditional serial dependence in lags >1 . For the autocorrelation of a $MA(1)$ process, we have:

$$\rho(1) = \frac{\gamma(1)}{\gamma(0)} = \frac{\beta_1}{1 + \beta_1^2} \text{ and } \rho(k) = 0 \text{ for all } k > q = 1.$$

From this we conclude that $\rho(1) \leq 0.5$, no matter what the choice for β_1 is. Thus if in practice we observe a series where the first-order autocorrelation coefficient clearly exceeds this value, we have counterevidence to a $MA(1)$ process.

Furthermore, we have shown that any $MA(1)$ has zero mean, constant variance and an ACF that only depends on the lag k , hence it is stationary. Note that the stationarity does (in contrast to AR processes) not depend on the choice of the parameter β_1 . The stationarity property can be generalized to $MA(q)$ processes. Using some calculations and $\beta_0 = 1$, we obtain:

$$\rho(k) = \begin{cases} \sum_{j=0}^{q-k} \beta_j \beta_{j+k} / \sum_{j=0}^q \beta_j^2 & \text{for } k = 1, \dots, q \\ 0 & \text{for } k > q \end{cases}$$

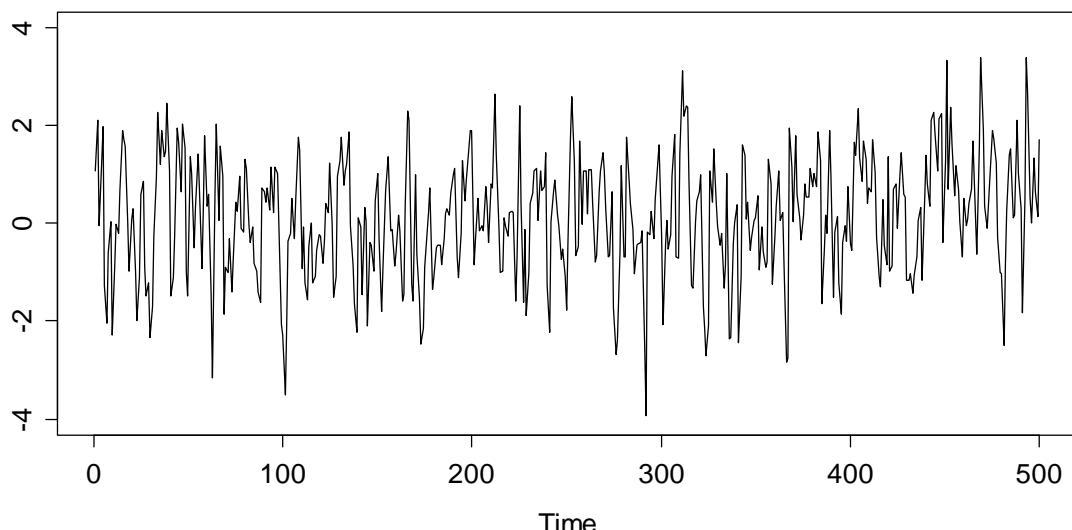
Hence, $\rho(k)$ is independent of the time t for any $MA(q)$ process, irrespective of the order q . The main results which can be derived from this property is that $MA(q)$ processes are *always stationary, independent of β_1, \dots, β_q* . Moreover, we learn from the above that the autocorrelation is zero for all orders $k > q$. And there obviously is a relation between the model parameters and the autocorrelation, although it gets quite complex for higher orders. While this formula may be employed for finding the true ACF of a given $MA(q)$, the most convenient way of doing this in practice remains with the R function `ARMAacf()`.

Example of a $MA(1)$

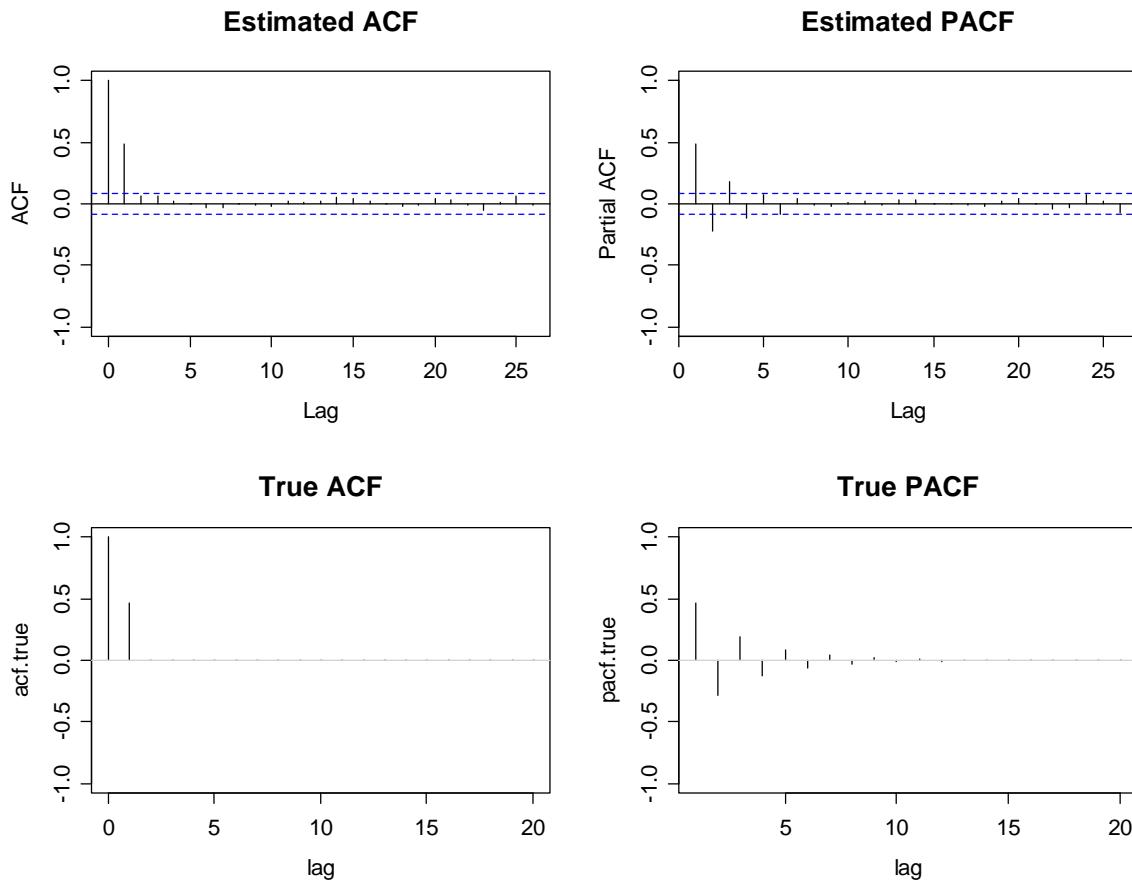
For illustration, we generate a realization consisting of 500 observations, from a $MA(1)$ process with $\beta_1 = 0.7$, and display time series plot, along with both estimated and true ACF/PACF.

```
> set.seed(21)
> ts.ma1 <- arima.sim(list(ma=0.7), n=500)
>
> plot(ts.ma1, ylab="", ylim=c(-4,4))
> title("Simulation from a MA(1) Process")
```

Simulation from a $MA(1)$ Process



```
> acf.true <- ARMAacf(ma=0.7, lag.max=20)
> pacf.true <- ARMAacf(ma=0.7, pacf=TRUE, lag.max=20)
```



We observe that the estimates are pretty accurate: the ACF has a clear cut-off, whereas the PACF seems to feature some alternating behavior with an exponential decay in absolute value. This behavior is typical: the PACF of any $MA(q)$ process shows an exponential decay, while the ACF has a cut-off. In this respect, $MA(q)$ processes are in full contrast to the $AR(p)$'s, i.e. the appearance of ACF and PACF is swapped.

Invertibility

It is easy to show that the first autocorrelation coefficient $\rho(1)$ of an $MA(1)$ process can be written in standard form, or also as follows:

$$\rho(1) = \frac{\beta_1}{1 + \beta_1^2} = \frac{1/\beta_1}{1 + (1/\beta_1)^2}$$

Apparently, any $MA(1)$ process with coefficient β_1 has exactly the same ACF as the one with $1/\beta_1$. Thus, the two processes $X_t = E_t + 0.5E_{t-1}$ and $U_t = E_t + 2E_{t-1}$ have the same dependency structure. Or in other words, given some ACF, we cannot identify the generating MA process uniquely. This problem of ambiguity leads to the concept of *invertibility*. Now, if we express the processes X_t and U_t in terms of X_{t-1}, X_{t-2}, \dots resp. U_{t-1}, U_{t-2}, \dots , we find by successive substitution:

$$\begin{aligned} E_t &= X_t - \beta_1 X_{t-1} + \beta_1^2 X_{t-2} - \dots \\ E_t &= U_t - (1/\beta_1) U_{t-1} + (1/\beta_1^2) U_{t-2} - \dots \end{aligned}$$

Hence, if we rewrite the $MA(1)$ as an $AR(\infty)$, only one of the processes will converge. That is the one where $|\beta_1| < 1$, and it will be called *invertible*. It is important to know that invertibility of MA processes is central when it comes to fitting them to data, because parameter estimation is based on rewriting them in the form of an $AR(\infty)$.

For higher-order $MA(q)$ processes, the property of invertibility is equally central. If it is met, the series can be rewritten in form of an $AR(\infty)$ and it is guaranteed that there is a unique MA process for any given ACF. Invertibility of a $MA(q)$ is met if the roots of the characteristic polynomial $\Theta(B)$ all lie outside of the unit circle. As was explained earlier in chapter 5.3.1, we can verify this using the **R** function `polyroot()`. Please note that the estimation procedure described below will always result in coefficients $\hat{\beta}_1, \dots, \hat{\beta}_q$ that define an invertible $MA(q)$ process.

5.4.2 Fitting

The process of fitting $MA(q)$ models to data is more difficult than for $AR(p)$, as there are no (efficient) explicit estimators and numerical optimization is mandatory. Perhaps the simplest idea for estimating the parameters is to exploit the relation between the model parameters and the autocorrelation coefficients, i.e.:

$$\rho(k) = \begin{cases} \sum_{j=0}^{q-k} \beta_j \beta_{j+k} / \sum_{j=0}^q \beta_j^2 & \text{for } k = 1, \dots, q \\ 0 & \text{for } k > q \end{cases}$$

Hence in case of a $MA(1)$, we would determine $\hat{\beta}_1$ by plugging-in $\hat{\rho}(1)$ into the equation $\rho(1) = \beta_1 / (1 + \beta_1^2)$. This can be seen as an analogon to the Yule-Walker approach in AR modelling. Unfortunately, the plug-in idea yields an inefficient estimator and is not a viable option for practical work.

Conditional Sum of Squares

Another appealing idea would be to use some (adapted) least squares procedure for determining the parameters. A fundamental requirement for doing so is that we can express the sum of squared residuals $\sum E_t^2$ in terms of the observations X_1, \dots, X_n and the parameters β_1, \dots, β_q only, and do not have to rely on the unobservable E_1, \dots, E_n directly. This is (up to the choice of some initial values) possible for all invertible $MA(q)$ processes. For simplicity, we restrict our illustration to the $MA(1)$ case, where we can replace any innovation term E_t by:

$$E_t = X_t - \beta_1 X_{t-1} + \beta_1^2 X_{t-2} + \dots + (-\beta_1)^{t-1} X_1 + \beta_1^t E_0$$

By doing so, we managed to express the innovation/residual at time t as a function of the model parameter β_1 and a combination of the current and past observations of the series. What is also remaining is the (hypothetical) initial innovation term E_0 . Conditional on the assumption $E_0 = 0$, we can indeed rewrite the residuals sum of squares $\sum E_t^2$ of any $MA(1)$ using X_1, \dots, X_n and β_1 only. However, there is no closed form solution for the minimization of $\sum E_t^2$, since powers of the parameter β_1 appear; but the problem can be tackled using numerical optimization. This approach is known as the Conditional Sum of Squares (CSS) method. It works similarly for higher orders q , i.e. fundamentally relies on the invertibility of the $MA(q)$ and assumes that $E_t = 0$ for all $t = -\infty, \dots, 0$. In R, the method is implemented in function `arima()` if argument `method = "CSS"` is set.

Maximum-Likelihood Estimation

As can be seen from the R help file, the Conditional Sum of Squares method is only secondary to `method = "CSS-ML"` in the R function `arima()`. This means that it is preferable to use CSS only to obtain a first estimate of the coefficients β_1, \dots, β_q . They are then used as starting values for a Maximum-Likelihood estimation, which is based on the assumption of Gaussian innovations E_t . It is pretty obvious that $X_t = E_t + \beta_1 E_{t-1} + \dots + \beta_q E_{t-q}$, as a linear combination of normally distributed random variables, follows a Gaussian too. By taking the covariance terms into account, we obtain a multivariate Gaussian for the time series vector:

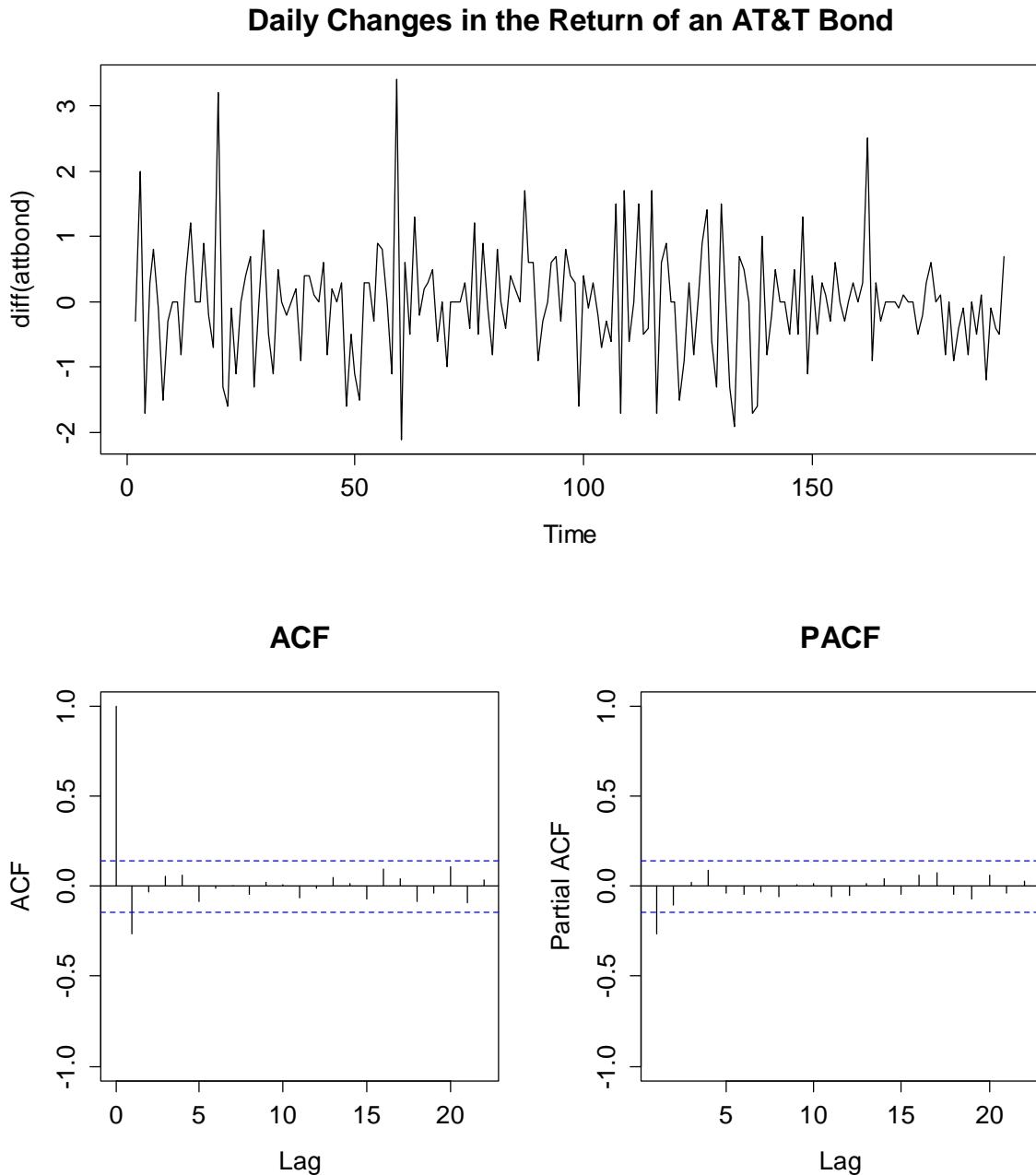
$$X = (X_1, \dots, X_n) \sim N(0, V), \text{ resp. } Y = (Y_1, \dots, Y_n) \sim N(m \cdot \underline{1}, V).$$

MLE then relies on determining the parameters m (if a shifted $MA(q)$ is estimated), β_1, \dots, β_q and σ_E^2 simultaneously by maximizing the probability density function of the above multivariate Gaussian with assuming the data x_1, \dots, x_n as given quantities. This is a quite complex non-linear problem which needs to be solved numerically. A good implementation is found in R's `arima()`.

The benefit of MLE is that (under mild and in practice usually fulfilled conditions) certain optimality conditions are guaranteed. It is well known that the estimates are asymptotically normal with minimum variance among all asymptotically normal estimators. Additionally, it is pretty easy to derive standard errors for the estimates, which further facilitates their interpretation. And even though MLE is based on assuming Gaussian innovations, it still produces reasonable results if the deviations from that model are not too strong. Be especially wary in case of extremely skewed data or massive outliers. In such cases, applying a log-transformation before the modelling/estimation starts is a wise idea.

5.4.3 Example: Return of AT&T Bonds

As an example, we consider the daily changes in the return of an AT&T bond from April 1975 to December 1975, which makes for a total of 192 observations. The data are displayed along with their ACF and PACF on the next page.



The series seems to originate from a stationary process. There are no very clear cycles visible, hence it is hard to say anything about correlation and dependency, and it is impossible to identify the stochastic process behind the generation of these data from a time series plot alone. Using the ACF and PACF as a visual aid, we observe a pretty clear cut-off situation in the ACF at lag 1 which lets us assume that a $MA(1)$ might be suitable. That opinion is undermined by the fact that the PACF drops off to small values quickly, i.e. we can attribute some exponential decay to it for lags 1 and 2.

Our next goal is now to fit the $MA(1)$ to the data. As explained above, the simplest solution would be to determine $\hat{\rho}(1) = -0.266$ and derive $\hat{\beta}_1$ from $\rho(1) = \beta_1 / (1 + \beta_1^2)$. This yields two solutions, namely $\hat{\beta}_1 = -0.28807$ and $\hat{\beta}_1 = -3.47132$. Only one of these (the former) defines an invertible $MA(1)$, hence we would stick to that

solution. A better alternative is to use the CSS approach for parameter estimation. The code for doing so is as follows:

```
> arima(diff(atbond), order=c(0,0,1), method="CSS")

Call:
arima(x = diff(atbond), order = c(0, 0, 1), method = "CSS")

Coefficients:
      ma1  intercept
     -0.2877    -0.0246
  s.e.   0.0671     0.0426

sigma^2 estimated as 0.6795:  part log likelihood = -234.11
```

Even more elegant and theoretically sound is the MLE. We can also perform this in R using function `arima()`. It yields a very similar but not identical result:

```
> arima(diff(atbond), order=c(0,0,1))

Call:
arima(x = diff(atbond), order = c(0, 0, 1))

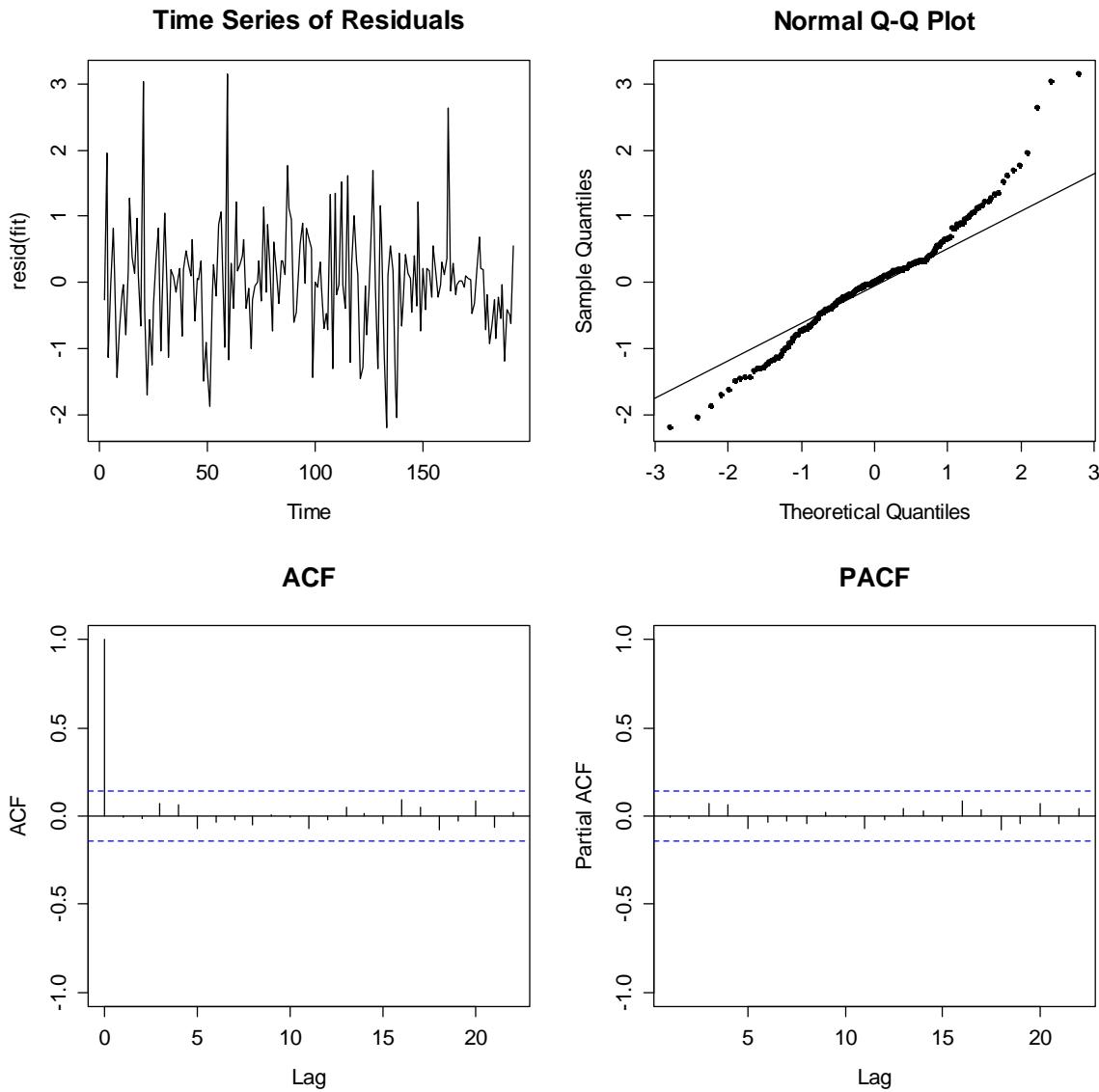
Coefficients:
      ma1  intercept
     -0.2865    -0.0247
  s.e.   0.0671     0.0426

sigma^2 = 0.6795: log likelihood = -234.16, aic = 474.31
```

Please note that the application of the three estimation procedures here was just for illustrative purposes, and to show the (slight) differences that manifest themselves when different estimators are employed. In any practical work, you can easily restrict yourself to the application of the `arima()` procedure using the default fitting by `method= "CSS-ML"`.

For verifying the quality of the fit, a residual analysis is mandatory. The residuals of the $MA(1)$ are estimates of the innovations E_t . The model can be seen as adequate if the residuals reflect the main properties of the innovations. Namely, they should be stationary and free of any dependency, as well as approximately Gaussian. We can verify this by producing a time series plot of the residuals, along with their ACF and PACF, and a Normal QQ-Plot. Sometimes, it is also instructive to plot the fitted values into the original data, or to simulate from the fitted process, since this further helps verifying that the fit is good.

```
> par(mfrow=c(2,2))
> fit <- arima(diff(atbond), order=c(0,0,1))
> plot(resid(fit))
> qqnorm(resid(fit)); qqline(resid(fit))
> acf(resid(fit)); pacf(resid(fit))
```



There are no (partial) autocorrelations that exceed the confidence bounds, hence we can safely conjecture that the residuals are not correlated and hence, all the dependency signal has been captured by the $MA(1)$. When inspecting the time series of the residuals, it seems stationary. However, what catches the attention is the presence of three positive outliers and the fact that the residuals are generally long-tailed. We might try to overcome this problem by a log-transformation, but this is left to the reader.

5.5 ARMA(p,q) Models

Here, we discuss models that feature both dependency on previous observations X_{t-1}, X_{t-2}, \dots as well as previous innovations terms E_{t-1}, E_{t-2}, \dots . Thus, they are a hybrid between $AR(p)$ and $MA(q)$ models, and aptly named $ARMA(p,q)$. Their importance lies in the fact that it is possible to model a far wider spectrum of dependency structures, and that they are parsimonious: often, an $ARMA(p,q)$ requires (far) fewer parameters than pure AR or MA processes would.

5.5.1 Definition and Properties

The formal definition of an $ARMA(p,q)$ process is as follows:

$$X_t = \alpha_1 X_{t-1} + \dots + \alpha_p X_{t-p} + E_t + \beta_1 E_{t-1} + \dots + \beta_q E_{t-q}$$

As before, we assume that E_t is causal and White Noise, i.e. an innovation with mean $E[E_t] = 0$ and finite variance $Var(E_t) = \sigma_E^2$. It is much more convenient to use the characteristic polynomials $\Phi(\cdot)$ for the AR part, and $\Theta(\cdot)$ for the MA part, because this allows for a very compact notation:

$$\Phi(B)X_t = \Theta(B)E_t.$$

It is obvious that all relevant properties of an $ARMA(p,q)$ process lie in the characteristic polynomials. If the roots of $\Phi(\cdot)$ are outside of the unit circle, the process will be stationary and have mean zero. On the other hand, if the roots of $\Theta(\cdot)$ are outside of the unit circle, the process is invertible. Both properties are important for practical application. If they are met, we can rewrite any $ARMA(p,q)$ in the form of a $AR(\infty)$ or an $MA(\infty)$. This explains why fitting an $ARMA(p,q)$ can in practice often be replaced by fitting AR - or MA -models with high orders (although it is not a good idea to do so!). As has been argued above, any stationary $ARMA(p,q)$ has mean zero, i.e. $E[X_t] = 0$. Thus, in practice we will often consider shifted $ARMA$ -processes that are of the form:

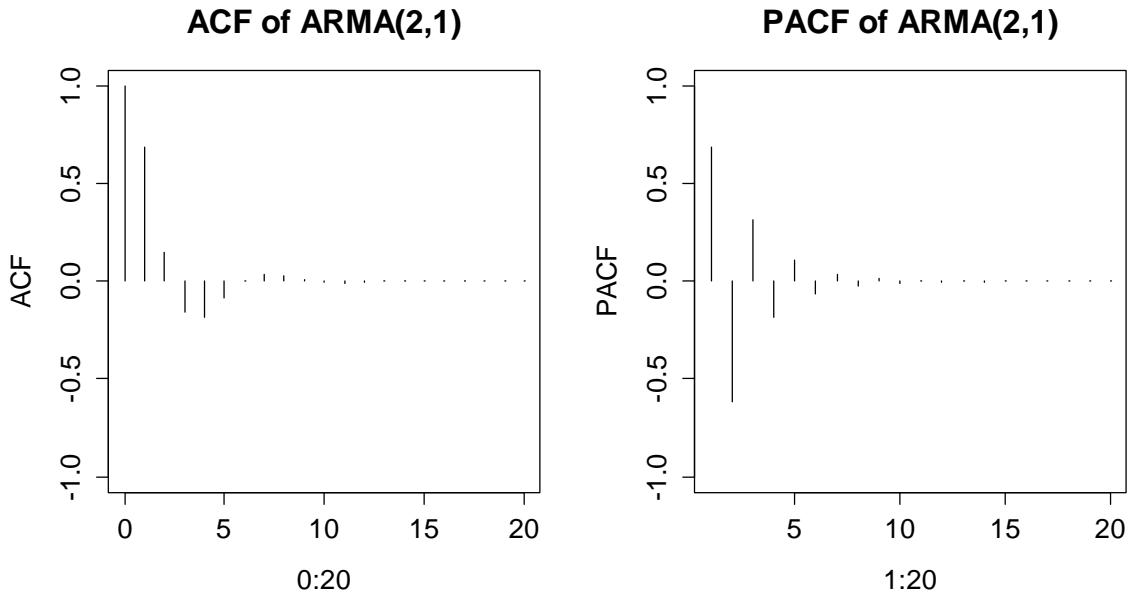
$$Y_t = m + X_t, \text{ where } X_t \text{ is an } ARMA(p,q).$$

In principle, it is straightforward to derive the ACF of an $ARMA(p,q)$, though algebraically a bit tedious. Given the applied focus of this scriptum, we do without and focus on the results and consequences instead. We illustrate the typical behavior of the $ARMA$ autocorrelations on the basis of an example. Namely, we consider the $ARMA(2,1)$ defined by:

$$X_t - 0.8X_{t-1} + 0.4X_{t-2} = E_t + 0.6E_{t-1}$$

On the next page, we exhibit the (theoretical) ACF and PACF. It is typical that neither the ACF nor the PACF cut-off strictly at a certain lag. Instead, they both show some infinite behavior, i.e. an exponential decay in the magnitude of the coefficients. However, superimposed on that is a sudden drop-off in both ACF and PACF. In our example, it is after lag 1 in the ACF, as induced by the moving average order $q=1$. In the PACF, the drop-off happens after lag 2, which is the logical consequence of the autoregressive order of $p=2$. The general behavior of the ACF and PACF is summarized in the table below.

Model	ACF	PACF
$AR(p)$	infinite / exp. decay	cut-off at lag p
$MA(q)$	cut-off at lag q	infinite / exp. decay
$ARMA(p,q)$	infinite / mix of decay & cut-off	infinite / mix of decay & cut-off



It is important to know that with $ARMA(p,q)$ processes, a wealth of autocorrelation structures can be generated. As to how visible the two cut-offs in ACF and PACF are, resp. whether the cut-off or the decay is dominating, depends on the model's coefficients. There are $ARMA$'s where the AR part is dominating, there are others where the MA is stronger, and of course they can also be on an equal footing.

5.5.2 Fitting

The above properties of ACF and PACF can be exploited for choosing the type and order of a time series model. In particular, if neither the ACF nor the PACF shows a pronounced cut-off, where after some low lag (i.e. p or $q < 10$) all the following correlations are non-significantly different from zero, then it is usually wise to choose an $ARMA(p,q)$. For determining the order (p,q) , we search for the superimposed cut-off in the ACF (for q), respectively PACF (for p). The drop-off is not always easy to identify in practice. In “difficult” situations, it has also proven beneficial to support the choice of the right order with the AIC criterion. We could for example perform a grid search on all possible $ARMA(p,q)$ models which do not use more than 5 parameters, i.e. $p+q < 5$. This can readily be done in **R** by programming a `for()` loop.

It is very important to know that $ARMA$ models are parsimonious, i.e. they usually do not require high orders p and q . In particular, they work well with far fewer parameters than pure AR or MA models would. Or in other words: often it is possible to fit high-order $AR(p)$'s or $MA(q)$'s instead of a low-order $ARMA(p,q)$. That property does not come as a surprise: as we conjectured above, any stationary and invertible $ARMA$ can be represented in the form of an $AR(\infty)$ or an $MA(\infty)$. However, this is not a good idea in practice: estimating parameters “costs money”, i.e. will lead to less precise estimates. Thus, a low-order $ARMA(p,q)$ is always to be preferred over a high-order $AR(p)$ or $MA(q)$.

For estimating the coefficients, we are again confronted with the fact that there are no explicit estimators available. This is due to the *MA* component in the model which involves innovation terms that are unobservable. By rearranging terms in the model equation, we can again represent any *ARMA*(p, q) in a form where it only depends on the observed X_t , the coefficients $\alpha_1, \dots, \alpha_p; \beta_1, \dots, \beta_q$ and some previous innovations E_t with $t < 1$. If these latter terms are all set to zero, we can determine the optimal set of model coefficients by minimizing the sum of squared residuals (i.e. innovations) with a numerical method. This is the CSS approach that was already mentioned in 5.4.2 and is implemented in function `arima()` when `method = "CSS"`. By default however, these CSS estimates are only used as starting values for a MLE. If Gaussian innovations are assumed, then the joint distribution of any *ARMA*(p, q) process vector $X = (X_1, \dots, X_n)$ has a multivariate normal distribution.

$$X = (X_1, \dots, X_n) \sim N(0, V), \text{ resp. } Y = (Y_1, \dots, Y_n) \sim N(m \cdot \underline{1}, V).$$

MLE then relies on determining the parameters m (if a shifted *ARMA*(p, q) is estimated), $\alpha_1, \dots, \alpha_p; \beta_1, \dots, \beta_q$ and σ_E^2 simultaneously by maximizing the probability density function of the above multivariate Gaussian with assuming the data x_1, \dots, x_n as given quantities. This is a quite complex non-linear problem which needs to be solved numerically. A good implementation is found in R's `arima()`. As was stated previously, the benefit of MLE is that (under mild and mostly met conditions) some optimality is guaranteed. In particular, the estimates are asymptotically normal with minimum variance among all asymptotically normal estimators. Another benefit is provided by the standard errors which allow for judging the precision of the estimates.

6 SARIMA and GARCH Models

As we have discovered previously, many time series are non-stationary due to deterministic trends and/or seasonal effects. While we have learned to remove these and then explain the remainder with some time series models, there are other models that can directly incorporate trend and seasonality. While they usually lack some transparency for the decomposition, their all-in-one approach allows for convenient forecasting, and also AIC-based decisions for choosing the right amount of trend and seasonality modeling become feasible.

Time series from financial or economic background often show serial correlation in the conditional variance, i.e. are conditionally heteroskedastic. This means that they exhibit periods of high and low volatility. Understanding the behavior of such series pays off, and the usual approach is to set up autoregressive models for the conditional variance. These are the famous ARCH models, which we will discuss along with their generalized variant, the GARCH class.

6.1 ARIMA Models

ARIMA models are aimed at describing series which exhibit a deterministic trend that can be removed by differencing; and where these differences can be described by an $ARMA(p,q)$ model. Thus, the definition of an $ARIMA(p,d,q)$ process arises naturally:

Definition: A series X_t follows an $ARIMA(p,d,q)$ model if the d th order difference of X_t is an $ARMA(p,q)$ process. If we introduce

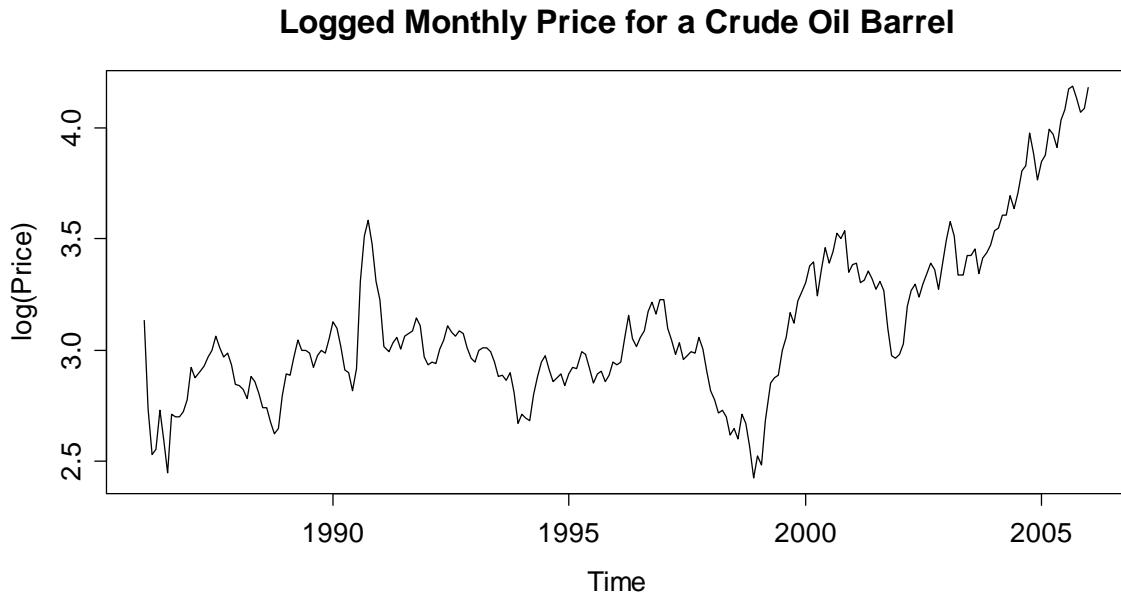
$$Y_t = (1 - B)^d X_t,$$

where B is the backshift operator, then we can write the $ARIMA$ process using the characteristics polynomials, i.e. $\Theta(\cdot)$ that accounts for the AR , and $\Phi(\cdot)$ that stands for the MA part.

$$\begin{aligned} \Phi(B)Y_t &= \Theta(B)E_t \\ \Phi(B)(1 - B)^d X_t &= \Theta(B)E_t \end{aligned}$$

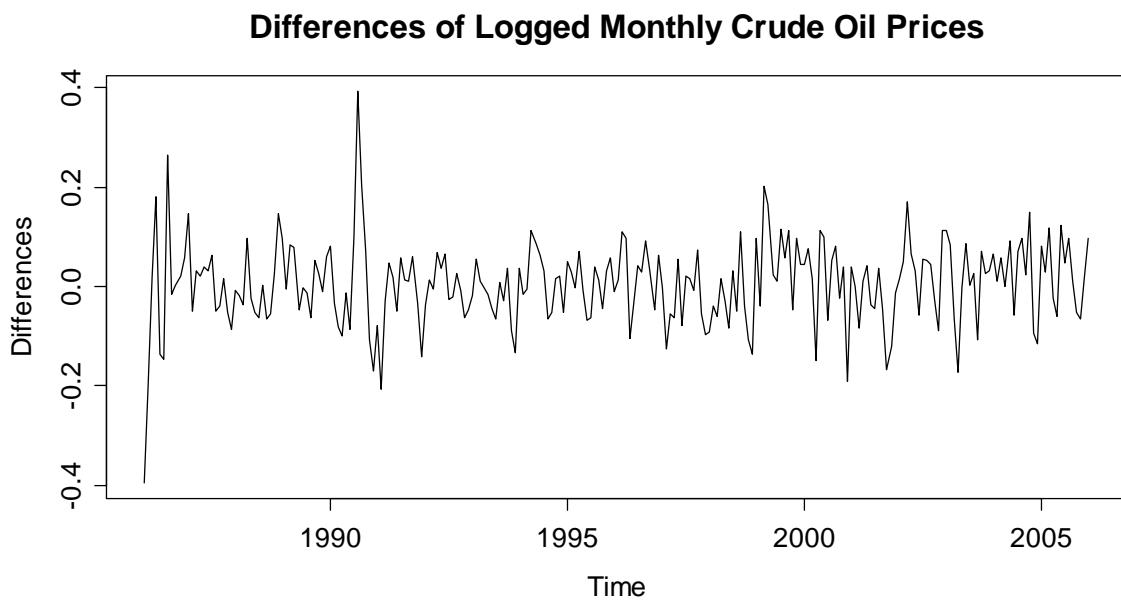
Such series do appear in practice, as our example of the monthly prices for a barrel of crude oil (in US\$) from January 1986 to January 2006 shows. To stabilize the variance, we decide to log-transform the data, and model these.

```
> library(TSA)
> data(oil.price)
> lop <- log(oil.price)
> plot(lop, ylab="log(Price)")
> title("Logged Monthly Price for a Crude Oil Barrel")
```



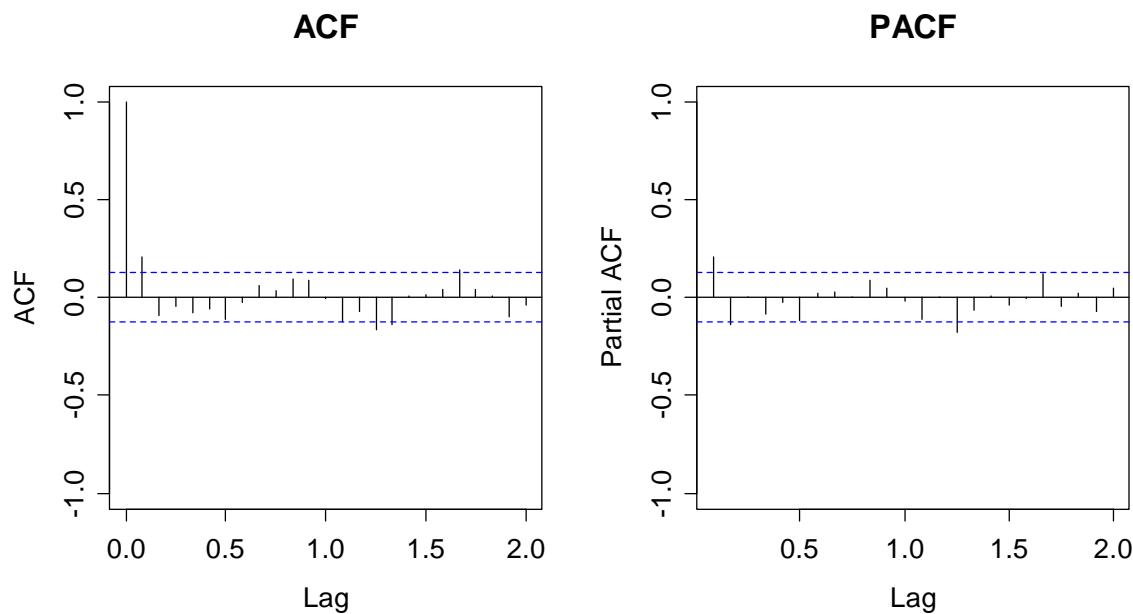
The series does not exhibit any apparent seasonality, but there is a clear trend, so that it is non-stationary. We try first-order (i.e. $d = 1$) differencing, and then check whether the result is stationary.

```
> dlop <- diff(lop)
> plot(dlop, ylab="Differences")
> title("Differences of Logged Monthly Crude Oil Prices")
```



The trend was successfully removed by taking differences. ACF and PACF show that the result is serially correlated. There may be a drop-off in the ACF at lag 1, and in the PACF at either lag 1 or 2, suggesting an *ARIMA*(1,1,1) or an *ARIMA*(2,1,1) for the logged oil prices. However, the best performing model is somewhat surprisingly an *ARIMA*(1,1,2), for which we show the results.

```
> par(mfrow=c(1, 2))
> acf(dlop, main="ACF", ylim=c(-1,1), lag.max=24)
> pacf(dlop, main="PACF", ylim=c(-1,1), lag.max=24)
```



The fitting can be done with the `arima()` procedure that (by default) estimates the coefficients using Maximum Likelihood with starting values obtained from the Conditional Sum of Squares method. We can either let the procedure do the differencing:

```
> arima(lop, order=c(1,1,2))

Call:
arima(x = lop, order = c(1, 1, 2))

Coefficients:
      ar1      ma1      ma2
    0.8429   -0.5730   -0.3104
  s.e.  0.1548    0.1594    0.0675

sigma^2 = 0.006598:  log likelihood = 261.88,  aic = -515.75
```

Or, we can use the differenced series `dlop` as input and fit an ARMA(1,2). However, we need to tell `R` to not include an intercept – this is not necessary when the trend was removed by taking differences. The command is:

```
> arima(dlop, order=c(1,0,2), include.mean=FALSE)
```

The output from this is exactly the same as above. The next step is to perform residual analysis – if the model is appropriate, they must look like White Noise. This is (data not shown here) more or less the case. For decisions on the correct model order, also the AIC statistics can provide valuable information.

We finish this section by making some considerations on the model equation. We have:

$$\begin{aligned} Y_t &= 0.84 \cdot Y_{t-1} + E_t - 0.57 \cdot E_{t-1} - 0.31 \cdot E_{t-2} \\ X_t - X_{t-1} &= 0.84 \cdot (X_{t-1} - X_{t-2}) + E_t - 0.57 \cdot E_{t-1} - 0.31 \cdot E_{t-2} \\ X_t &= 1.84 \cdot X_{t-1} - 0.84 \cdot X_{t-2} + E_t - 0.57 \cdot E_{t-1} - 0.31 \cdot E_{t-2} \end{aligned}$$

Thus, the *ARIMA*(1,1,2) can be rewritten as a non-stationary *ARMA*(2,2). The non-stationarity is due to the roots of the AR characteristic polynomial, which are within the unit circle. Finally, we give some recipe for fitting *ARIMA* models:

- 1) Choose the appropriate order of differencing, usually $d = 1$ or (in rare cases) $d = 2$, such that the result is a stationary series.
- 2) Analyze ACF and PACF of the differenced series. If the stylized facts of an *ARMA* process are present, decide for the orders p and q .
- 3) Fit the model using the `arima()` procedure. This can be done on the original series by setting d accordingly, or on the differences, by setting $d = 0$ and argument `include.mean=FALSE`.
- 4) Analyze the residuals; these must look like White Noise. If several competing models are appropriate, use AIC to decide for the winner.

The fitted ARIMA model can then be used to generate forecasts including prediction intervals. This will, however, only be discussed in section 8.

6.2 SARIMA Models

After becoming acquainted with the *ARIMA* models, it is quite natural to ask for an extension to seasonal series; especially, because we learned that differencing at a lag equal to the period s does remove seasonal effects, too. Suppose we have a series X_t with monthly data. Then, series

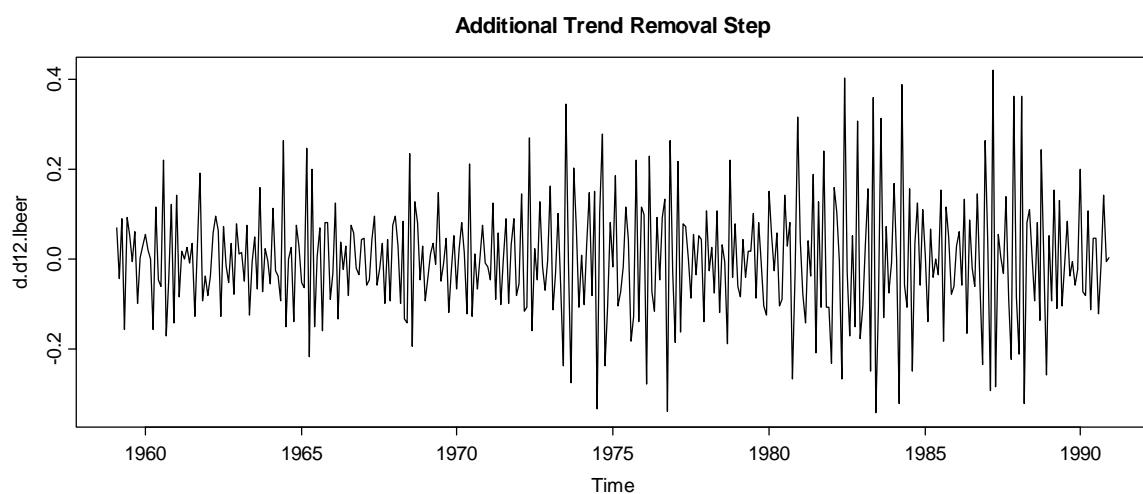
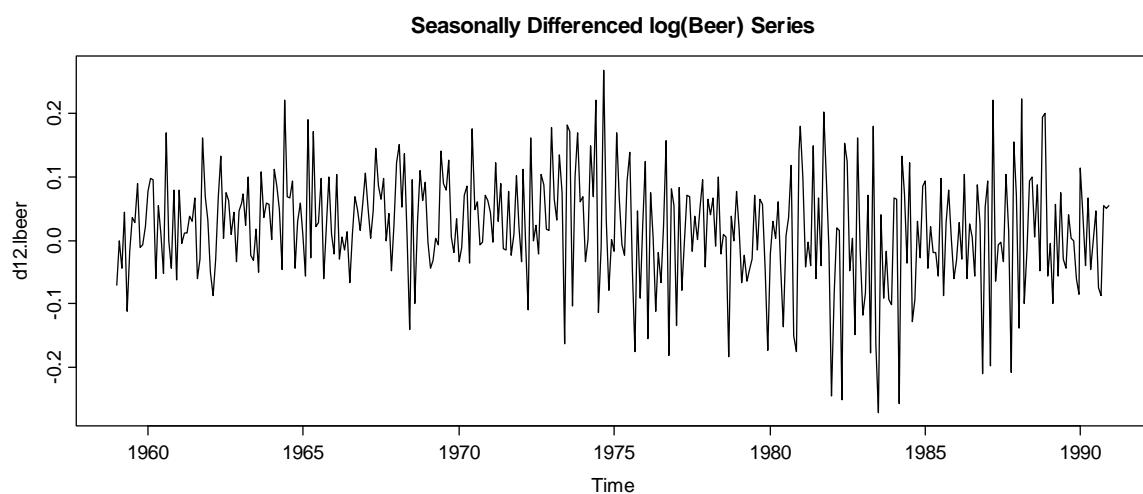
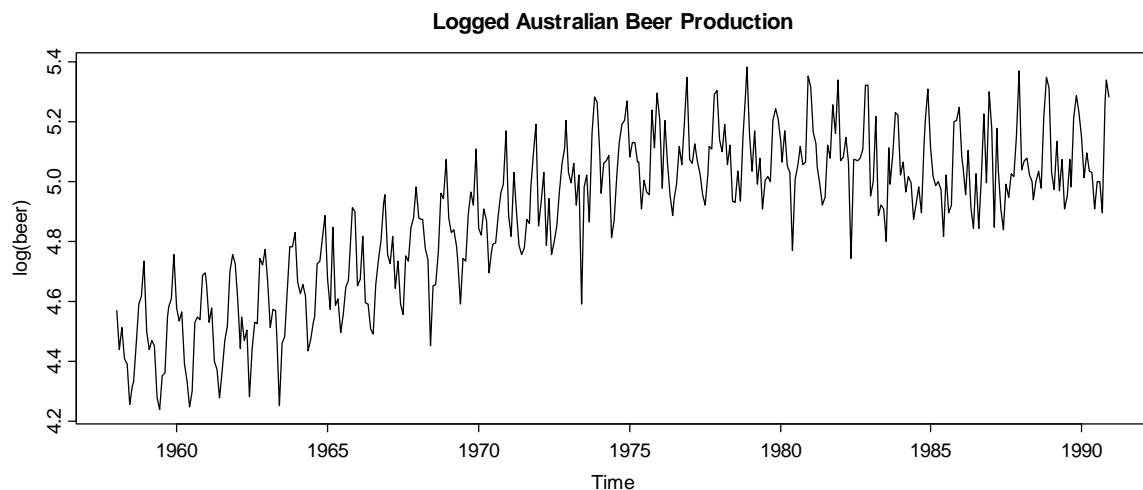
$$Y_t = X_t - X_{t-12} = (1 - B^{12})X_t$$

has the seasonality removed. However, it is quite often the case that the result has not yet constant mean, and thus, some further differencing at lag 1 is required to achieve stationarity:

$$Z_t = Y_t - Y_{t-1} = (1 - B)Y_t = (1 - B)(1 - B^{12})X_t = X_t - X_{t-1} - X_{t-12} + X_{t-13}$$

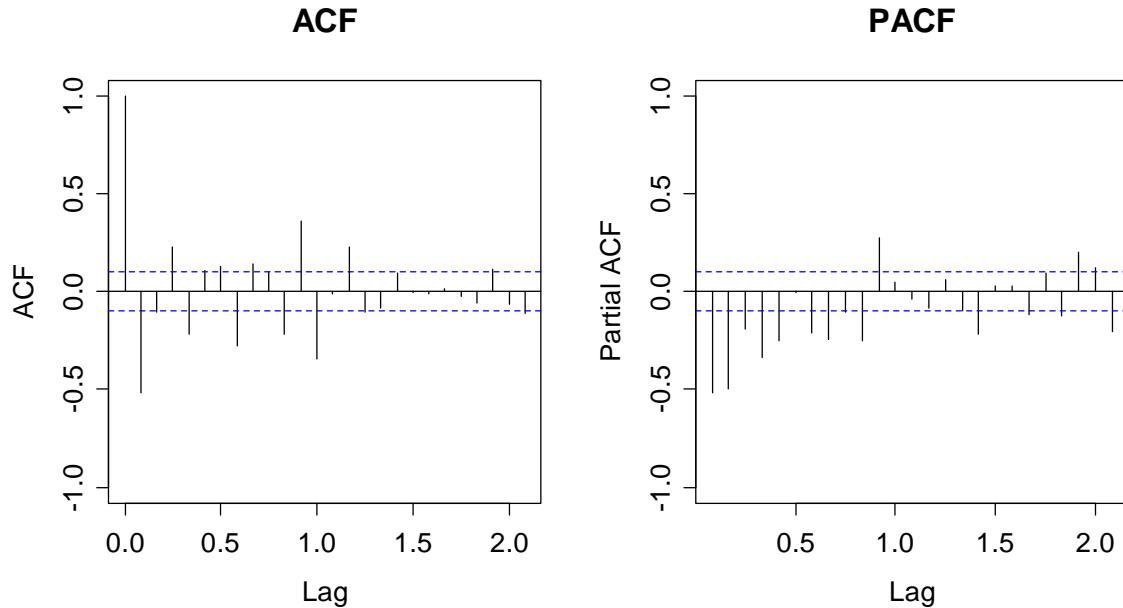
We illustrate this using the Australian beer production series that we had already considered in section 4. It has monthly data that range from January 1958 to December 1990. Again, a log-transformation to stabilize the variance is indicated. On the next page, we display the original series X_t , the seasonally differenced series Y_t and finally the seasonal-trend differenced series Z_t .

```
> www <- "http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/"
> dat <- read.table(paste(www,"cbe.dat",sep="", header=T)
> beer      <- ts(dat$beer, start=1958, freq=12)
> d12.lbeer <- diff(log(beer), lag=12)
> d.d12.lbeer <- diff(d12.lbeer)
> plot(log(beer))
> plot(d12.lbeer)
> plot(d.d12.lbeer))
```



While the two series X_t and Y_t are non-stationary, the last one, Z_t may be, although it is a bit debatable whether the assumption of constant variation is violated or not. We proceed by analyzing ACF and PACF of series Z_t .

```
> par(mfrow=c(1, 2))
> acf(d.d12.lbeer, ylim=c(-1,1))
> pacf(d.d12.lbeer, ylim=c(-1,1), main="PACF")
```



There is very clear evidence that series Z_t is serially dependent, and we could try an $ARMA(p, q)$ to model this dependence. As for the choice of the order, this is not simple on the basis of the above correlograms. They suggest that high values for p and q are required, and model fitting with subsequent residual analysis and AIC inspection confirm this: $p=14$ and $q=11$ yield a good result.

It is (not so much in the above, but generally when analyzing data of this type) quite striking that the ACF and PACF coefficients have large values at multiples of the period s . This is very typical behavior for seasonally differenced series, in fact it originates from the evolution of resp. changes in the seasonality over the years. A simple model accounting for this is the so-called *airline model*:

$$\begin{aligned} Z_t &= (1 + \beta_1 B)(1 + \xi_1 B^{12}) E_t \\ &= (1 + \beta_1 B + \xi_1 B^{12} + \beta_1 \xi_1 B^{13}) E_t \\ &= E_t + \beta_1 E_{t-1} + \xi_1 E_{t-12} + \beta_1 \xi_1 E_{t-13} \end{aligned}$$

This is a MA(13) model, where many of the coefficients are equal to 0. Because it was made up of an MA(1) with B as an operator in the characteristic polynomial, and another one with B^s as the operator, we call this a SARIMA(0,1,1)(0,1,1)¹². This idea can be generalized: we fit AR and MA parts with both B and B^s as operators in the characteristic polynomials, which again results in a high order ARMA model for Z_t .

Definition: A series X_t follows a $SARIMA(p,d,q)(P,D,Q)^s$ process if the following equation holds:

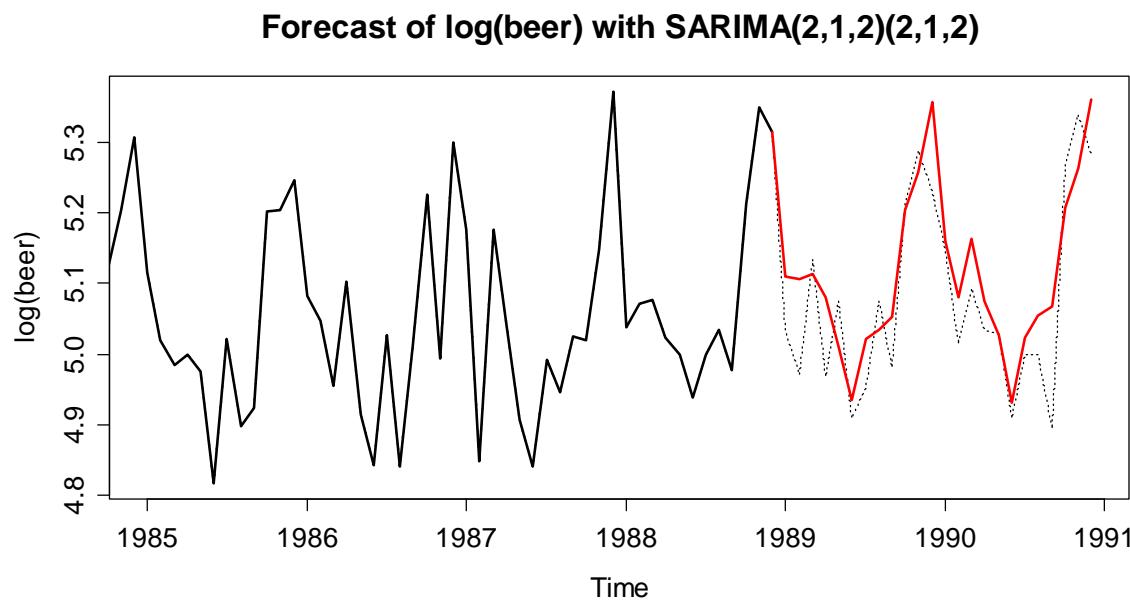
$$\Phi(B)\Phi_S(B^s)Z_t = \Theta(B)\Theta_S(B^s)E_t,$$

where series Z_t originated from X_t after appropriate seasonal and trend differencing, i.e. $Z_t = (1-B)^d(1-B^s)^D$.

Fortunately, it turns out that usually $d = D = 1$ is enough. As for the model orders p, q, P, Q , the choice can be made on the basis of ACF and PACF, by searching for cut-offs. Mostly, these are far from evident, and thus, an often applied alternative is to consider all models with $p, q, P, Q \leq 2$ and doing an AIC-based grid search.

For our example, the $SARIMA(2,1,2)(2,1,2)^{12}$ has the lowest value and also shows satisfactory residuals, although it seems to perform slightly less well than the $SARIMA(14,1,11)(0,1,0)^{12}$. The R-command for the former is:

```
> fit <- arima(log(beer), order=c(2,1,2), seasonal=c(2,1,2))
```



As it was mentioned in the introduction to this section, one of the main advantages of ARIMA and SARIMA models is that they allow for quick and convenient forecasting. While this will be discussed in depth later in section 8, we here provide a first example to show the potential.

From the logged beer production data, the last 2 years were omitted before the SARIMA model was fitted to the (shortened) series. On the basis of this model, a 2-year-forecast was computed, which is displayed by the red line in the plot above. The original data are shown as a solid (insample, 1985-1988) line, respectively as a dotted (out-of-sample, 1989-1990) line. We see that the forecast is reasonably accurate.

To facilitate the fitting of *SARIMA* models, we finish this chapter by providing some guidelines:

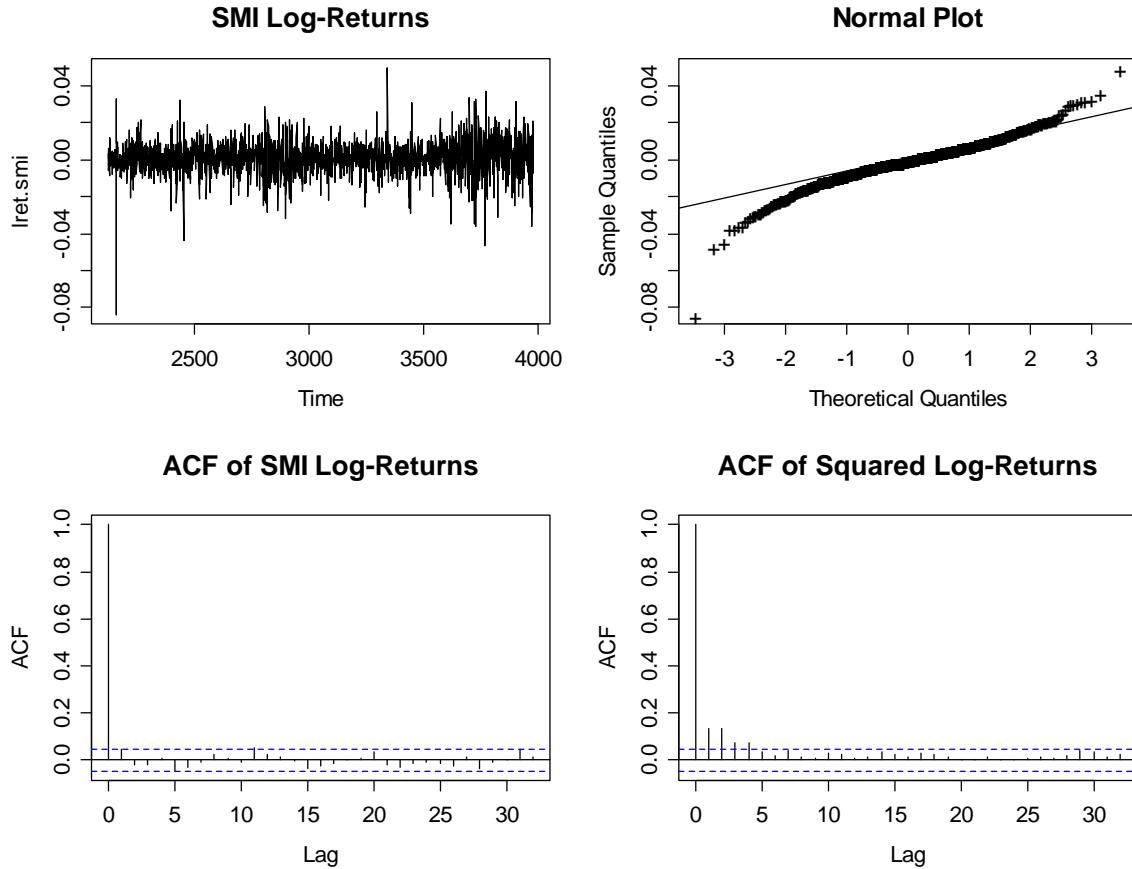
- 1) Perform seasonal differencing on the data. The lag s is determined by the periodicity of the data, for the order, in most cases $D = 1$ is sufficient.
- 2) Do a time series plot of the output of the above step. Decide whether it is stationary, or whether additional differencing at lag 1 is required to remove a potential trend. If not, then $d = 0$, and proceed. If yes, $d = 1$ is enough for most series.
- 3) From the output of step 2, i.e. series Z_t , generate ACF and PACF plots to study the dependency structure. Look for coefficients/cut-offs at low lags that indicate the direct, short-term dependency and determine orders p and q . Then, inspect coefficients/cut-offs at multiples of the period s , which imply seasonal dependency and determine P and Q .
- 4) Fit the model using procedure `arima()`. In contrast to *ARIMA* fitting, this is now exclusively done on the original series, with setting the two arguments `order=c(p,d,q)` and `seasonal=c(P,D,Q)` accordingly.
- 5) Check the accuracy of the fitted model by residual analysis. These must look like White Noise. If thus far, there is ambiguity in the model order, AIC analysis can serve to come to a final decision.

Next, we turn our attention to series that have neither trend nor seasonality, but show serial dependence in the conditional variance.

6.3 ARCH/GARCH Models

In this chapter, we consider the SMI log-returns that were already presented in section 1.2.4. By closer inspection of the time series plot, we observe some long-tailedness, and also, the series exhibits periods of increased variability, which is usually termed volatility in the (financial) literature. We had previously observed series with non-constant variation, such as the oil prices and beer production in the previous sections. Such series, where the variance increases with increasing level of the series, are called *heteroskedastic*, and can often be stabilized using a log-transformation.

However, that matter is different with the SMI log-returns: here, there are periods of increased volatility, and thus the conditional variance of the series is serially correlated, a phenomenon that is called *conditional heteroskedasticity*. This is not a violation of the stationarity assumption, but some special treatment for this type of series is required. Furthermore, the ACF of such series typically does not differ significantly from White Noise. Still, the data are not *iid*, which can be shown with the ACF of the squared observations. With the plots on the next page, we illustrate the presence of these stylized facts for the SMI log-returns:



6.3.1 The ARCH and GARCH Models

In order to account for volatility, we require a model that reflects the dependency in the conditional variance. We operate under the assumption that:

$$X_t = \mu_t + E_t,$$

where the disturbance term E_t can be rewritten as $\sigma_t W_t$: $X_t = \mu_t + \sigma_t W_t$. Here, W_t is a White Noise innovation and $\sigma_t = \text{Var}(X_t | X_{t-1}, X_{t-2}, \dots)$ is the conditional variance that is assumed to be non-constant. Finally $\mu_t = E[X_t | X_{t-1}, X_{t-2}, \dots]$ is the conditional expectation as before. It is perfectly allowed to have both dependence in the conditional mean and variance, and hence a mixture of ARMA and GARCH processes. However, for simplicity we assume throughout this section that both the conditional and the global mean are zero: $\mu = \mu_t = 0$ and thus $X_t = \sigma_t W_t$.

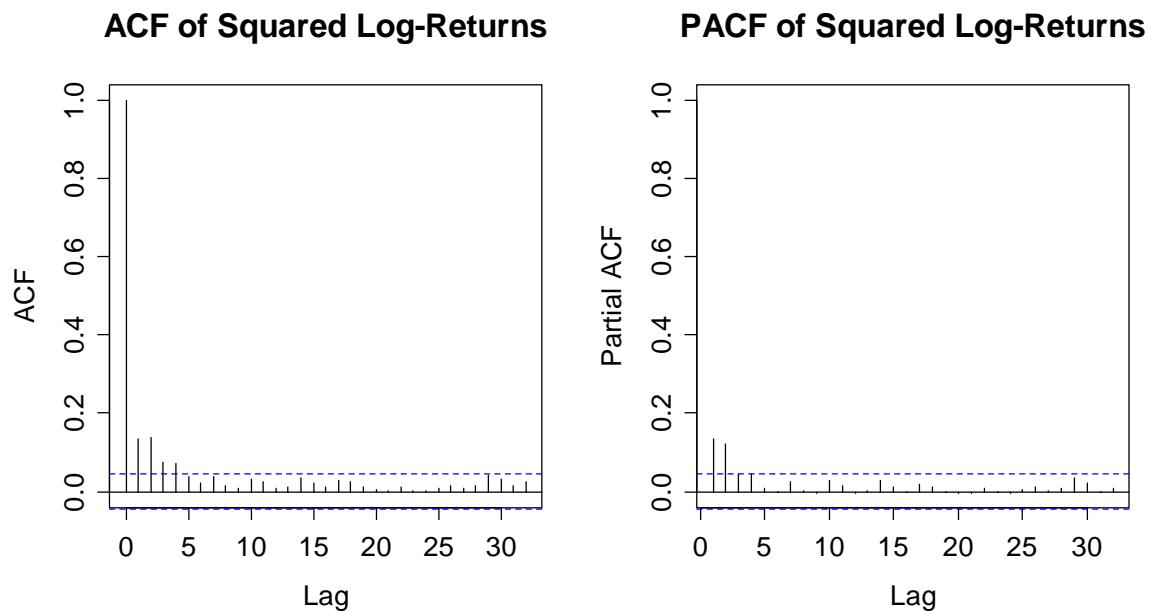
The most simple and intuitive way of doing this is to use an autoregressive model for the variance process. Thus, a series E_t is first-order autoregressive conditional heteroskedastic, denoted as $ARCH(1)$, if:

$$E_t = W_t \sqrt{\alpha_0 + \alpha_1 E_{t-1}^2}.$$

Here, W_t is a White Noise process with mean zero and unit variance. The two parameters α_0, α_1 are the model coefficients. An $ARCH(1)$ process shows volatility, as can easily be derived:

$$\begin{aligned} Var(E_t) &= E[E_t^2] \\ &= E[W_t^2]E[\alpha_0 + \alpha_1 E_{t-1}^2] \\ &= E[\alpha_0 + \alpha_1 E_{t-1}^2] \\ &= \alpha_0 + \alpha_1 \cdot Var(E_{t-1}) \end{aligned}$$

Note that this derivation is based on $E[W_t^2] = 1$ and $E[E_t] = E[W_t] = 0$. As we had aimed for, the variance of an $ARCH(1)$ process behaves just like an $AR(1)$ model. Hence, the decay in the autocorrelations of the squared residuals should indicate whether an $ARCH(1)$ is appropriate or not.



In our case, the analysis of ACF and PACF of the squared log-returns suggests that the variance may be well described by an $AR(2)$ process. This is not what we had discussed, but an extension exists. An $ARCH(p)$ process is defined by:

$$E_t = W_t \sqrt{\alpha_0 + \sum_{i=1}^p \alpha_i E_{t-i}^2}$$

Fitting in **R** can be done using procedure `garch()`. This is a more flexible tool, which also allows for fitting GARCH processes, as discussed below. The command in our case is as follows:

```
> fit <- garch(lret.smi, order = c(0, 2), trace=FALSE); fit
Call: garch(x = lret.smi, order = c(0, 2), trace = FALSE)
Coefficient(s):
      a0          a1          a2
6.568e-05 1.309e-01 1.074e-01
```

For verifying appropriate fit of the $ARCH(2)$, we need to check the residuals of the fitted model. This includes inspecting ACF and PACF for both the “normal” and the squared residuals. We here do without showing plots, but the $ARCH(2)$ is OK.

A nearby question is whether we can also use an $ARMA(p,q)$ process for describing the dependence in the variance of the process. The answer is yes. This is what a $GARCH(p,q)$ model does. A series $E_t = W_t \sqrt{H_t}$ is $GARCH(p,q)$ if:

$$H_t = \alpha_0 + \sum_{i=1}^q \alpha_i E_{t-i}^2 + \sum_{j=1}^p \beta_j H_{t-j}$$

6.3.2 Use of GARCH Models

GARCH models are useless for the prediction of the level of a series, i.e. for the SMI log-returns, they do not provide any idea whether the stocks’ value will increase or decrease on the next day. However, they allow for a more precise understanding in the (up or down) changes that might be expected during the next day(s). This allows stockholders to adjust their position, so that they do not take any unduly risks.

7 Time Series Regression

7.1 What Is the Problem?

It is often the case that we aim for describing some time series Y_t with a linear combination of some explanatory series x_{t1}, \dots, x_{tp} . As we will see below, the predictors can either be true covariates, or terms that are derived from time, as for example linear trends or seasonal effects. We employ the universally known linear model for linking the response series with the predictors:

$$Y_t = \beta_0 + \beta_1 x_{t1} + \dots + \beta_p x_{tp} + E_t$$

The regression coefficients β_1, \dots, β_p are usually estimated with the least squares algorithm, for which an error term with zero expectation, constant variation and no correlation is assumed. However, if response and predictors are time series with autocorrelation, the last condition often turns out to be violated, though this is not always the case.

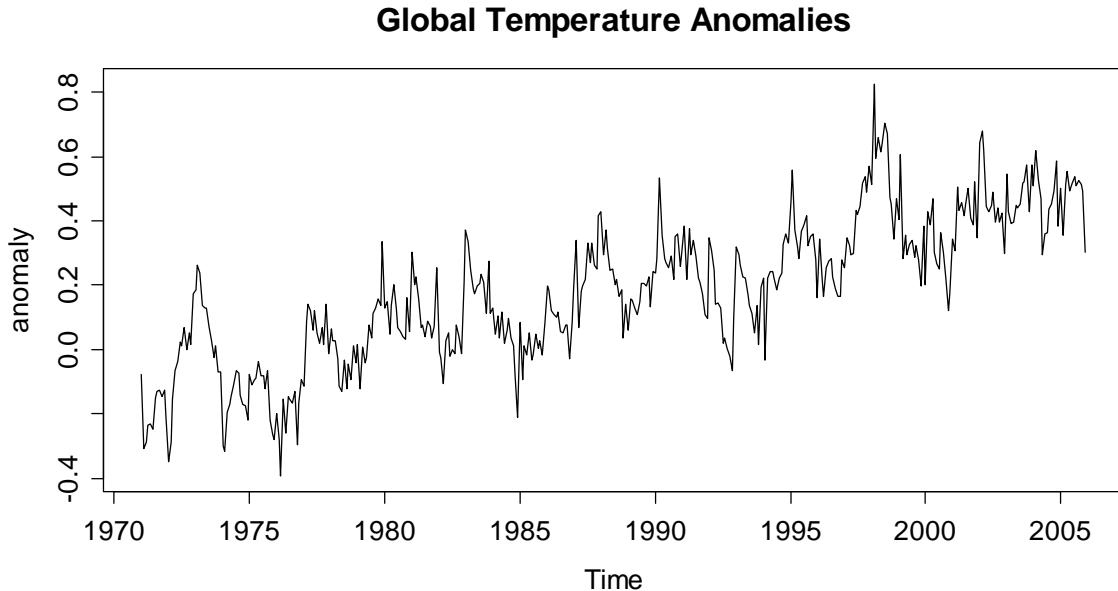
Now, if we are facing a (time series) regression problem *with* correlated errors, the estimates $\hat{\beta}_j$ will remain being unbiased, but the least squares algorithm is no longer efficient. Or in other words: more precisely working estimators exist. Even more problematic are the standard errors of the regression coefficients $\hat{\beta}_j$: they are often grossly wrong in case of correlated errors. As they are routinely underestimated, inference on the predictors often yields spurious significance, i.e. one is prone to false conclusions from the analysis.

Thus, there is a need for more general linear regression procedures that can deal with serially correlated errors, and fortunately, they exist. We will here discuss the simple, iterative Cochrane-Orcutt procedure, and the Generalized Least Squares method, which marks a theoretically sound approach to regression with correlated errors. But first, we present some time series regression problems to illustrating what we are dealing with.

Example 1: Global Temperature

In climate change studies time series with global temperature values are analyzed. The scale of measurement is anomalies, i.e. the difference between the monthly global mean temperature versus the overall mean between 1961 and 1990. The data can be obtained at <http://www.cru.uea.ac.uk/cru/data>. For illustrative purposes, we here restrict to a period from 1971 to 2005 which corresponds to a series of 420 records. For a time series plot, see the next page.

```
> ## Time Series Plot
> my.temp <- window(global, c(1971,1), c(2005,12))
> plot(my.temp, ylab="anomaly")
> title("Global Temperature Anomalies")
```



There is a clear trend which seems to be linear. Despite being monthly measured, the data show no evident seasonality. This is not overly surprising, since we are considering a global mean, i.e. the season should not make for a big difference. But on the other hand, because the landmass is not uniformly distributed over both halves of the globe, it could still be present. It is natural to try a season-trend-decomposition for this series. We will employ a parametric model featuring a linear trend plus a seasonal factor.

$$Y_t = \beta_0 + \beta_1 \cdot t + \beta_2 \cdot 1_{[month="Feb"]} + \dots + \beta_{12} \cdot 1_{[month="Dec"]} + E_t,$$

where $t = 1, \dots, 420$ and $1_{[month="Feb"]}$ is a dummy variable that takes the value 1 if an observation is from month February, and zero else. Clearly, this is a time series regression model. The response Y_t is the global temperature anomalies, and even the predictors, i.e. the time and the dummies, can be seen as time series, even if simple ones.

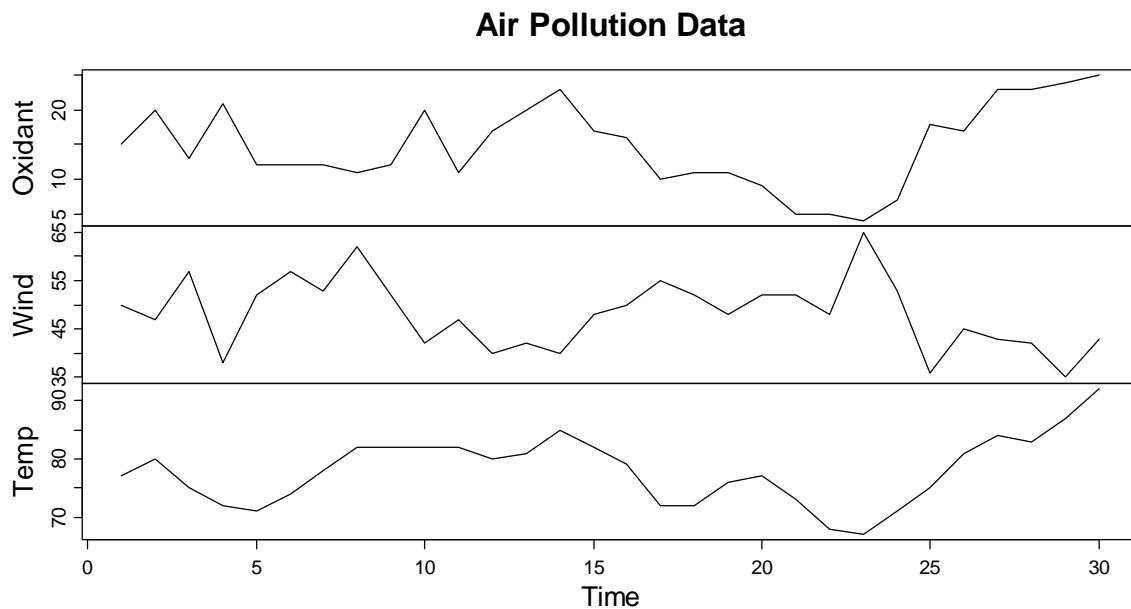
As we have seen previously, the goal with such parametric decomposition models is to obtain a stationary remainder term E_t . But stationary does not necessarily mean White Noise, and in practice it often turns out that E_t shows some serial correlation. Thus, if the regression coefficients are obtained from the least squares algorithm, we apparently feature some violated assumption.

This violation can be problematic, even in an applied setting: a question of utter importance with the above series is whether trend and seasonal effect are significantly present. It would be nice to answer such questions using the inference approaches (tests and confidence intervals) that linear regression provides. However, for obtaining reliable inference results, we need to account for the correlation among the errors. We will show this below, after introducing some more examples and theory.

Example 2: Air Pollution

In this second example, we consider a time series that is stationary, and where the regression aims at understanding the series, rather than decomposing it into some deterministic and random components. We examine the dependence of a photochemical pollutant (morning maximal value) on the two meteorological variables wind and temperature. The series, which constitute of 30 observations taken on consecutive days, come from the Los Angeles basin. They are not publicly available, but can be obtained from the lecturer upon request.

```
> ## Importing the data
> tmp <- read.table("pollute.dat", header=TRUE)
> dat <- ts.union(Oxidant=ts(tmp$Oxidant), Wind=ts(tmp$Wind),
+                   Temp=ts(tmp$Temp))
> ## Visualizing the data
> plot(dat, main="Air Pollution Data")
```



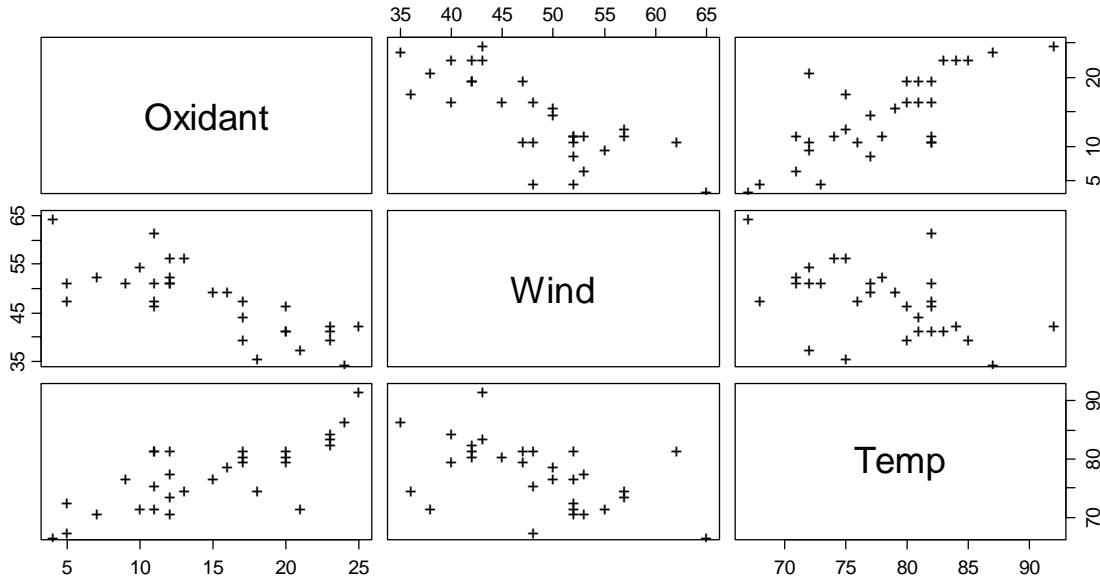
There is no counterevidence to stationarity for all three series. What could be the goal here? Well, we aim for enhancing the understanding of how the pollution depends on the meteorology, i.e. what influence wind and temperature have on the oxidant values. We can naturally formulate the relation with a linear regression model:

$$Y_t = \beta_0 + \beta_1 x_{t1} + \beta_2 x_{t2} + E_t .$$

In this model, the response Y_t is the oxidant, and as predictors we have x_{t1} , wind, and x_{t2} , the temperature. For the index, we have $t = 1, \dots, 30$, and obviously, this is a time series regression model.

For gaining some more insight with these data, it is also instructive to visualize the data using a pairs plot, as shown on the next page. There, a strong, positive linear

association is recognizable between pollutant and the temperature. In contrast, there is a negative linear relation between pollutant and wind. Lastly, between the predictors wind and temperature, there is not much of a correlation. This data structure is not surprising because wind causes a stronger movement of the air and thus the pollutant is "better" distributed. Also, the wind causes some cooling.



For achieving our practical goals with this dataset, we require precise and unbiased estimates of the regression coefficients β_1 and β_2 . Moreover, we might like to give some statements about the significance of the predictors, and thus, we require some sound standard errors for the estimates. However, also with these data, it is well conceivable that the error term E_t will be serially correlated. Thus again, we will require some procedure that can account for this.

Time Series Regression Model

The two examples have shown that time series regression models do appear when decomposing series, but can also be important when we try to understand the relation between response and predictors with measurements that were taken sequentially. Generally, with the model

$$Y_t = \beta_0 + \beta_1 x_{t1} + \dots + \beta_p x_{tp} + E_t$$

we assume that the influence of the series x_{t1}, \dots, x_{tp} on the response Y_t is simultaneous. Nevertheless, lagged variables are also allowed, i.e. we can also use terms such as $x_{(t-k);j}$ with $k > 0$ as predictors. While this generalization can be easily built into our model, one quickly obtains models with many unknown parameters. Thus, when exploring the dependence of a response series to lags of some predictor series, there are better approaches than regression. In particular, this is the cross correlations and the transfer function model, which will be exhibited in later chapters of this scriptum.

In fact, there are not many restrictions for the time series regression model. As we have seen, it is perfectly valid to have non-stationary series as either the response or as predictors. However, it is crucial that there is no feedback from Y_t to the x_{ij} . Additionally, the error E_t must be independent of the explanatory variables, but it may exhibit serial correlation.

7.2 Finding Correlated Errors

When dealing with a time series regression problem, we first always assume uncorrelated errors and start out with an ordinary least squares regression. Based on its residuals, the assumption can be verified, and if necessary, action can be taken. For identifying correlation among the residuals, we analyze their time series plot, ACF and PACF.

Example 1: Global Temperature

Our goal is the decomposition of the global temperature series into a linear trend plus some seasonal factor. First and foremost, we prepare the data:

```
> num.temp <- as.numeric(my.temp)
> num.time <- as.numeric(time(my.temp))
> mn01      <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun")
> mn02      <- c("Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
> month     <- factor(cycle(my.temp), labels=c(mn01, mn02))
> dat       <- data.frame(temp=num.temp, time=num.time, month)
```

The regression model is the estimated with R's function `lm()`. The summary function returns estimates, standard errors plus the results from some hypothesis tests. It is important to notice that all of these results are in question should the errors turn out to be correlated.

```
> fit.lm <- lm(temp ~ time + season, data=dat)
> summary(fit.lm)
```

Call:

```
lm(formula = temp ~ time + season, data = dat)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.36554	-0.07972	-0.00235	0.07497	0.43348

Coefficients:

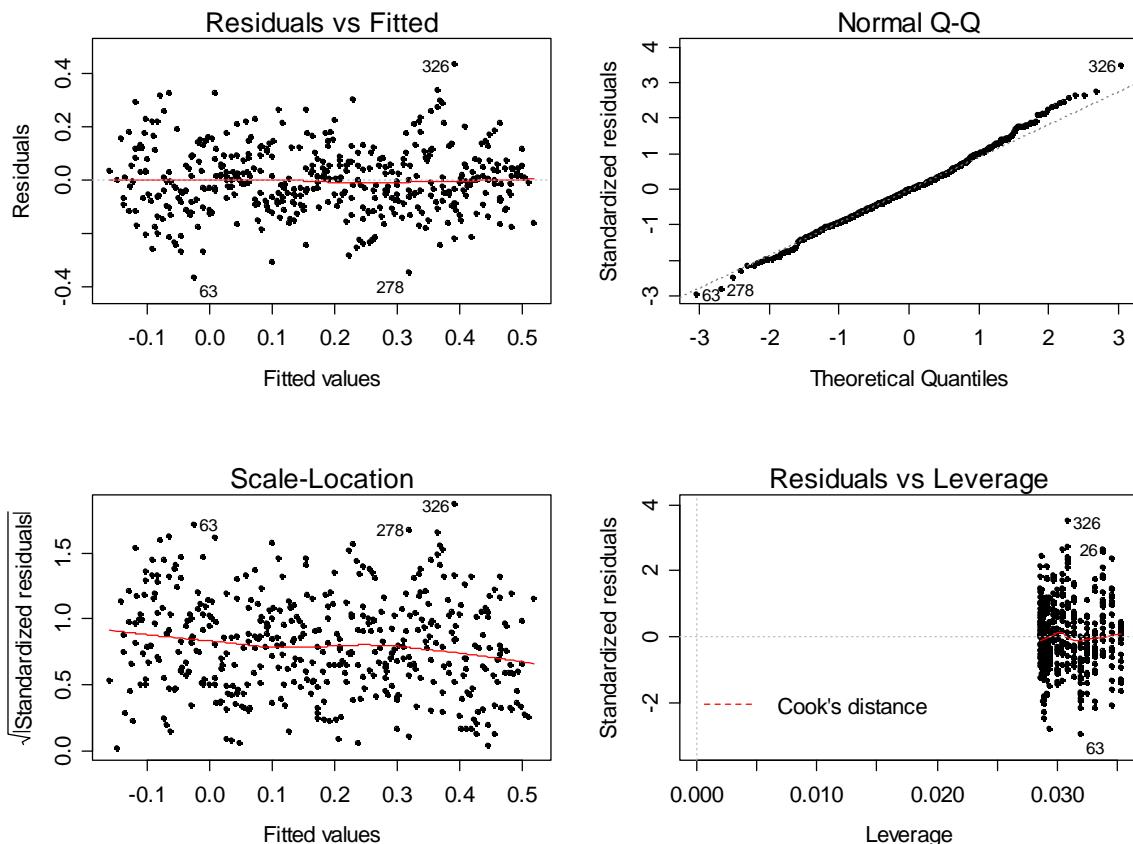
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-3.603e+01	1.211e+00	-29.757	<2e-16 ***
time	1.822e-02	6.089e-04	29.927	<2e-16 ***
seasonFeb	6.539e-03	3.013e-02	0.217	0.8283
seasonMar	-1.004e-02	3.013e-02	-0.333	0.7392
seasonApr	-1.473e-02	3.013e-02	-0.489	0.6252
seasonMay	-3.433e-02	3.013e-02	-1.139	0.2552

seasonJun	-2.628e-02	3.013e-02	-0.872	0.3836
seasonJul	-2.663e-02	3.013e-02	-0.884	0.3774
seasonAug	-2.409e-02	3.013e-02	-0.799	0.4245
seasonSep	-3.883e-02	3.013e-02	-1.289	0.1982
seasonOct	-5.212e-02	3.013e-02	-1.730	0.0844 .
seasonNov	-6.633e-02	3.013e-02	-2.201	0.0283 *
seasonDec	-4.485e-02	3.013e-02	-1.488	0.1374

Residual standard error: 0.126 on 407 degrees of freedom
 Multiple R-squared: 0.6891, Adjusted R-squared: 0.68
 F-statistic: 75.18 on 12 and 407 DF, p-value: < 2.2e-16

As the next step, we need to perform some residual diagnostics. The `plot()` function, applied to a regression fit, serves as a check for zero expectation, constant variation and normality of the errors, and can give hints on potentially problematic leverage points.

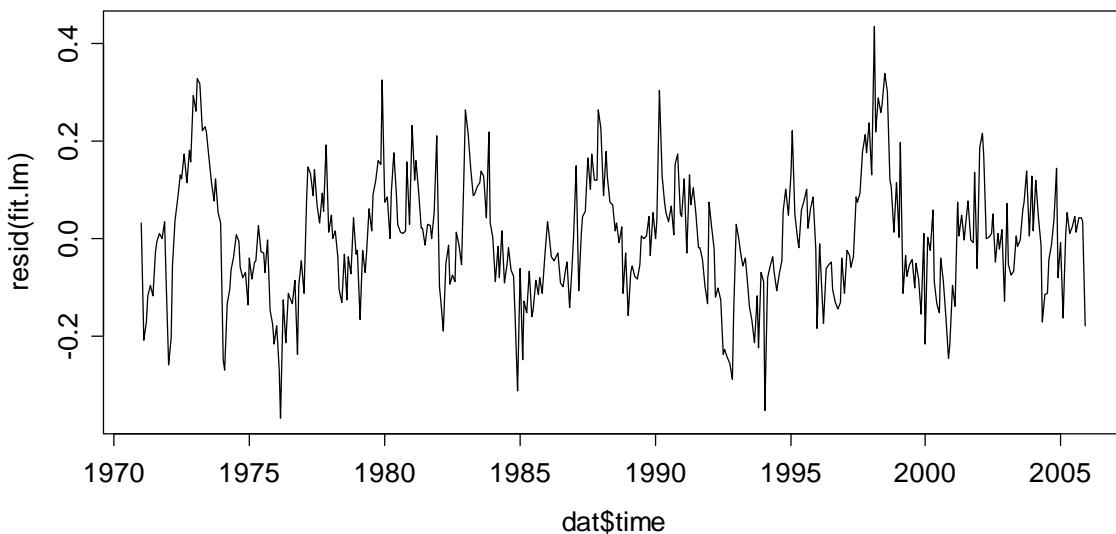
```
> par(mfrow=c(2, 2))
> plot(fit.lm, pch=20)
```



Except for some very slightly long tailed errors, which do not require any action, the residual plots look fine. What has not yet been verified is whether there is any serial correlation among the residuals. If we wish to see a time series plot, the following commands are useful:

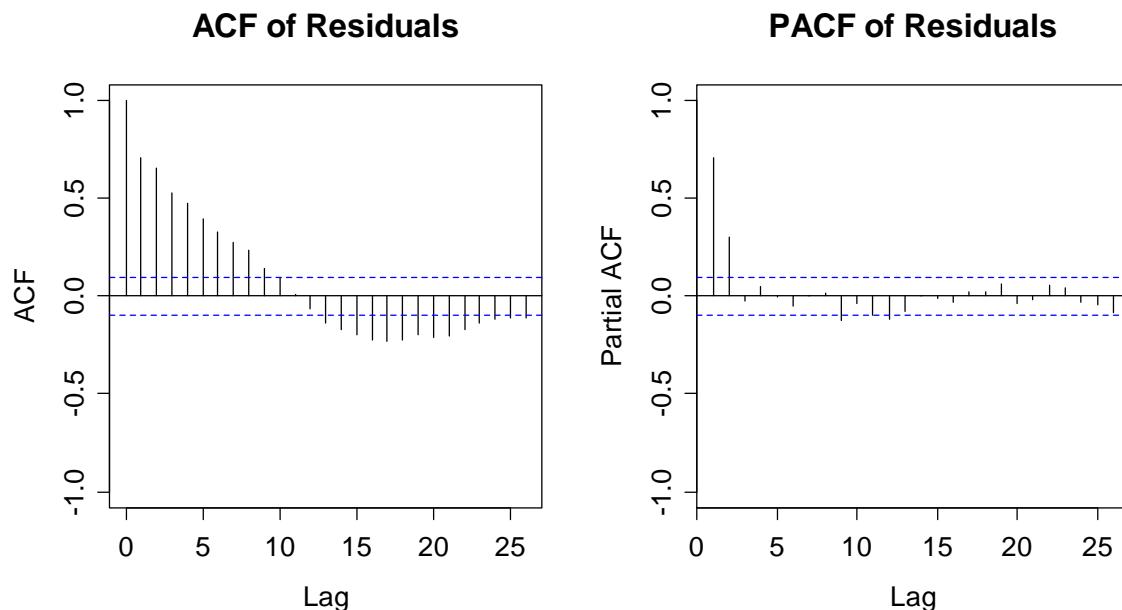
```
> plot(dat$time, resid(fit.lm), type="l")
```

Residuals of the lm() Function



It is fairly obvious from the time series plot that the residuals are correlated. Our main tool for describing the dependency structure is the ACF and PACF plots, however. These are as follows:

```
> par(mfrow=c(1, 2))
> acf(resid(fit.lm), main="ACF of Residuals")
> pacf(resid(fit.lm), main="PACF of Residuals")
```



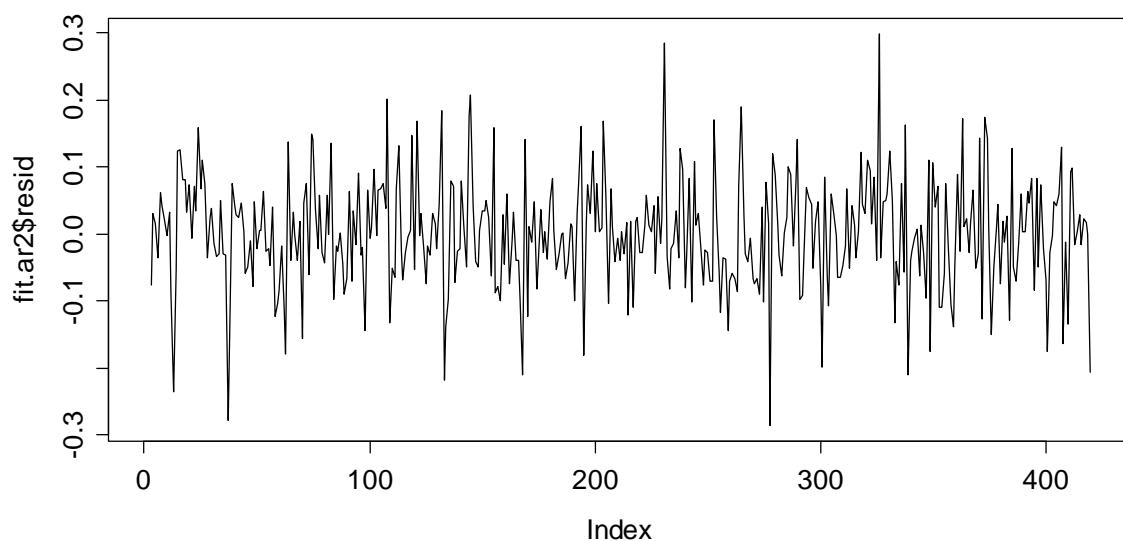
The ACF shows a rather slow exponential decay, whereas the PACF shows a clear cut-off at lag 2. With these stylized facts, it might well be that an AR(2)

model is a good description for the dependency among the residuals. We verify this:

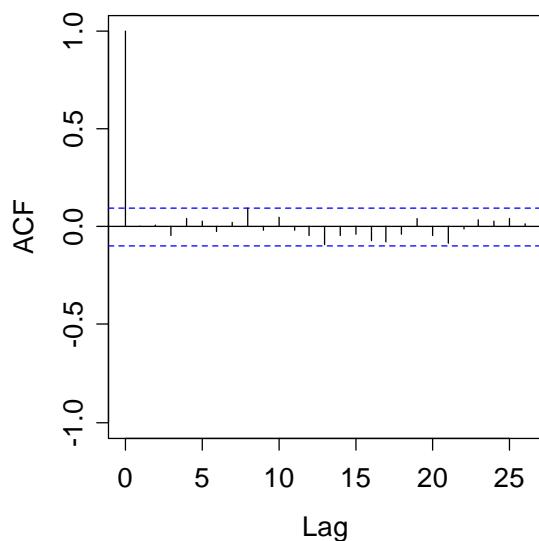
```
> fit.ar2 <- ar.burg(resid(fit.lm)); fit.ar2
Call: ar.burg.default(x = resid(fit.lm))
Coefficients:
 1      2
0.4945 0.3036
Order selected 2  sigma^2 estimated as  0.00693
```

When using Burg's algorithm for parameter estimation and doing model selection by AIC, order 2 also turns out to be optimal. For verifying an adequate fit, we visualize the residuals from the AR(2) model. These need to look like White Noise.

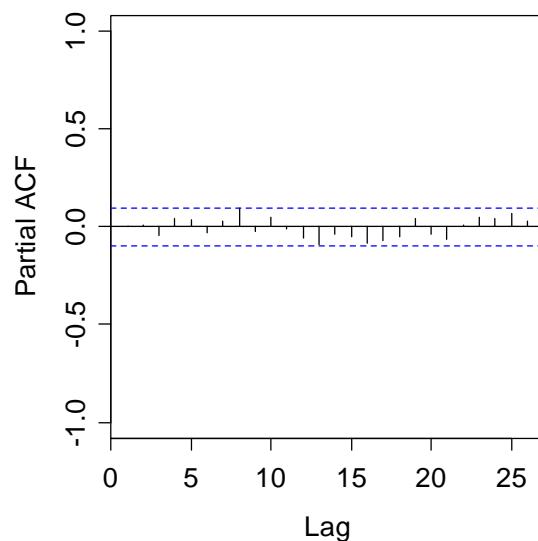
Residuals of AR(2)



ACF of AR(2) Residuals



ACF of AR(2) Residuals



There is no contradiction to the White Noise hypothesis for the residuals from the AR(2) model. Thus, we can summarize as follows: the regression model that was used for decomposing the global temperature series into a linear trend and a seasonal factor exhibit correlated errors that seem to originate from an AR(2) model. Theory tells us that the point estimates for the regression coefficients are still unbiased, but they are no longer efficient. Moreover, the standard errors for these coefficients can be grossly wrong. Thus, we need to be careful with the regression summary approach that was displayed above. And since our goal is inferring significance of trend and seasonality, we need to come up with some better suited method.

Example 2: Air Pollution

Now, we are dealing with the air pollution data. Again, we begin our regression analysis using the standard assumption of uncorrelated errors. Thus, we start out by applying the `lm()` function and printing the `summary()`.

```
> fit.lm <- lm(Oxidant ~ Wind + Temp, data=dat)
> summary(fit.lm)

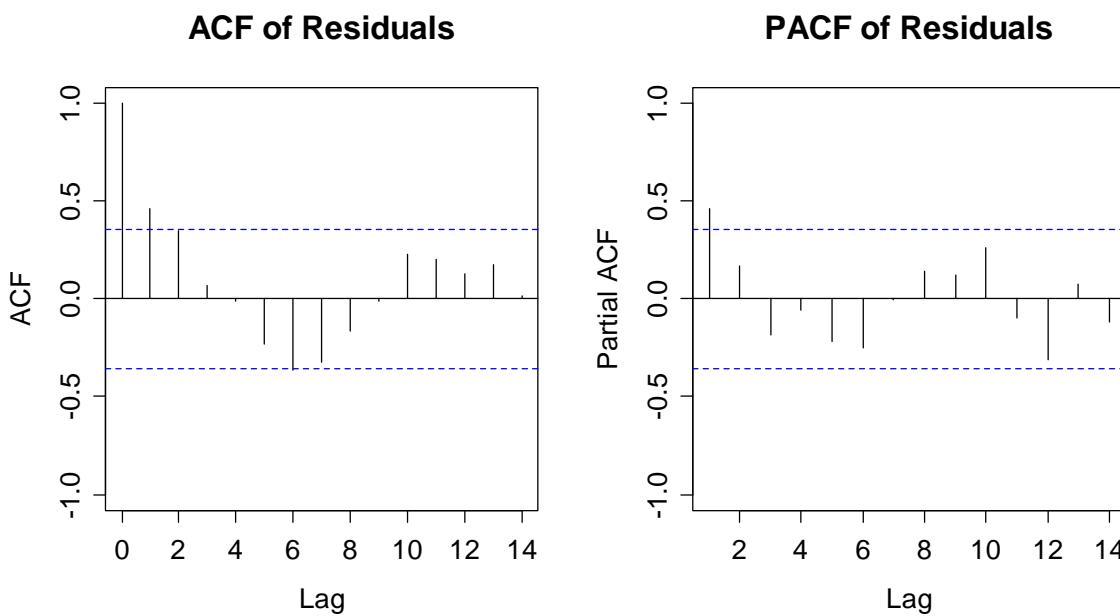
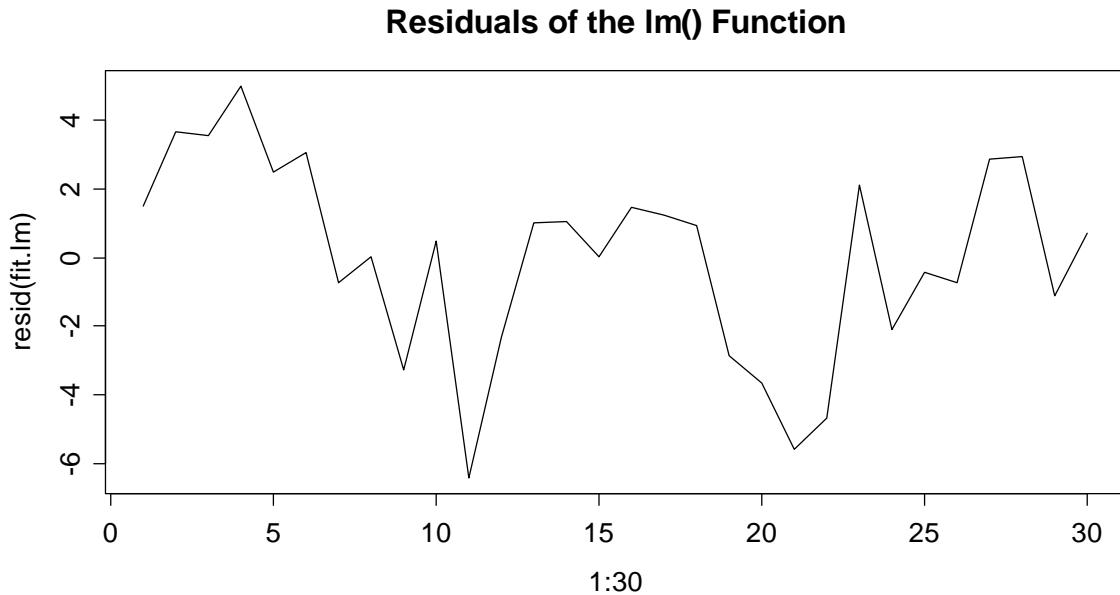
Call:
lm(formula = Oxidant ~ Wind + Temp, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.3939 -1.8608  0.5826  1.9461  4.9661 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -5.20334   11.11810  -0.468   0.644    
Wind        -0.42706    0.08645  -4.940 3.58e-05 ***  
Temp         0.52035    0.10813   4.812 5.05e-05 ***  
---
Residual standard error: 2.95 on 27 degrees of freedom
Multiple R-squared:  0.7773 , Adjusted R-squared:  0.7608 
F-statistic: 47.12 on 2 and 27 DF,  p-value: 1.563e-09
```

We will do without showing the 4 standard diagnostic plots, and here only report that they do not show any model violations. Because we are performing a time series regression, we also need to check for potential serial correlation of the errors. As before, this is done on the basis of time series plot, ACF and PACF:

```
> plot(1:30, resid(fit.lm), type="l")
> title("Residuals of the lm() Function")
> par(mfrow=c(1,2))
> acf(resid(fit.lm), ylim=c(-1,1), main="ACF of Residuals")
> pacf(resid(fit.lm), ylim=c(-1,1), main="PACF of Residuals")
```



Also in this example, the time series of the residuals exhibits serial dependence. Because the ACF shows an exponential decay and the PACF cuts off at lag 1, we hypothesize that an $AR(1)$ model is a good description. While the AIC criterion suggests an order of $p = 14$, the residuals of an $AR(1)$ show the behavior of White Noise. Additionally, using an $AR(14)$ would be spending too many degrees of freedom for a series with only 30 observations.

Thus, we can summarize that also in our second example with the air pollution data, we feature a time series regression that has correlated errors. Again, we must not communicate the above regression summary and for sound inference, we require more sophisticated models.

7.2.1 Durbin-Watson Test

For the less proficient user, hypothesis tests always seem like an attractive alternative to visual inspection of graphical output. This is certainly also the case when the task is identifying a potential serial correlation in a regression analysis. Indeed, there is a formal test that addresses the issue, called the Durbin-Watson test. While we will here briefly go into it, we do not recommend it for practical application. The Durbin-Watson test tests the null hypothesis $H_0: \rho(1)=0$ against the alternative $H_A: \rho(1) \neq 0$. The test statistic \hat{D} is calculated as follows

$$\hat{D} = \frac{\sum_{t=2}^n (r_t - r_{t-1})^2}{\sum_{t=1}^n r_t^2}$$

where $r_t = y_t - \hat{y}_t$ is the residual from the regression model, observed at time t . There is an approximate relationship between the test statistic \hat{D} and the autocorrelation coefficient at lag 1:

$$\hat{D} \approx 2(1 - \hat{\rho}(1))$$

The test statistic takes values between 0 if $r_t = r_{t-1}$ and 4 if $r_t = -r_{t-1}$. These extremes indicate perfect correlation of the residuals. Values around 2, on the other hand, are evidence for uncorrelated errors. The exact distribution of \hat{D} is rather difficult to derive. However, we do not need to bother with this. The R package `lmtest` holds an implementation of the Durbin-Watson test with function `dwtest()`, where the p-value is either (for large sample size) determined by a normal approximation, or (for short series) by an iterative procedure.

Example 1: Global Temperature

```
> dwtest(fit.lm)
data: fit.lm
DW = 0.5785, p-value < 2.2e-16
alt. hypothesis: true autocorrelation is greater than 0
```

Example 2: Air Pollution

```
> dwtest(fit.lm)
data: fit.lm
DW = 1.0619, p-value = 0.001675
alt. hypothesis: true autocorrelation is greater than 0
```

Thus, the null hypothesis is rejected in both examples and we come to the same conclusion (“errors are correlated”) as with our visual inspection. It is very important to note that this is not necessary: In cases where the errors follow an $AR(p)$ process where $p > 1$ and $|\rho(1)|$ is small, the null hypothesis will not be rejected despite the fact that the errors are correlated.

7.3 Cochrane-Orcutt Method

The goal of this section is to solve the time series regression problem with errors that have an $AR(1)$ structure. This simple case is illustrative and helps to build the comprehension for more complicated error dependencies. We consider the Air Pollution example, where we have:

$$Y_t = \beta_0 + \beta_1 x_{t1} + \beta_2 x_{t2} + E_t \text{ with } E_t = \alpha E_{t-1} + U_t, \text{ where } U_t \sim N(0, \sigma_U^2) \text{ iid.}$$

The fundamental trick, on which in fact all time series regression methods are based will be presented here and now. We make the transformation:

$$Y'_t = Y_t - \alpha Y_{t-1}$$

Next, we plug-in the model equation and rearrange the terms. Finally, we build on the fundamental property that $E_t - \alpha E_{t-1} = U_t$. The result is:

$$\begin{aligned} Y'_t &= \beta_0 + \beta_1 x_{t1} + \beta_2 x_{t2} + E_t - \alpha(\beta_0 + \beta_1 x_{t-1,1} + \beta_2 x_{t-1,2} + E_{t-1}) \\ &= \beta_0(1 - \alpha) + \beta_1(x_{t1} - x_{t-1,1}) + \beta_2(x_{t2} - x_{t-1,2}) + E_t - \alpha E_{t-1} \\ &= \beta'_0 + \beta'_1 x'_{t1} + \beta'_2 x'_{t2} + U_t \end{aligned}$$

Obviously, this is a time series regression problem where the error term U_t is *iid*. Also note that both the response and the predictors have undergone a transformation. The coefficients however, are identical in both the original and the modified regression equation. For implementing this approach in practice, we require knowledge about the $AR(1)$ parameter α . Usually, it is not known previously. A simple idea to overcome this and solve the time series regression problem for the Air Pollution data is as follows:

- 1) Run OLS regression to obtain estimates $\hat{\beta}_0, \dots, \hat{\beta}_p$
- 2) Estimate an $AR(1)$ on the OLS residuals to obtain $\hat{\alpha}$
- 3) Determine the prime variables $Y'; x'$ and derive $\hat{\beta}'_0, \hat{\beta}'_1, \dots, \hat{\beta}'_p$ by OLS

This procedure is known as the *Cochrane-Orcutt* iterative method. Please note that the estimates $\hat{\beta}'_0, \hat{\beta}'_1, \dots, \hat{\beta}'_p$ and their standard errors from the OLS regression in step 3) are sound and valid. But while the Cochrane-Orcutt procedure has its historical importance and is very nice for illustration, it lacks of a direct **R** implementation, and, as an iterative procedure, also of mathematical closeness and quality. The obvious improvement is to solve the prime regression problem by simultaneous Maximum-Likelihood estimation of all parameters:

$$\beta_0, \dots, \beta_p; \alpha; \sigma_U^2$$

This is possible and implemented in the **R** function `gls()`. Also, we need to be able to handle more complicated structure for the regression error E_t . For this, we resort to matrix notation, see the next section.

7.4 Generalized Least Squares

The ordinary least squares regression model assumes that $\text{Var}(E) = \sigma^2 I$, i.e. the covariance matrix of the errors is diagonal with identical values on the diagonal itself. As we have seen in our examples above, this is not a good model for time series regression. There, we rather have $\text{Var}(E) = \sigma^2 \Sigma$, where Σ reports the correlation among the errors. Using a Cholesky decomposition, we can write $\Sigma = SS^T$, where S is a triangular matrix. This allows us to rewrite the regression model in matrix notation as follows:

$$\begin{aligned} y &= X\beta + E \\ S^{-1}y &= S^{-1}X\beta + S^{-1}E \\ y' &= X'\beta + E' \end{aligned}$$

This transformation is successful, because in the prime model, we have uncorrelated errors again:

$$\text{Var}(E) = \text{Var}(S^{-1}E) = S^{-1}\text{Var}(E)S^{-T} = S^{-1}\sigma^2 SS^T S^{-T} = \sigma^2 I$$

With some algebra, it is easy to show that the estimated regression coefficients for the generalized least squares approach turn out to be:

$$\hat{\beta} = (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} y$$

This is what is known as the *generalized least squares estimator*. Moreover, the covariance matrix of the coefficient vector is:

$$\text{Var}(\hat{\beta}) = (X^T \Sigma^{-1} X)^{-1} \sigma^2$$

This covariance matrix then also contains standard errors in which the correlation of the errors has been accounted for, and with which sound inference is possible. However, while this all neatly lines up, we of course require knowledge about the error covariance matrix Σ , which is generally unknown in practice. What we can do is estimate it from the data, for which two approaches exist.

Cochrane-Orcutt Method

As we have seen above, this method is iterative: it starts with an ordinary least squares (OLS) regression, from which the residuals are determined. For these residuals, we can then fit an appropriate $ARMA(p, q)$ model and with its estimated model coefficients $\alpha_1, \dots, \alpha_p$ and $\beta_1^{MA(q)}, \dots, \beta_q^{MA(q)}$. On the basis of the estimated $AR(MA)$ model coefficients, an estimate of the error covariance matrix Σ can be derived. We denote it by $\hat{\Sigma}$, and plug it into the formulae presented above. This yields adjusted regression coefficients and correct standard errors for these regression problems. As mentioned above, the iterative approach is secondary to a simultaneous MLE. Thus, we do without further performing Cochrane-Orcutt on our examples.

The `gls()` Procedure

A better, yet more sophisticated approach is to estimate the regression coefficients and the *ARMA* parameters simultaneously. This can be done using the Maximum-Likelihood principle. Even under the assumption of Gaussian errors, this is a nonlinear and numerically difficult problem. However, for practical application, we do not need to worry. The **R** package `nlme` features the `gls()` procedure which tackles this problem. Thus, we focus on correct application of the **R** function.

Example 1: Global Temperature

Every GLS regression analysis starts by fitting an OLS and analyzing the residuals, as we have done above. Remember that the only model violation we found were correlated residuals that were well described with an *AR*(2) model. Please note that for performing GLS, we need to provide a dependency structure for the errors. Here, this is the *AR*(2) model, in general, it is an appropriate *ARMA*(p, q). The syntax and output is as follows:

```
> library(nlme)
> corStruct <- corARMA(form=~time, p=2)
> fit.gls <- gls(temp~time+season, data=dat, corr=corStruct)
> fit.gls
Generalized least squares fit by REML
  Model: temp ~ time + season
  Data: dat
  Log-restricted-likelihood: 366.3946

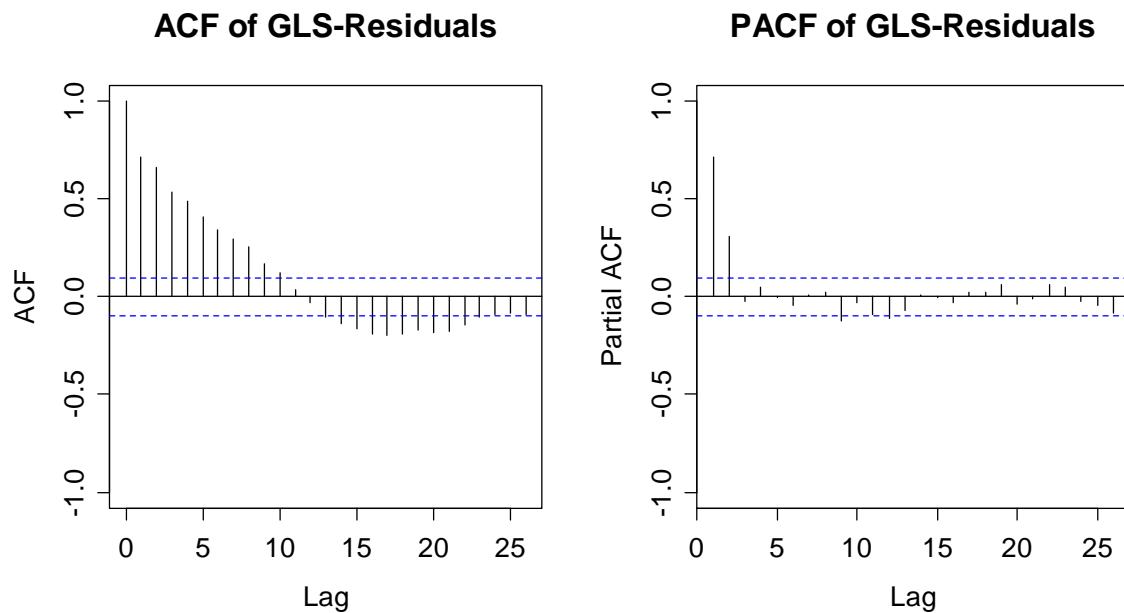
Coefficients:
              (Intercept)          time        seasonFeb        seasonMar
-39.896981987   0.020175528   0.008313205   -0.006487876
               seasonApr        seasonMay        seasonJun        seasonJul
-0.009403242   -0.027232895   -0.017405404   -0.015977913
               seasonAug        seasonSep        seasonOct        seasonNov
-0.011664708   -0.024637218   -0.036152584   -0.048582236
               seasonDec
-0.025326174

Correlation Structure: ARMA(2,0)
  Formula: ~time
  Parameter estimate(s):
    Phi1      Phi2
  0.5539900 -0.1508046
Degrees of freedom: 420 total; 407 residual
Residual standard error: 0.09257678
```

The result reports the regression and the *AR* coefficients. Using the `summary()` function, even more output with all the standard errors can be generated. We omit this here and instead focus on the necessary residual analysis for the GLS model. We can extract the residuals using the usual `resid()` command. Important: these

residuals must not look like White Noise, but as from an $ARMA(p,q)$ with orders p and q as provided in the `corStruct` object – which in our case, is an $AR(2)$.

```
> par(mfrow=c(1, 2))
> acf(resid(fit.gls), main="ACF of GLS-Residuals")
> pacf(resid(fit.gls), main="PACF of GLS-Residuals")
```



The plots look similar to the ACF/PACF plots of the OLS residuals. This is often the case in practice, only for more complex situations, there can be a bigger discrepancy. And because we observe an exponential decay in the ACF, and a clear cut-off at lag 2 in the PACF, we conjecture that the GLS residuals meet the properties of the correlation structure we hypothesized, i.e. of an $AR(2)$ model. Thus, we can now use the GLS fit for drawing inference. We first compare the OLS and GLS point estimate for the trend, along with its confidence interval:

```
> coef(fit.lm)[ "time" ]
      time
0.01822374
> confint(fit.lm, "time" )
      2.5 %   97.5 %
time 0.01702668 0.0194208
> coef(fit.gls)[ "time" ]
      time
0.02017553
> confint(fit.gls, "time" )
      2.5 %   97.5 %
time 0.01562994 0.02472112
```

We obtain a temperature increase of 0.0182 degrees per year with the OLS, and of 0.0202 with the GLS. While this may seem academic, the difference among the point estimates is around 10%, and theory tells us that the GLS result is more reliable. Moreover, the length of the confidence interval is 0.0024 with the OLS,

and 0.0091 and thus 3.5 times as large with the GLS. Thus, without accounting for the dependency among the errors, the precision of the trend estimate is by far overestimated. Nevertheless, also the confidence interval obtained from GLS regression does not contain the value 0, and thus, the null hypothesis on no global warming trend is rejected (with margin!).

Finally, we investigate the significance of the seasonal effect. Because this is a factor variable, i.e. a set of dummy variables, we cannot just produce a confidence interval. Instead, we have to rely on a significance test, i.e. a partial F-test. Again, we compare what is obtained from OLS and GLS:

```
> drop1(fit.lm, test="F")
Single term deletions

Model:
temp ~ time + season
      Df Sum of Sq    RSS     AIC   F value    Pr(F)
<none>          6.4654 -1727.0
time      1  14.2274 20.6928 -1240.4 895.6210 <2e-16 ***
season   11   0.1744  6.6398 -1737.8   0.9982 0.4472

> anova(fit.gls)
Denom. DF: 407
      numDF   F-value p-value
(Intercept)     1 78.40801  <.0001
time            1 76.48005  <.0001
season          11  0.64371   0.7912
```

As for the trend, the result is identical with OLS and GLS. The seasonal effect is non-significant with p-values of 0.45 (OLS) and 0.79 (GLS). Again, the latter is the value we must believe in. We conclude that there is no seasonality in global warming – but there is a trend.

Example 2: Air Pollution

For finishing the air pollution example, we also perform a GLS fit here. We identified an *AR(1)* as the correct dependency structure for the errors. Thus, we define it accordingly:

```
> dat      <- cbind(dat, time=1:30)
> corStruct <- corARMA(form=~time, p=1)
> model    <- formula(Oxidant ~ Wind + Temp)
> fit.gls  <- gls(model, data=dat, correlation=corStruct)
```

The output then is as follows:

```
> fit.gls
Generalized least squares fit by REML
  Model: model
  Data: dat
Log-restricted-likelihood: -72.00127
```

Coefficients:

(Intercept)	Wind	Temp
-3.7007024	-0.4074519	0.4901431

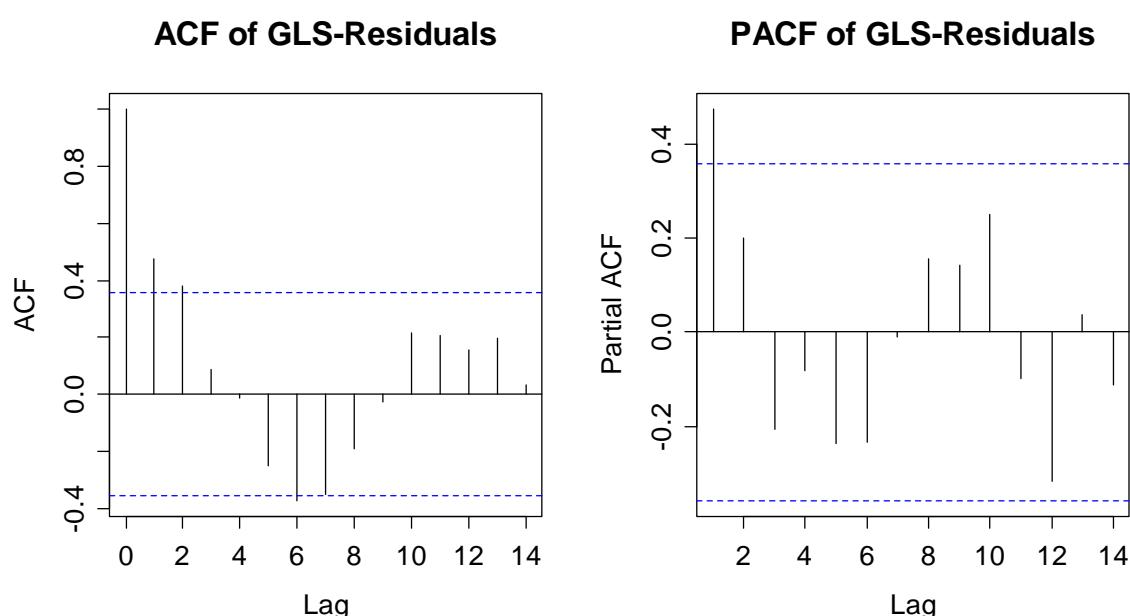
Correlation Structure: AR(1)

Formula: ~time

Parameter estimate(s):

Phi
0.5267549
Degrees of freedom: 30 total; 27 residual
Residual standard error: 3.066183

Again, we have to check if the GLS residuals show the stylized facts of an *AR(1)*:



This is the case, and thus we can draw inference from the GLS results. The confidence intervals of the regression coefficients are:

```
> confint(fit.lm, c("Wind", "Temp"))
      2.5 %    97.5 %
Wind -0.6044311 -0.2496841
Temp  0.2984794  0.7422260

> confint(fit.gls, c("Wind", "Temp"))
      2.5 %    97.5 %
Wind -0.5447329 -0.2701709
Temp  0.2420436  0.7382426
```

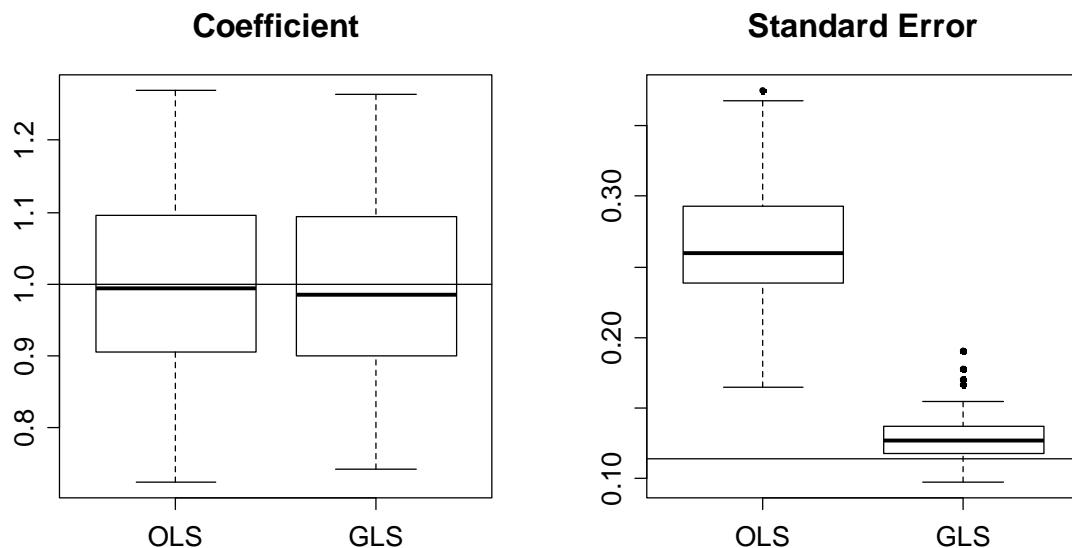
Here the differences among point estimates and confidence intervals are not very pronounced. This has to do with the fact that the correlation among the errors is weaker than in the global temperature example. But we emphasize again that the GLS results are the one to be relied on and the magnitude of the difference between OLS and GLS can be tremendous.

Simulation Study

We provide further evidence for the importance of the GLS approach by performing a simulation study in which the resulting coefficients and standard errors are compared to the ones obtained from OLS regression. We consider the following, relatively simple model:

$$\begin{aligned}x_t &= t / 50 \\y_t &= x_t + 2(x_t)^2 + E_t\end{aligned}$$

where E_t is from an AR(1) process with $\alpha_1 = -0.65$. The innovations are Gaussian with $\sigma = 0.1$. Regression coefficients and the true standard deviations of the estimators are known in this extraordinary situation. However, we generate 100 realizations with series length $n = 50$, on each perform OLS and GLS regression and record both point estimate and standard error.



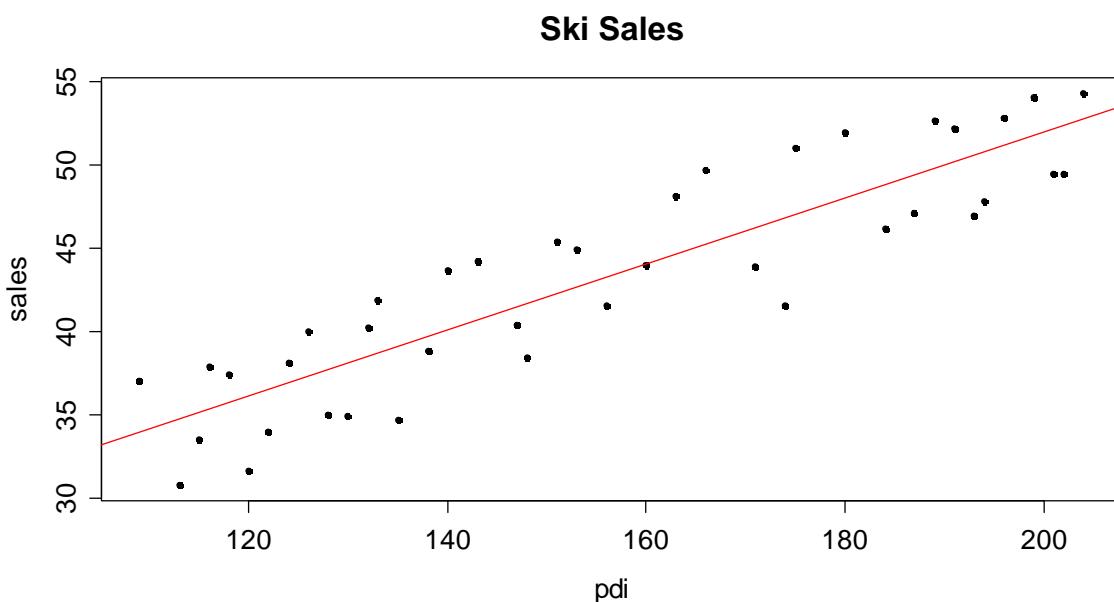
The simulation outcome is displayed by the boxplots in the figure above. While the point estimator for β_1 in the left panel is unbiased for both OLS and GLS, we observe that the standard error for $\hat{\beta}_1$ is very poor when the error correlation is not accounted for. We emphasize again that OLS regression with time series will inevitably lead to spuriously significant predictors and thus, false conclusions. Hence, it is absolutely key to inspect for possible autocorrelation in the regression residuals and apply the `gls()` procedure if necessary.

However, while `gls()` can cure the problem of autocorrelation in the error term, it does not solve the issue from the root. Sometimes, even this is possible. In the next subsection, we conclude the chapter about time series regression by showing how correlated errors can originate, and what practice has to offer for deeper understanding of the problem.

7.5 Missing Predictor Variables

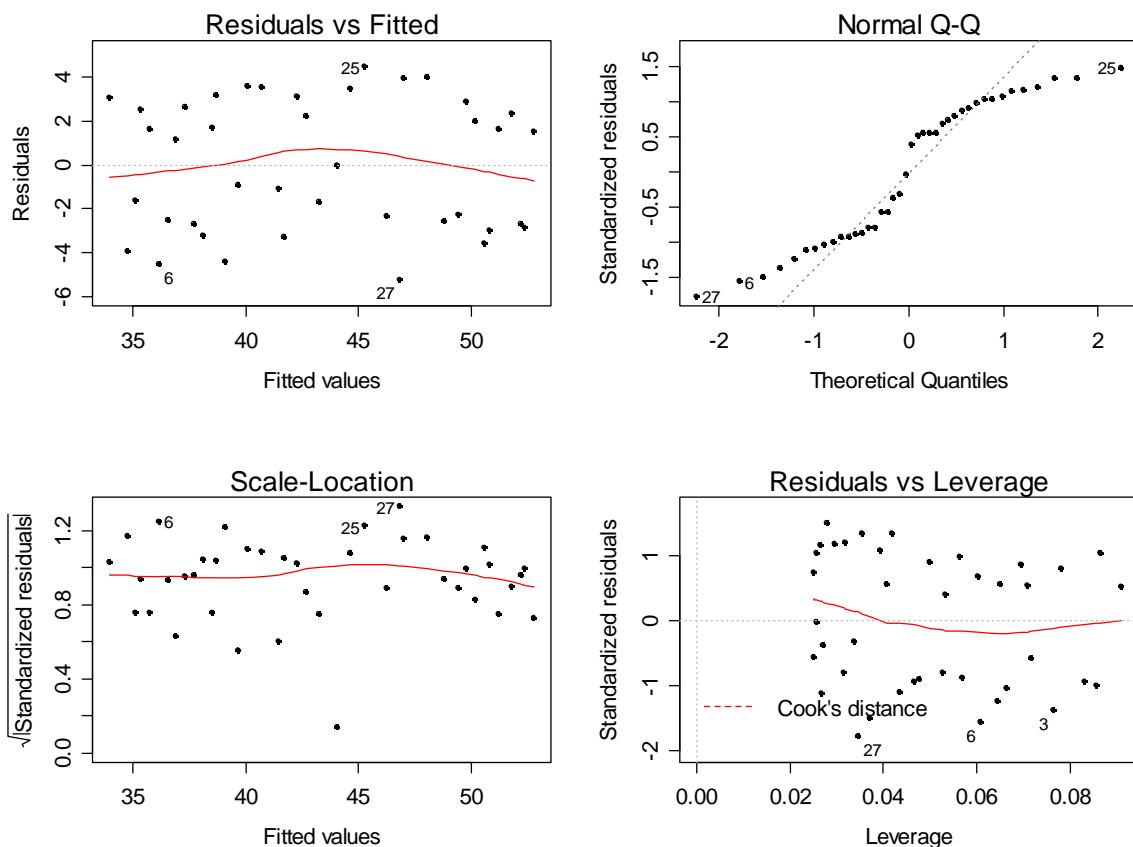
The presence correlated errors is often due to missing predictors. For illustration, we consider a straightforward example of a ski selling company in the US. The quarterly sales Y_t are regressed on the personal disposable income (PDI) which is the one and only predictor x_t . We display the two time series in a scatterplot and enhance it with the OLS regression line.

```
> ## Loading the data
> ski           <- read.table("ski.dat", header=TRUE)
> names(ski) <- c("time", "sales", "pdi", "season")
>
> ## Scatterplot
> par(mfrow=c(1,1))
> plot(sales ~ pdi, data=ski, pch=20, main="Ski Sales")
>
> ## LS modeling and plotting the fit
> fit <- lm(sales ~ pdi, data=ski)
> abline(fit, col="red")
```



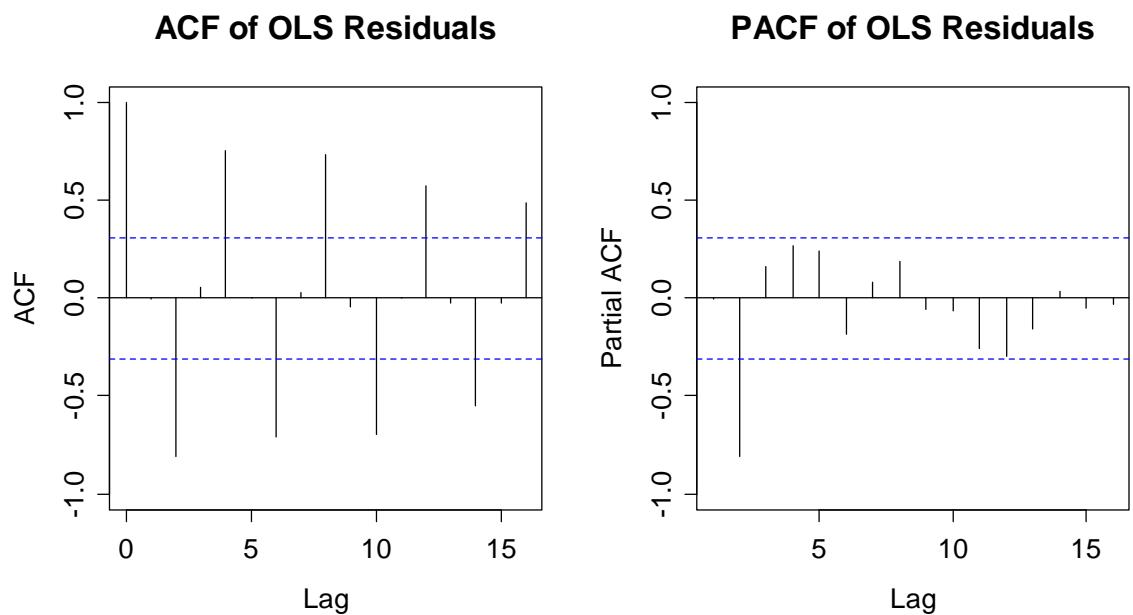
The coefficient of determination is rather large, i.e. $R^2 = 0.801$ and the linear fit seems adequate, i.e. a straight line seems to correctly describe the systematic relation between sales and PDI. However, the model diagnostic plots (see the next page) show some rather special behavior, i.e. there are hardly any “small” residuals (in absolute value). Or more precisely, the data points almost lie on two lines around the regression line, with almost no points near or on the line itself.

```
> ## Residual diagnostics
> par(mfrow=c(2,2))
> plot(fit, pch=20)
```

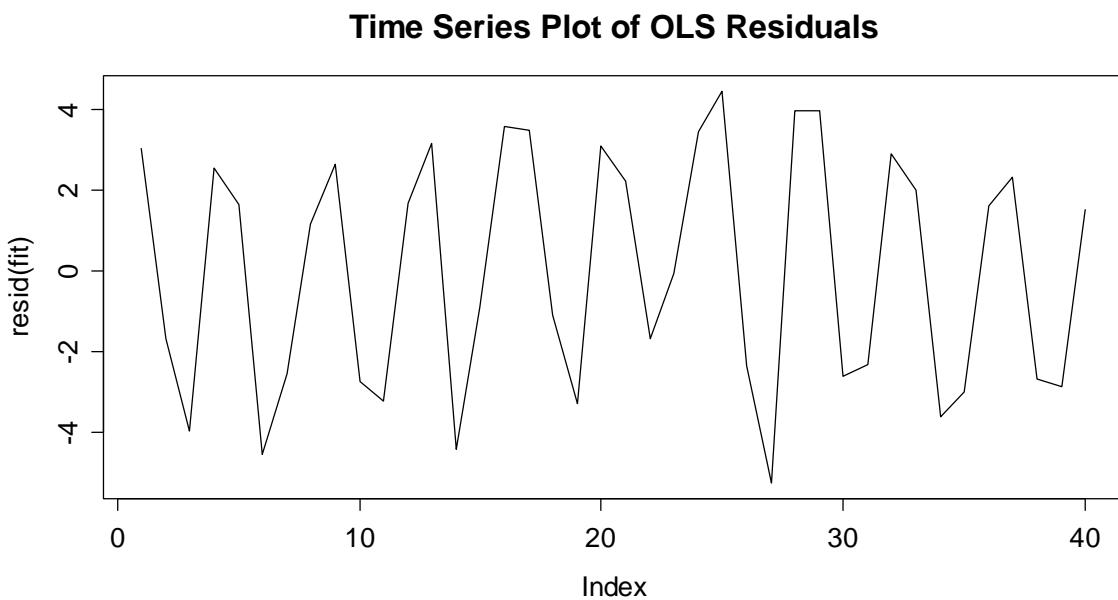


As the next step, we analyze the correlation of the residuals and perform a Durbin-Watson test. The result is as follows:

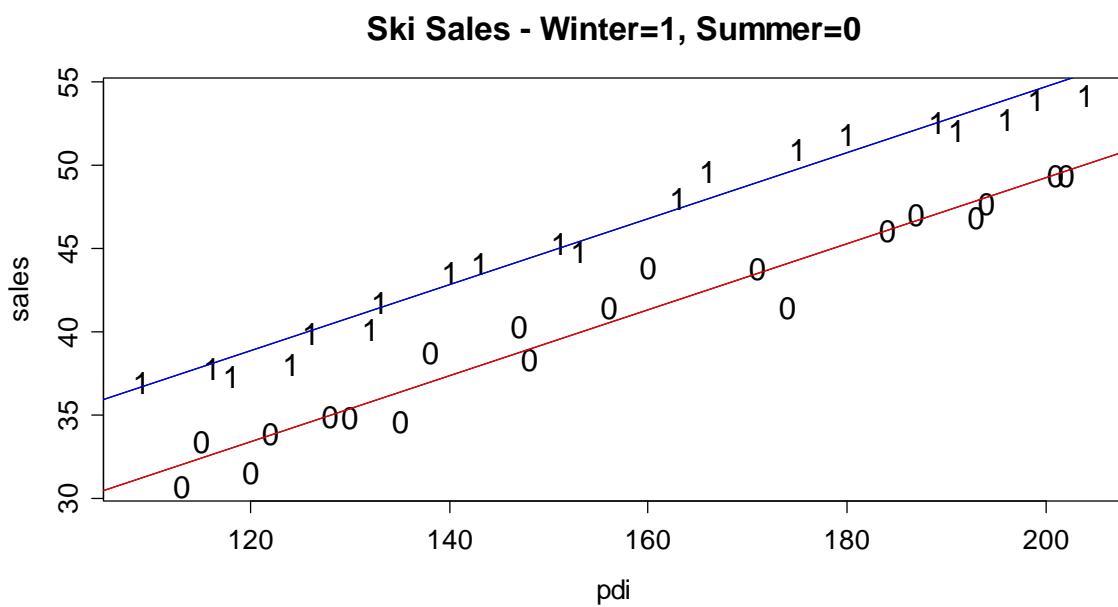
```
> dwtest(fit)
data: fit
DW = 1.9684, p-value = 0.3933
alt. hypothesis: true autocorrelation is greater than 0
```



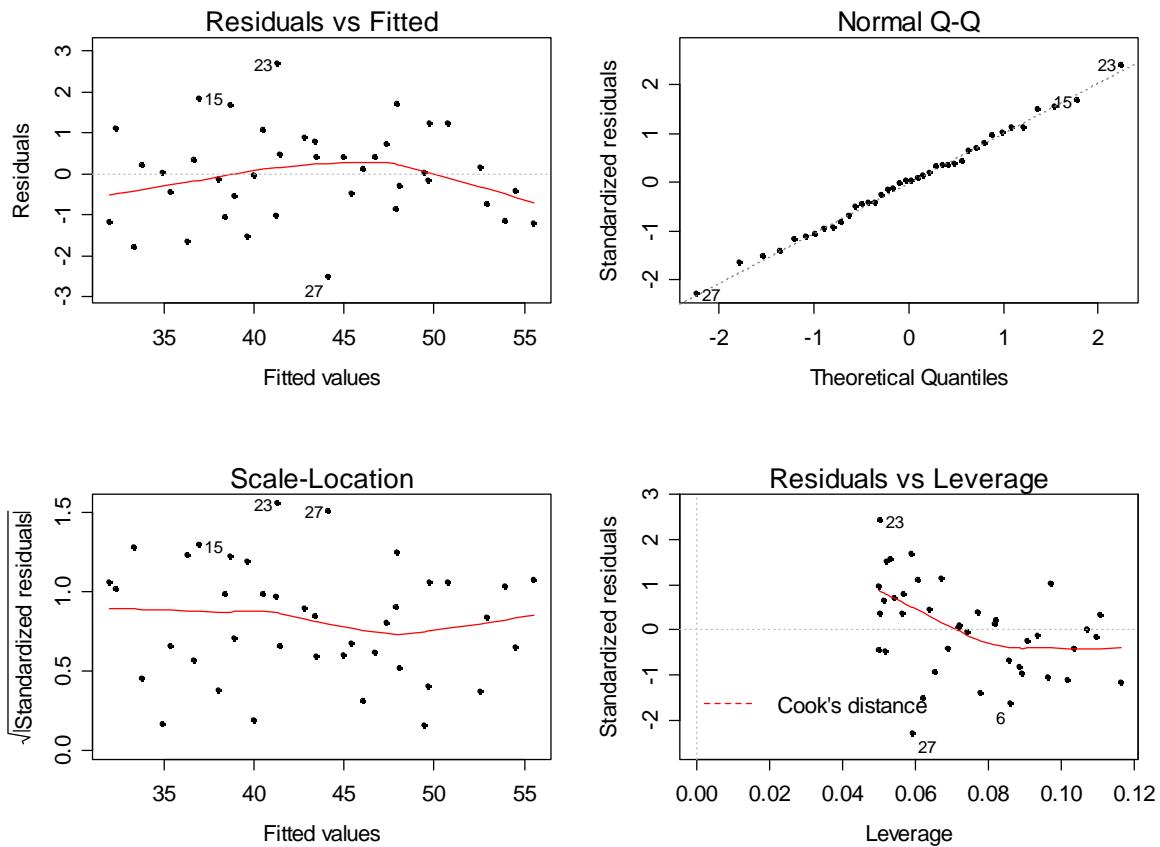
While the Durbin-Watson test does not reject the null hypothesis, the residuals seem very strongly correlated. The ACF exhibits some decay that may still qualify as exponential, and the PACF has a clear cut-off at lag 2. Thus, an $AR(2)$ model could be appropriate. And because it is an $AR(2)$ where α_1 and $\rho(1)$ are very small, the Durbin-Watson test fails to detect the dependence in the residuals. The time series plot is as follows:



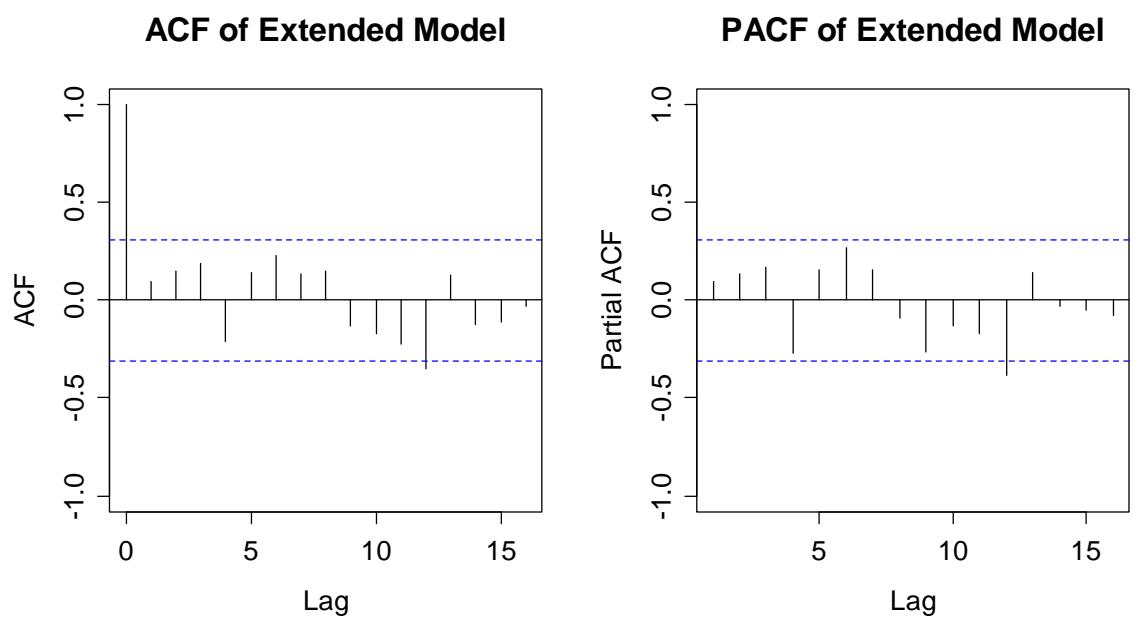
While we could now account for the error correlation with a GLS, it is always better to identify the reason behind the dependence. I admit this is suggestive here, but as mentioned in the introduction of this example, these are quarterly data and we might have forgotten to include the seasonality. It is not surprising that ski sales are much higher in fall and winter and thus, we introduce a factor variable which takes the value 0 in spring and summer, and 1 else.



Introducing the seasonal factor variable accounts to fitting two parallel regression lines for the winter and summer term. Eyeballing already lets us assume that the fit is good. This is confirmed when we visualize the diagnostic plots:



The unwanted structure is now gone, as is the correlation among the errors:



Apparently, the addition of the season as an additional predictor has removed the dependence in the errors. Rather than using GLS, a sophisticated estimation procedure, we have found a simple model extension that describes the data well and is certainly easier to interpret (especially when it comes to prediction) than a model that is built on correlated errors.

We conclude by saying that using GLS for modeling dependent errors should only take place if care has been taken that no important and/or obvious predictors are missing in the model.

8 Forecasting

One of the principal goals with time series analysis is to produce predictions which show the future evolution of the data. This is what it is: an extrapolation in the time domain. And as we all know, extrapolation is always (at least slightly) problematic and can lead to false conclusions. Of course, this is no different with time series forecasting.

The saying is that the task we are faced with can be compared to driving a car by looking through the rear window mirror. While this may work well on a wide motorway that runs mostly straight and has a few gentle bends only, things get more complicated as soon as there are some sharp and unexpected bends in the road. Then, we would need to drive very slowly to stay on track. This all translates directly to time series analysis. For series where the signal is much stronger than the noise, accurate forecasting is possible. However, for noisy series, there is a great deal of uncertainty in the predictions, and they are at best reliable for a very short horizon.

From the above, one might conclude that the principal source of uncertainty is inherent in the process, i.e. comes from the innovations. However, in practice, this is usually different, and several other factors can threaten the reliability of any forecasting procedure. In particular:

- We need to be certain that the data generating process does not change over time, i.e. continues in the future as it was observed in the past.
- When we choose/fit a model based on a realization of data, we have no guarantee that it is the correct, i.e. data-generating one.
- Even if we are so lucky to find the correct data-generating process (or in cases we know it), there is additional uncertainty arising from the estimation of the parameters.

Keeping these general warnings in mind, we will now present several approaches to time series forecasting. First, we deal with stationary processes and present, how AR, MA and ARMA processes can be predicted. These principles can be extended to the case of ARIMA and SARIMA models, such that forecasting series with either trend and/or seasonality is also possible.

As we had seen in section 4.3, the decomposition approach for non-stationary time series helps a great deal for visualization and modeling. Thus, we will present some heuristics about how to produce forecasts with series that were decomposed into trend, seasonal pattern and a stationary remainder. Last but not least, we present the method of exponential smoothing. This was constructed as a model-free, intuitive weighting scheme that allows forecasting of time series. Due to its simplicity and the convenient implementation in the `HoltWinters()` procedure in R, it is very popular and often used in applied sciences.

8.1 Forecasting ARMA

We suppose that we are given a time series, for which an appropriate AR, MA or ARMA model was identified, the parameters were successfully estimated and where the residuals exhibited the required properties, i.e. looked like White Noise. Under these circumstances, forecasts may be readily computed. Given data up to time n , the forecasts will involve either involve the past observations, and/or the residuals.

In mathematical statistics, many forecasting methods have been studied on a theoretical basis with the result that the minimum mean squared error forecast $\hat{X}_{n+k,1:n}$ for k steps ahead is given by the conditional expectation, i.e.:

$$\hat{X}_{n+k,1:n} = E[X_{n+k} | X_1, \dots, X_n]$$

In evaluating this term, we use the fact that the best forecast of all future innovation terms $E_t, t > n$ is simply zero. We will be more specific in the following subsections.

8.1.1 Forecasting AR(1)

For simplicity, we first consider a mean-zero, stationary AR(1) process with model equation:

$$X_t = \alpha_1 X_{t-1} + E_t,$$

where E_t is the innovation, for which we do not need to assume a particular distribution. As we will see below, it is convenient to assume Gaussian E_t , because this allows for an easy derivation of a prediction interval. The conditional expectation at time $n+1$ is given by:

$$E[X_{n+1} | X_1, \dots, X_n] = \alpha_1 x_n.$$

Thus, we can forecast the next instance of a time series with the observed value of the previous one, in particular:

$$\hat{X}_{n+1,1:n} = \alpha_1 x_n.$$

For the k -step forecast with $k > 1$, we just need to repeatedly plug-in the model equation, and as use the fact that the conditional expectation of the innovations is zero:

$$\begin{aligned}\hat{X}_{n+k,1:n} &= E[X_{n+k} | X_1, \dots, X_n] \\ &= E[\alpha_1 X_{n+k-1} + E_{n+k} | X_1, \dots, X_n] \\ &= \alpha_1 E[X_{n+k-1} | X_1, \dots, X_n] \\ &= \dots \\ &= \alpha_1^k x_n\end{aligned}$$

For any stationary $AR(1)$ process, the k -step forecast beyond the end of a series depends on the last observation x_n only and goes to zero exponentially quickly. For practical implementation with real data, we would just plug-in the estimated model parameter $\hat{\alpha}_1$ and can so produce a forecast for any desired horizon.

As always, a prediction is much more useful in practice if one knows how precise it is. Under the assumption of Gaussian innovations, a 95% prediction interval can be derived from the conditional variance $Var(X_{n+k} | X_1, \dots, X_n)$. For the special case of $k=1$ we obtain:

$$\alpha_1 x_n \pm 1.96 \cdot \sigma_E .$$

Again, for practical implementation, we need to plug-in $\hat{\alpha}_1$ and $\hat{\sigma}_E$. For a k -step prediction, the 95% prognosis interval is:

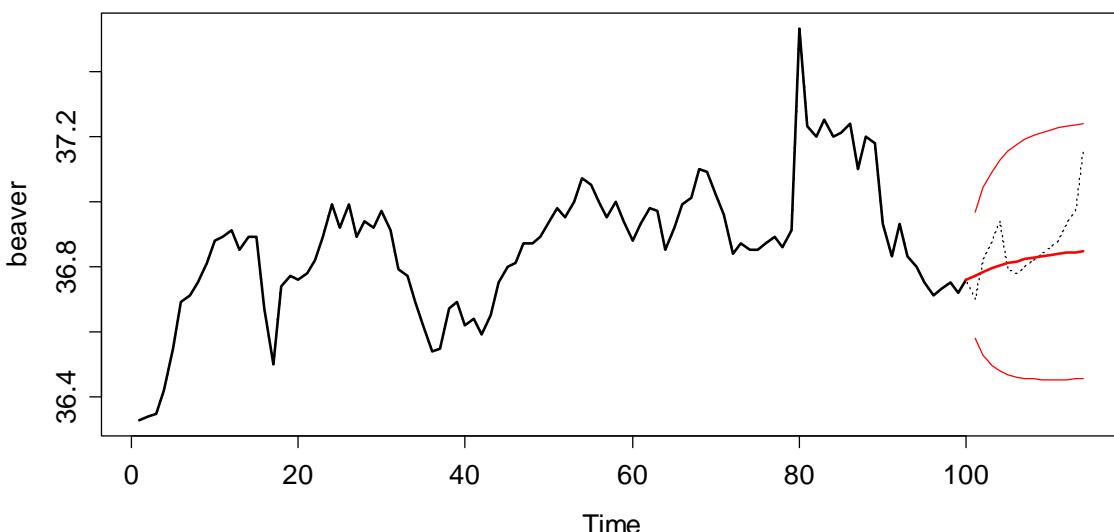
$$\alpha_1 x_n \pm 1.96 \cdot \left(1 + \sum_{j=1}^{k-1} \alpha_1^{2j}\right) \cdot \sigma_E .$$

For increasing prediction horizon k , the conditional variance goes to $\sigma_E^2 / (1 - \alpha_1^2)$, which is the process variance σ_x^2 . Thus, for the 1-step forecast, the uncertainty in the prediction is given by the innovation variance σ_E alone, while for increasing horizon k the prognosis interval gets wider is finally determined by the process variance.

Practical Example

We now turn our attention to a practical example, where we apply the **R** functions which implement the above theory. This is the Beaver data we had already discussed in section 4.4.3. An $AR(1)$ model is appropriate, and for estimating the coefficients, we omit the last 14 observations from the data. These will be predicted, and the true values are needed for verifying the prediction.

Beaver Data: 14-Step Prediction Based on AR(1)



The R commands for fitting the model on the training data and producing the 14-step prediction are as follows:

```
> btrain <- window(beaver, 1, 100)
> fit      <- ar.burg(btrain, order=1)
> forecast <- predict(fit, n.ahead=14)
```

The `forecast` object is a list that has two components, `pred` and `se`, which contain the point predictions and the standard error, respectively. We now turn our attention to how the forecast is visualized:

```
> plot(beaver, lty=3)
> lines(btrain, lwd=2)
> lines(pred$pred, lwd=2, col="red")
> lines(pred$pred+pred$se*1.96, col="red")
> lines(pred$pred-pred$se*1.96, col="red")
```

One more issue requires some attention here: for the Beaver data, a pure AR(1) process is not appropriate, because the global series mean is clearly different from zero. The way out is to de-mean the series, then fit the model and produce forecasts, and finally re-adding the global mean. R does all this automatically.

We conclude by summarizing what we observe in the example: the forecast is based on the last observed value $x_{100} = 36.76$, and from there approaches the global series mean $\hat{\mu} = 36.86$ exponentially quick. Because the estimated coefficient is $\hat{\alpha}_1 = 0.87$, and thus relatively close to one, the approach still takes some time.

8.1.2 Forecasting AR(p)

Forecasting from AR(p) processes works based on the same principles as explained above for the AR(1), i.e. we use the conditional expected value. The algebra for writing the forecasting formulae is somewhat more laborious, but not more difficult. Thus, we do without displaying it here, and directly present the 1-step-forecast:

$$\hat{X}_{n+1,1:n} = \alpha_1 x_n + \alpha_2 x_{n-1} + \dots + \alpha_p x_{n-p}$$

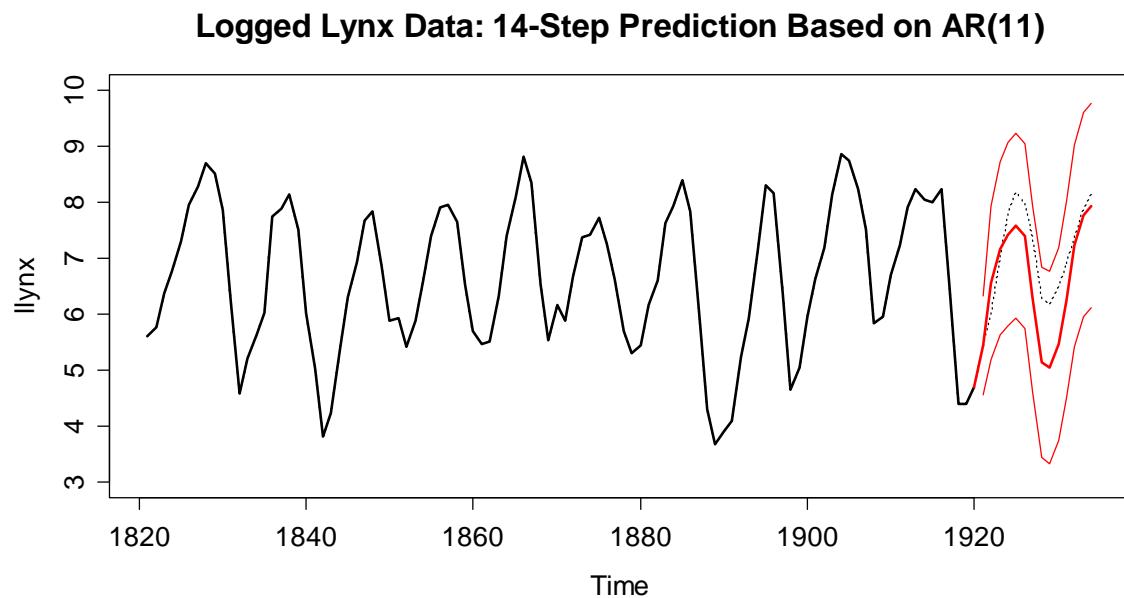
The question is, what do we do for longer forecasting horizons? There, the forecast is again based on the linear combination of the p past instances. For the ones with an index between 1 and n , the observed value x_t is used. Else, if the index exceeds n , we just plug-in the forecasted values $\hat{x}_{t,1:n}$. Thus, the general formula is:

$$\hat{X}_{n+k,1:n} = \alpha_1 \hat{X}_{n+k-1,1:n} + \dots + \alpha_p \hat{X}_{n+k-p,1:n},$$

where $\hat{X}_{t,1:n} = x_t$ in all cases where $t \leq n$, i.e. an observed value is available.

Practical Example

We consider the lynx data for which we had identified an AR(11) as a suitable model. Again, we use the first 100 observations for fitting the model and lay aside the last 14, which are in turn used for verifying the result. Also, we do without showing the R code, because it is identical to the one from the previous example.



We observe that the forecast tracks the general behavior of the series well, though the level of the series is underestimated somewhat. This is, however, not due to an “error” of ours, it is just that the values were higher than the past observations suggested. We finish this section with some remarks:

- Forecasting from an $AR(p)$ only requires knowledge about the last p instances of the series, plus the model parameters $\alpha_1, \dots, \alpha_p$ and the global series mean μ . Earlier values of the series are not required, the model thus has a Markov property of order p .
- The prediction intervals only account for the uncertainty caused by the innovation variance, but not for the one caused by model misconception, and the plug-in of estimated parameters. Thus in practice, a true 95% interval would most likely be wider than shown above.

8.1.3 Forecasting MA(1)

We here consider an invertible $MA(1)$ process, where the model equation is as follows:

$$X_t = E_t + \beta_1 E_{t-1},$$

where E_t is an innovation with expectation zero and constant variance.

As above, the forecast $\hat{X}_{n+k,\text{lin}}$ will again be based on the conditional expectation $E[X_{n+k} | X_1, \dots, X_n]$. We get to a solution if we plug-in the model equation. First, we assume that $k \geq 2$, i.e. predict at least 2 time steps ahead.

$$\begin{aligned}\hat{X}_{n+k,\text{lin}} &= E[X_{n+k} | X_1, \dots, X_n] \\ &= E[E_{n+k} + \beta_1 E_{n+k-1} | X_1, \dots, X_n] \\ &= E[E_{n+k} | X_1, \dots, X_n] + \beta_1 E[E_{n+k-1} | X_1, \dots, X_n] \\ &= 0\end{aligned}$$

The best forecast $MA(1)$ forecast for horizons 2 and up is thus zero. Remember that we require E_t being an innovation, and thus independent from previous instances $X_s, s < t$ of the time series process. Next, we address the 1-step forecast. This is more problematic, because the above derivation leads to:

$$\begin{aligned}\hat{X}_{n+1,\text{lin}} &= \dots \\ &= \beta_1 E[E_n | X_1, \dots, X_n] \\ &\neq 0 \text{ (generally)}\end{aligned}$$

The 1-step forecast thus generally is different from zero. Moreover, the term $E[E_n | X_1, \dots, X_n]$ is difficult to determine. Using some mathematical trickery, we can at least propose an approximate value. This trick is to move the point of reference into the infinite past, i.e. conditioning on all previous instances of the $MA(1)$ process. We denote

$$e_n := E[E_n | X_{-\infty}^n].$$

By successive substitution, we then write the $MA(1)$ as an $AR(\infty)$. This yields

$$E_n = \sum_{j=0}^{\infty} (-\beta_1)^j X_{n-j}.$$

If we condition the expectation of E_n on the infinite past of the series X_t , we can plug-in the realizations x_t and obtain:

$$E[E_n | X_{-\infty}^n] = e_n = \sum_{j=0}^{\infty} (-\beta_1)^j x_{n-j}.$$

This is of course somewhat problematic for practical implementation, because we only have realizations for x_1, \dots, x_n . However, because for invertible $MA(1)$ processes, $|\beta_1| < 1$, the impact of early observations dies out exponentially quickly. Thus, we let $x_t = 0$ for $t < 1$, and thus also have that $e_t = 0$ for $t < 1$. Also, we plug-in the estimated model parameter $\hat{\beta}_1$, and thus, the 1-step forecast for an $MA(1)$ is:

$$\hat{X}_{n+1,\text{lin}} = \sum_{j=0}^{n-1} \hat{\beta}_1 (-\hat{\beta}_1)^j x_{n-j}$$

This is a sum of all observed values, with exponentially decaying weights.

8.1.4 Forecasting MA(q)

When forecasting from $MA(q)$ processes, we encounter the same difficulties as above. The prediction for horizons exceeding q are all zero, but anything below contains terms for which the considerations in section 8.1.3 are again necessary. We do without displaying this, and proceed to giving the formulae for $ARMA(p,q)$ forecasting, from which the ones for $MA(q)$ can be learned.

8.1.5 Forecasting ARMA(p,q)

We are considering stationary and invertible $ARMA(p,q)$ processes. The model equation for X_{n+1} then is:

$$X_{n+1} = \alpha_1 X_n + \dots + \alpha_p X_{n+1-p} + E_{n+1} + \beta_1 E_n + \dots + \beta_q E_{n+1-q}$$

As this model equation contains past innovations, we face the same problems as in section 8.1.3 when trying to derive the forecast for horizons $\leq q$. These can be mitigated, if we again condition on the infinite past of the process.

$$\begin{aligned}\hat{X}_{n+1,ln} &= E[X_{n+1} | X_{-\infty}^n] \\ &= \sum_{i=1}^p \alpha_i E[X_{n+1-i} | X_{-\infty}^n] + E[E_{n+1} | X_{-\infty}^n] + \sum_{j=1}^q \beta_j E[E_{n+1-j} | X_{-\infty}^n] \\ &= \sum_{i=1}^p \alpha_i x_{n+1-i} + \sum_{i=1}^q \beta_i E[E_{n+1-i} | X_{-\infty}^n]\end{aligned}$$

If we are aiming for k -step forecasting, we can use a recursive prediction scheme:

$$\hat{X}_{n+k,ln} = \sum_{i=1}^p \alpha_i E[X_{n+k-i} | X_{-\infty}^n] + \sum_{j=1}^q \beta_j E[E_{n+k-j} | X_{-\infty}^n],$$

where for the AR - and MA -part the conditional expectations are:

$$E[X_t | X_{-\infty}^n] = \begin{cases} x_t, & \text{if } t \leq n \\ \hat{X}_{t,ln}, & \text{if } t > n \end{cases}$$

$$E[E_t | X_{-\infty}^n] = \begin{cases} e_t, & \text{if } t \leq n \\ 0, & \text{if } t > n \end{cases}$$

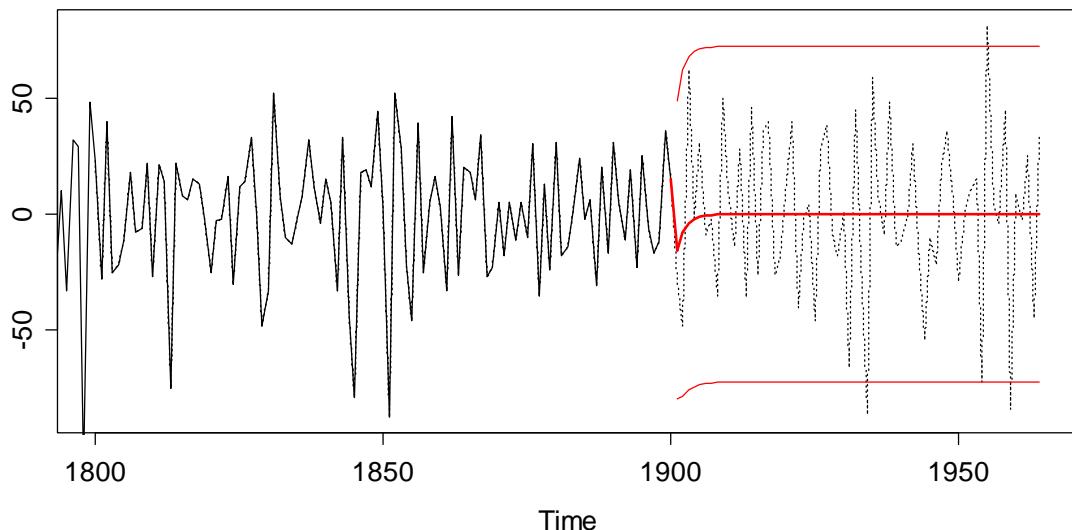
The terms e_t are then determined as outlined above in section 8.1.3, and for the model parameters, we are plugging-in the estimates. This allows us to generate any forecast from an $ARMA(p,q)$ model that we wish. The procedure is also known as Box-Jenkins procedure, after the two researchers who first introduced it.

Next, we illustrate this with a practical example, though in R, things are quite unspectacular. It is again the `predict()` procedure that is applied to a fit from `arima()`, the Box-Jenkins scheme that is employed runs in the background.

Practical Example

We here consider the Douglas Fir data which show the width of the year rings over a period from 1107 to 1964. Because the data are non-stationary, we take differences and model these. An ARMA(1,1) seems appropriate. We put the last 64 observations aside so that we can verify our predictions. Then, the model is fitted and the Box-Jenkins predictions are obtained. The result, including a 95% prognosis interval, is shown below.

Differenced Douglas Fir Data: 64-Step Prediction Based on ARMA(1,1)



We observe that the forecast goes to zero exponentially quickly. However, it is in fact different from zero for all times. Moreover, all observations down to the very first one are used for obtaining the predictions. Again, the ARMA model combines the properties from pure AR and MA processes.

8.2 Exponential Smoothing

8.2.1 Simple Exponential Smoothing

The objective in this section is to predict some future values X_{n+k} given an observed series $\{X_1, \dots, X_n\}$, and thus no different than before. We first assume that the data do not exhibit any deterministic trend or seasonality, or that these have been identified and removed. The (conditional) expected value of the process can change from one time step to the next, but we do not have any information about the direction of this change. A typical application is forecasting sales of a well-established product in a stable market. The model is:

$$X_t = \mu_t + E_t,$$

where μ_t is the non-stationary mean of the process at time t , and E_t are independent random innovations with expectation zero and constant variance σ_E^2 . We will here use the same notation as R does, and let a_t , called *level* of the series, be our estimate of μ_t . By assuming that there is no deterministic trend, an intuitive estimate for the level at time t is to take a weighted average of the current time series observation and the previous level:

$$a_t = \alpha x_t + (1 - \alpha)a_{t-1}, \text{ with } 0 < \alpha < 1.$$

Apparently, the value of α determines the amount of smoothing: if it is near 1, there is little smoothing and the level a_t closely tracks the series x_t . This would be appropriate if the changes in the mean of the series are large compared to the innovation variance σ_E^2 . At the other extreme, an α -value near 0 gives highly smoothed estimates of the current mean which take little account of the most recent observation. This would be the way to go for series with a large amount of noise compared to the signal size. A typical default value is $\alpha = 0.2$, chosen in the light that for most series, the change in the mean between t and $t-1$ is smaller than σ_E^2 . Alternatively, it is (with R) also possible to estimate α , see below.

Because we assume absence of deterministic trend and seasonality, the best forecast at time n for the future level of the series, no matter what horizon we are aiming for, is given by the level estimate at time n , i.e.

$$\hat{X}_{n+k,1:n} = a_n, \text{ for all } k = 1, 2, \dots$$

We can rewrite the weighted average equation in two further ways, which yields insight into how exponential smoothing works. Firstly, we can write the level at time t as the sum of a_{t-1} and the 1-step forecasting error and obtain the *update formula*:

$$a_t = \alpha(x_t - a_{t-1}) + a_{t-1}$$

Now, if we repeatedly apply back substitution, we obtain:

$$a_t = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + \alpha(1 - \alpha)^2 x_{t-2} + \dots$$

When written in this form, we see that the level a_t is a linear combination of the current and all past observations with more weight given to recent observations. The restriction $0 < \alpha < 1$ ensures that the weights $\alpha(1 - \alpha)^i$ become smaller as i increases. In fact, they are exponentially decaying and form a geometric series. When the sum over these terms is taken to infinity, the result is 1. In practice, the infinite sum is not feasible, but can be avoided by specifying $a_1 = x_1$.

For any given smoothing parameter α , the update formula plus the choice of $a_1 = x_1$ as a starting value can be used to determine the level a_t for all times $t = 2, 3, \dots$. The 1-step prediction errors e_t are given by:

$$e_t = x_t - \hat{x}_{t,k(t-1)} = x_t - a_{t-1} .$$

By default, R obtains a value for the smoothing parameter α by minimizing the sum of squared 1-step prediction errors, called *SS1PE*:

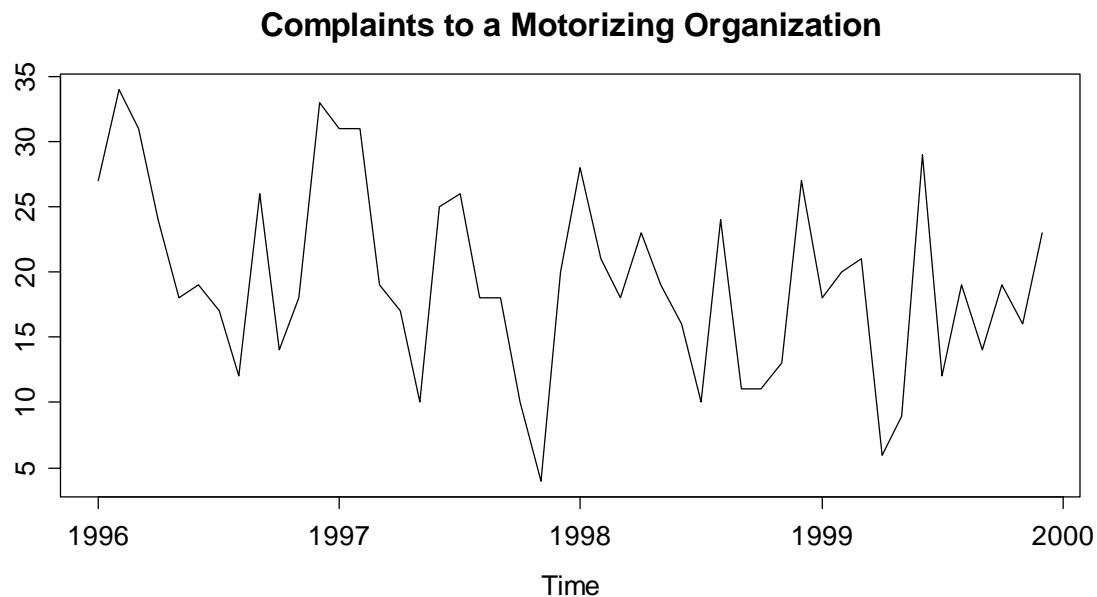
$$SS1PE = \sum_{t=2}^n e_t^2 .$$

There is some mathematical theory that examines the quality of the *SS1PE*-minimizing α . Not surprisingly, this depends very much on the true, underlying process. However in practice, this value is reasonable and allows for good predictions.

Practical Example

We here consider a time series that shows the number of complaint letters that were submitted to a motoring organization over the four years 1996-1999. At the beginning of year 2000, the organization wishes to estimate the current level of complaints and investigate whether there was any trend in the past. We import the data and do a time series plot:

```
> www <- "http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/"
> dat <- read.table(paste(www,"motororg.dat",sep="", head=T)
> cmpl <- ts(dat$complaints, start=c(1996,1), freq=12)
> plot(cmpl, ylab="", main="Complaints ...")
```



The series is rather short, and there is no clear evidence for a deterministic trend and/or seasonality. Thus, it seems sensible to use exponential smoothing here. The algorithm that was described above is implemented in R's `HoltWinters()` procedure. Please note that `HoltWinters()` can do more than plain exponential smoothing, and thus we have to set arguments `beta=FALSE` and `gamma=FALSE`.

If we do not specify a value for the smoothing parameter α with argument `alpha`, it will be estimated using the `SS1PE` criterion.

```
> fit <- HoltWinters(cmpl, beta=FALSE, gamma=FALSE); fit
Holt-Winters exponential smoothing without trend and without
seasonal component.
```

Call:

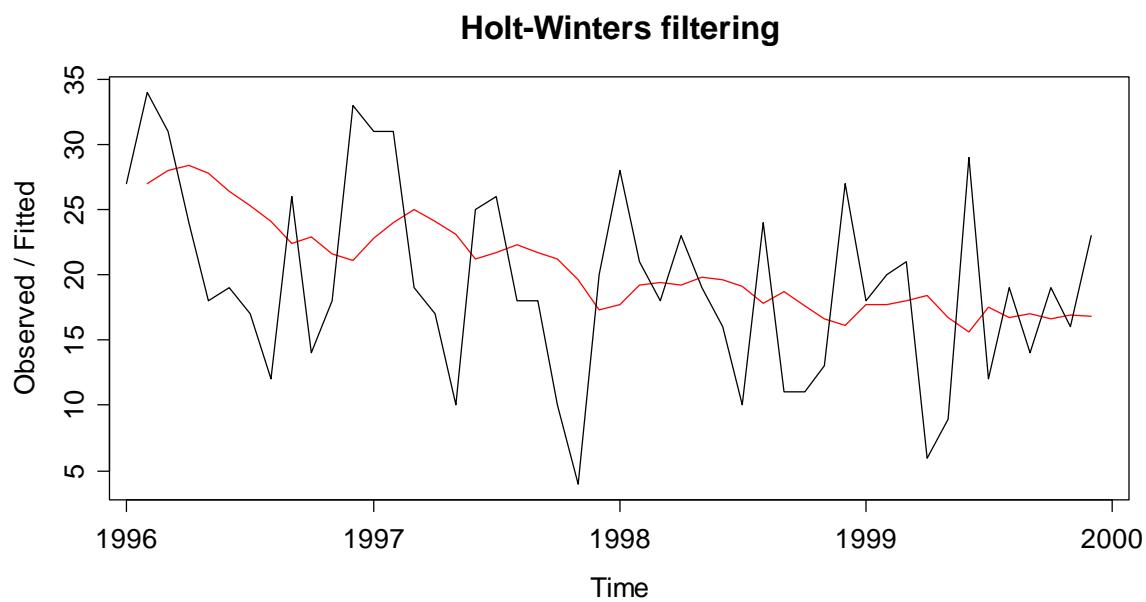
```
HoltWinters(x = cmpl, beta = FALSE, gamma = FALSE)
```

Smoothing parameters:

```
alpha: 0.1429622
beta : FALSE
gamma: FALSE
```

Coefficients:

```
[,1]
a 17.70343
> plot(fit)
```



The output shows that the level in December 1999, this is a_{48} , is estimated as 17.70. The optimal value for α according to the `SS1PE` criterion is 0.143, and the sum of squared prediction errors was 2502. Any other value for α will yield a worse result, thus we proceed and display the result visually.

8.2.2 The Holt-Winters Method

The simple exponential smoothing approach from above can be generalized for series which exhibit deterministic trend and/or seasonality. As we have seen in many examples, such series are the norm rather than the exception and thus, such a method comes in handy. It is based on these formulae:

$$\begin{aligned}a_t &= \alpha(x_t - s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \\b_t &= \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \\s_t &= \gamma(x_t - a_t) + (1-\gamma)s_{t-p}\end{aligned}$$

In the above equations, a_t is again the level at time t , b_t is called the slope and s_t is the seasonal effect. There are now three smoothing parameters α, β, γ which are aimed at level, slope and season. The explanation of these equations is as follows:

- The first updating equation for the level takes a weighted average of the most recent observation with the existing estimate of the previous period seasonal effect term subtracted, and the 1-step level forecast at $t-1$, which is given by level plus slope.
- The second updating equation takes a weighted average of the difference between the current and the previous level with the estimated slope at time $t-1$. Note that this can only be computed if a_t is available.
- Finally, we obtain another estimate for the respective seasonal term by taking a weighted average of the difference between observation and level with the previous estimate of the seasonal term for the same unit, which was made at time $t-p$.

If nothing else is known, the typical choice for the smoothing parameters is $\alpha = \beta = \gamma = 0.2$. Moreover, starting values for the updating equations are required. Mostly, one chooses $a_1 = x_1$, the slope $b_1 = 0$ and the seasonal effects s_1, \dots, s_p are either also set to zero or to the mean over the observations of a particular season. When applying the R function `HoltWinters()`, the starting values are obtained from the `decompose()` procedure, and it is possible to estimate the smoothing parameters through `SS1PE` minimization. The most interesting aspect are the predictions, though: the k -step forecasting equation for X_{n+k} at time n is:

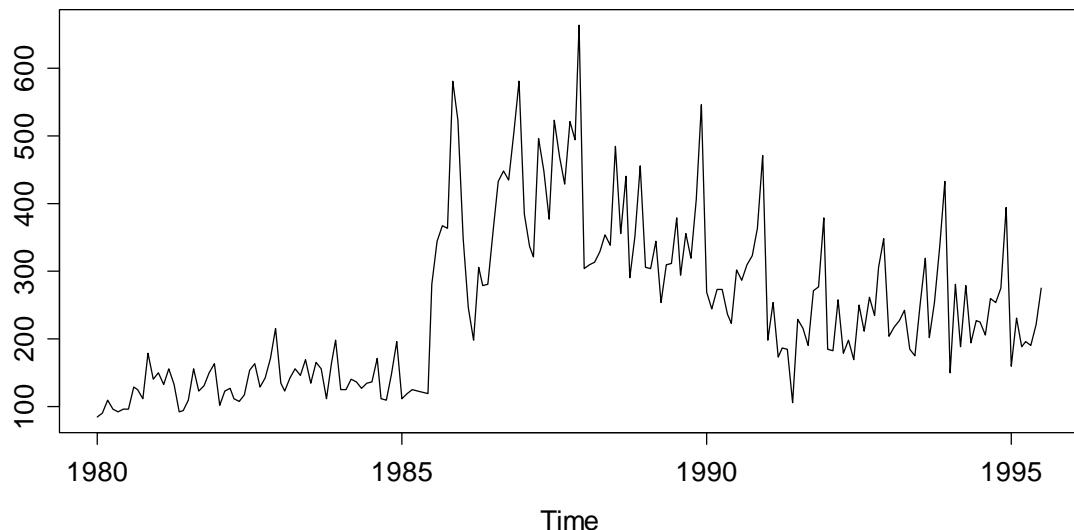
$$\hat{X}_{n+k,1:n} = a_n + kb_n + s_{n+k-p},$$

i.e. the current level with linear trend extrapolation plus the appropriate seasonal effect term. The following practical example nicely illustrates the method.

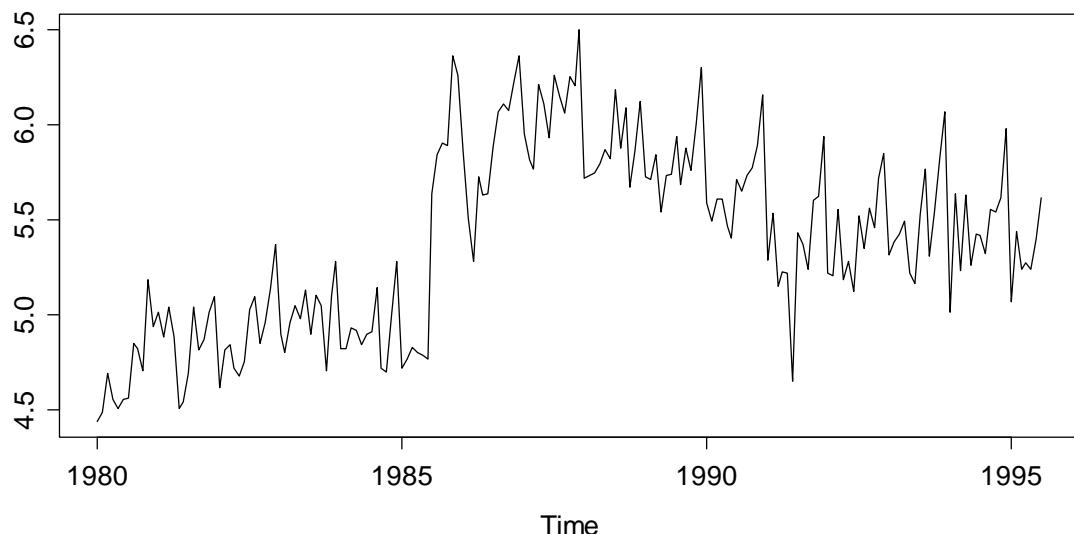
Practical Example

We here discuss the series of monthly sales (in thousands of litres) of Australian white wine from January 1980 to July 1995. This series features a deterministic trend, the most striking feature is the sharp increase in the mid-80ies, followed by a reduction to a distinctly lower level again. The magnitude of both the seasonal effect and the errors seem to be increasing with the level of the series, and are thus multiplicative rather than additive. We will cure this by a log-transformation of the series, even though there exists a multiplicative formulation of the Holt-Winters algorithm, too.

```
> www <- "http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/"
> dat <- read.table(paste(www,"wine.dat",sep="", header=T)
> aww <- ts(dat$sweetw, start=c(1980,1), freq=12)
> plot(aww, ylab="", main="Sales of Australian White Wine")
```

Sales of Australian White Wine

```
> plot(log(aww), ylab="", main="Logged Sales ...")
```

Logged Sales of Australian White Wine

The transformation seems successful, thus we proceed to the Holt-Winters modeling. When we apply parameter estimation by *SS1PE*, this is straightforward. The fit contains the current estimates for level, trend and seasonality. Note that these are only valid for time n , and not for the entire series. Anyhow, it is much better to visualize the sequence of a_t, b_t and γ_t graphically. Moreover, plotting the fitted values along with the time series is informative, too.

```

> fit
Holt-Winters exponential smoothing with trend and additive
seasonal component.

Call:
HoltWinters(x = log(aww))

Smoothing parameters:
alpha: 0.4148028
beta : 0
gamma: 0.4741967

Coefficients:
a      5.62591329
b      0.01148402
s1    -0.01230437
s2     0.01344762
s3     0.06000025
s4     0.20894897
s5     0.45515787
s6    -0.37315236
s7    -0.09709593
s8    -0.25718994
s9    -0.17107682
s10   -0.29304652
s11   -0.26986816
s12   -0.01984965

```

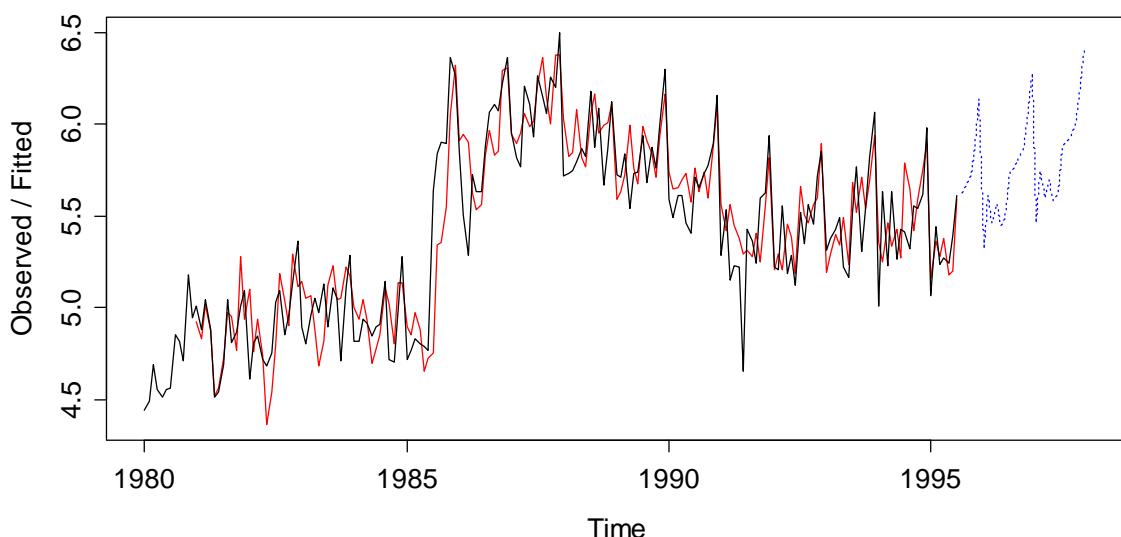
The coefficient values (at time n) are also the ones which are used for forecasting from that series with the formula given above. We produce a prediction up until the end of 1998, which is a 29-step forecast. The R commands are:

```

> plot(fit, xlim=c(1980, 1998))
> lines(predict(fit, n.ahead=29), col="blue", lty=3)

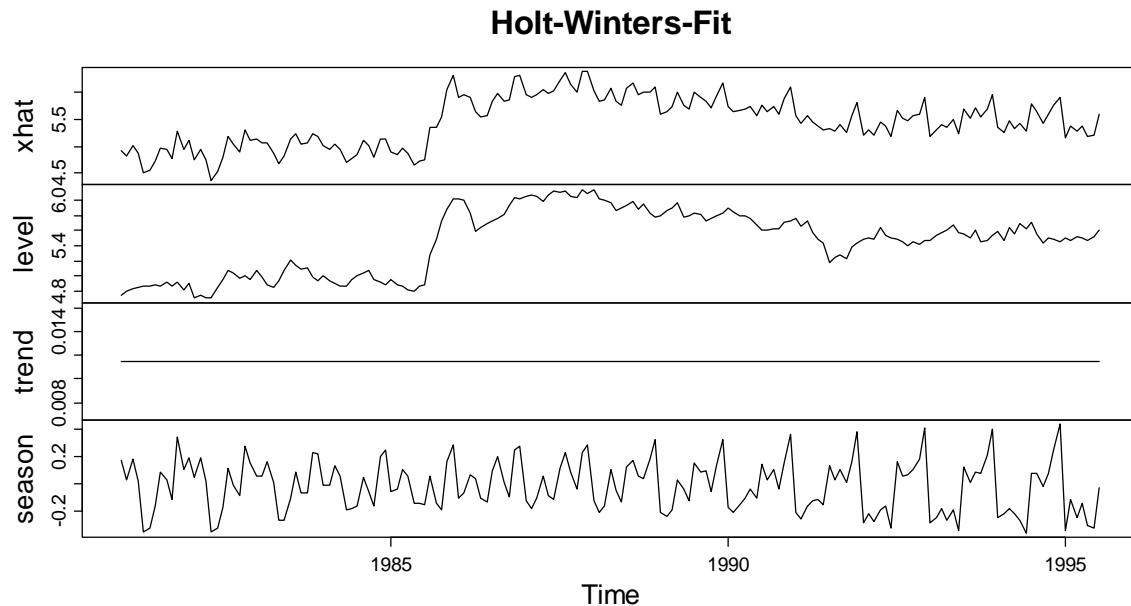
```

Holt-Winters filtering



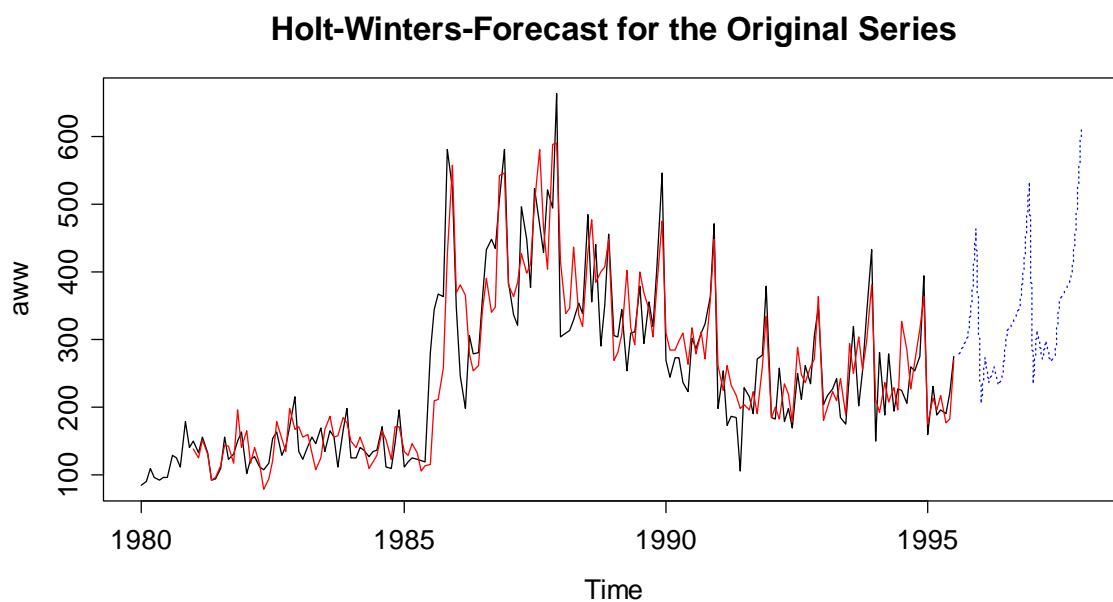
It is also very instructive to plot how level, trend and seasonality evolved over time. This can be done very simply in R:

```
> plot(fit$fitted, main="Holt-Winters-Fit")
```



Since we are usually more interested in the prediction on the original scale, i.e. in liters rather than log-liters of wine, we just re-exponentiate the values. Please note that the result is an estimate of the median rather than the mean of the series. There are methods for correction, but the difference is usually only small.

```
> plot(aww, xlim=c(1980, 1998))
> lines(exp(fit$fitted[,1]), col="red")
> lines(exp(predict(fit, n.ahead=29)), col="blue", lty=3)
```



Also, we note that the (insample) 1-step prediction error is equal to 50.04, which is quite a reduction when compared to the series' standard deviation which is 121.4. Thus, the Holt-Winters fit has substantial explanatory power. Of course, it would now be interesting to test the accuracy of the predictions. We recommend that you, as an exercise, put aside the last 24 observations of the Australian white wine data, and run a forecasting evaluation where all the methods (SARIMA, decomposition approaches, Holt-Winters) compete against each other.

9 Multivariate Time Series Analysis

While the header of this section says *multivariate* time series analysis, we will here restrict to two series $X_1 = (X_{1,t})$ and $X_2 = (X_{2,t})$, and thus *bivariate* time series analysis, because an extension to more than two series is essentially analogous. Please note that a prerequisite for all the theory in this section is that the series X_1 and X_2 are stationary.

Generally speaking, the goal of this section is to describe and understand the (inter)dependency between two series. We introduce the basic concepts of cross correlation and transfer function models, warn of arising difficulties in interpretation and show how these can be mitigated.

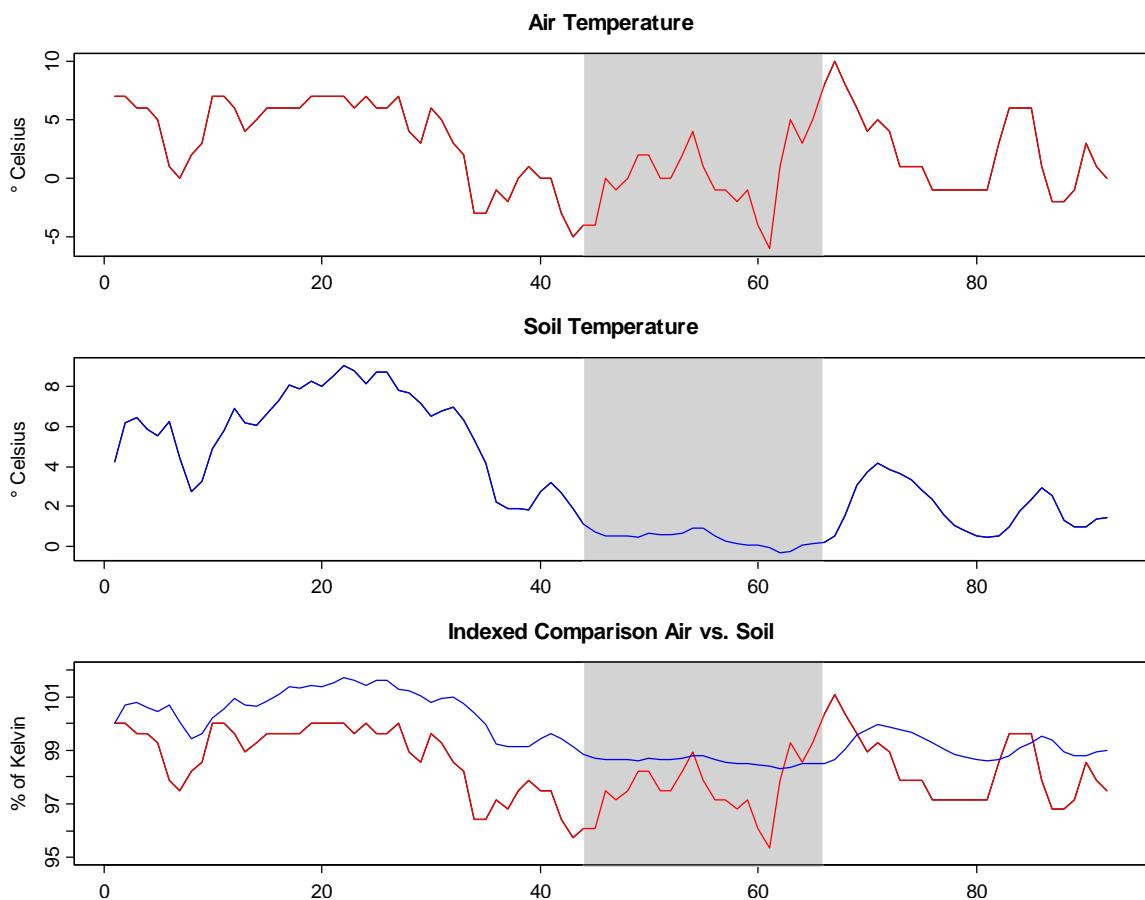
9.1 Practical Example

We will illustrate the theory on multivariate time series analysis with a practical example. The data were obtained in the context of the diploma thesis of Evelyn Zenklusen Mutter, a former WBL student who works for the Swiss Institute for Snow and Avalanche Research SLF. The topic is how the ground temperature in permafrost terrain depends on the ambient air temperature. The following section gives a few more details.

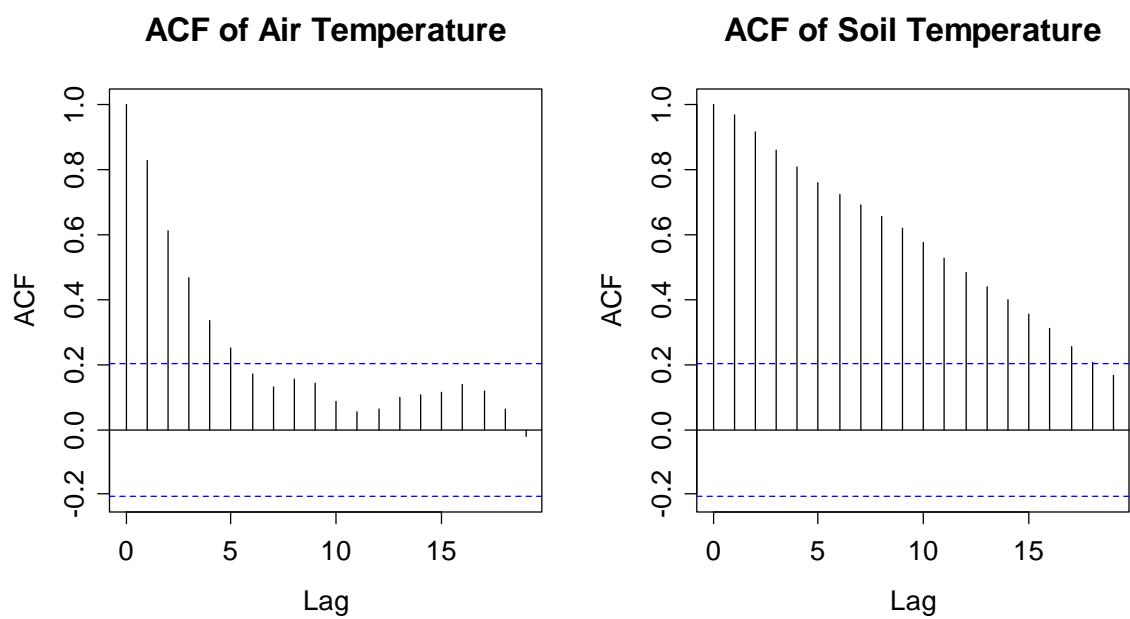
Ambient air temperatures influence ground temperatures with a certain temporal delay. Borehole temperatures measured at 0.5m depth in alpine permafrost terrain, as well as air temperatures measured at or nearby the boreholes will be used to model this dependency. The reaction of the ground on the air temperature is influenced by various factors such as ground surface cover, snow depth, water or ground ice content. To avoid complications induced by the insulating properties of the snow cover and by phase changes in the ground, only the snow-free summer period when the ground at 0.5m is thawed will be considered.

We here consider only one single borehole, it is located near the famous *Hörnli hut* at the base of *Matterhorn* near Zermatt/CH on 3295m above sea level. The air temperature was recorded on the nearby *Platthorn* at 3345m of elevation and 9.2km distance from the borehole. Data are available from beginning of July 2006 to the end of September 2006. After the middle of the observation period, there is a period of 23 days during which the ground was covered by snow, highlighted in grey color in the time series plots on the next page.

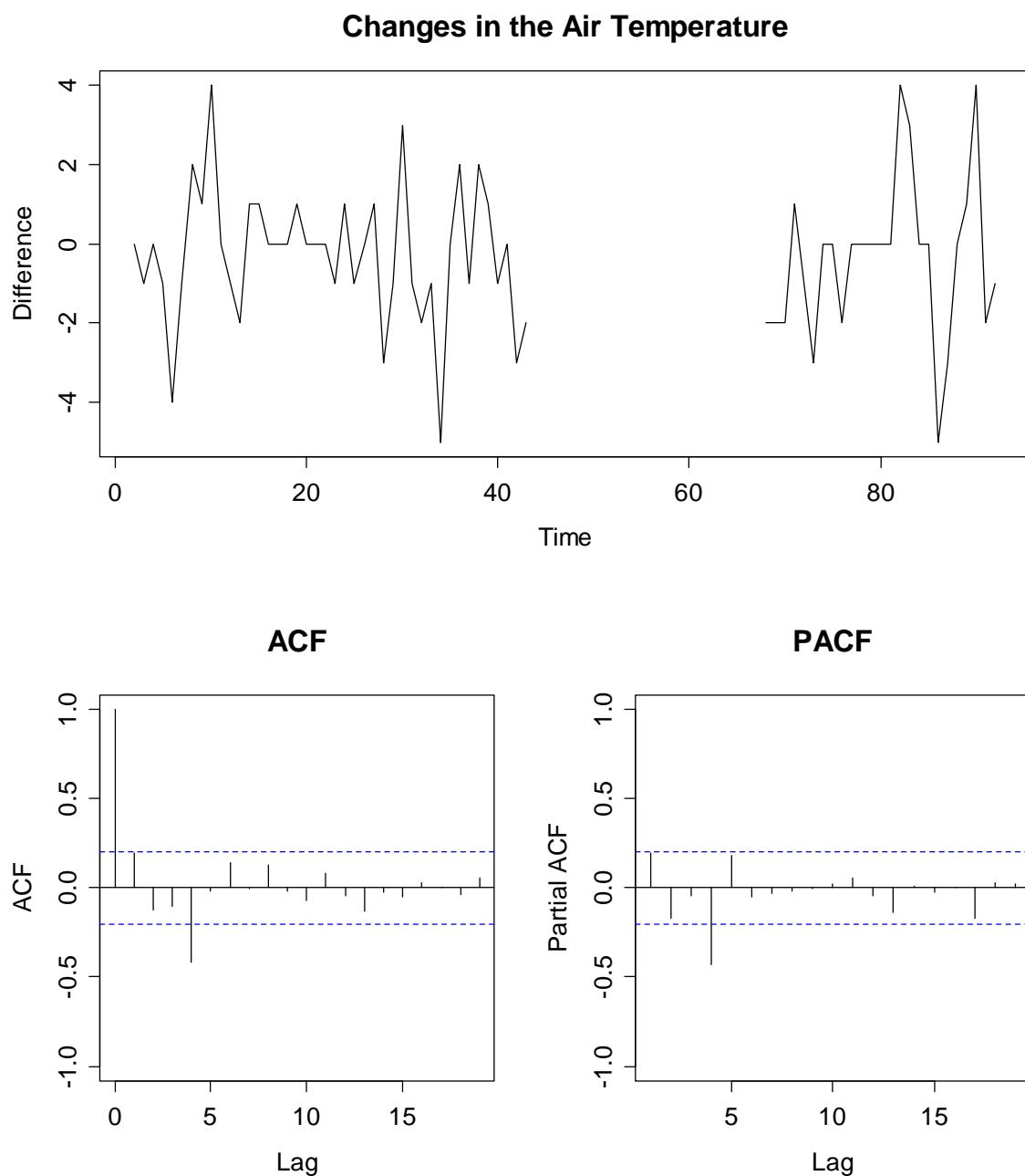
Because the snow insulates the ground, we do not expect the soil to follow the air temperature during that period. Hence, we set all values during that period equal to NA. The time series plots, and especially the indexed plot where both series are shown, clearly indicate that the soil reacts to the air temperature with a delay of a few days. We now aim for analyzing this relationship on a more quantitative basis, for which the methods of multivariate time series analysis will be employed.



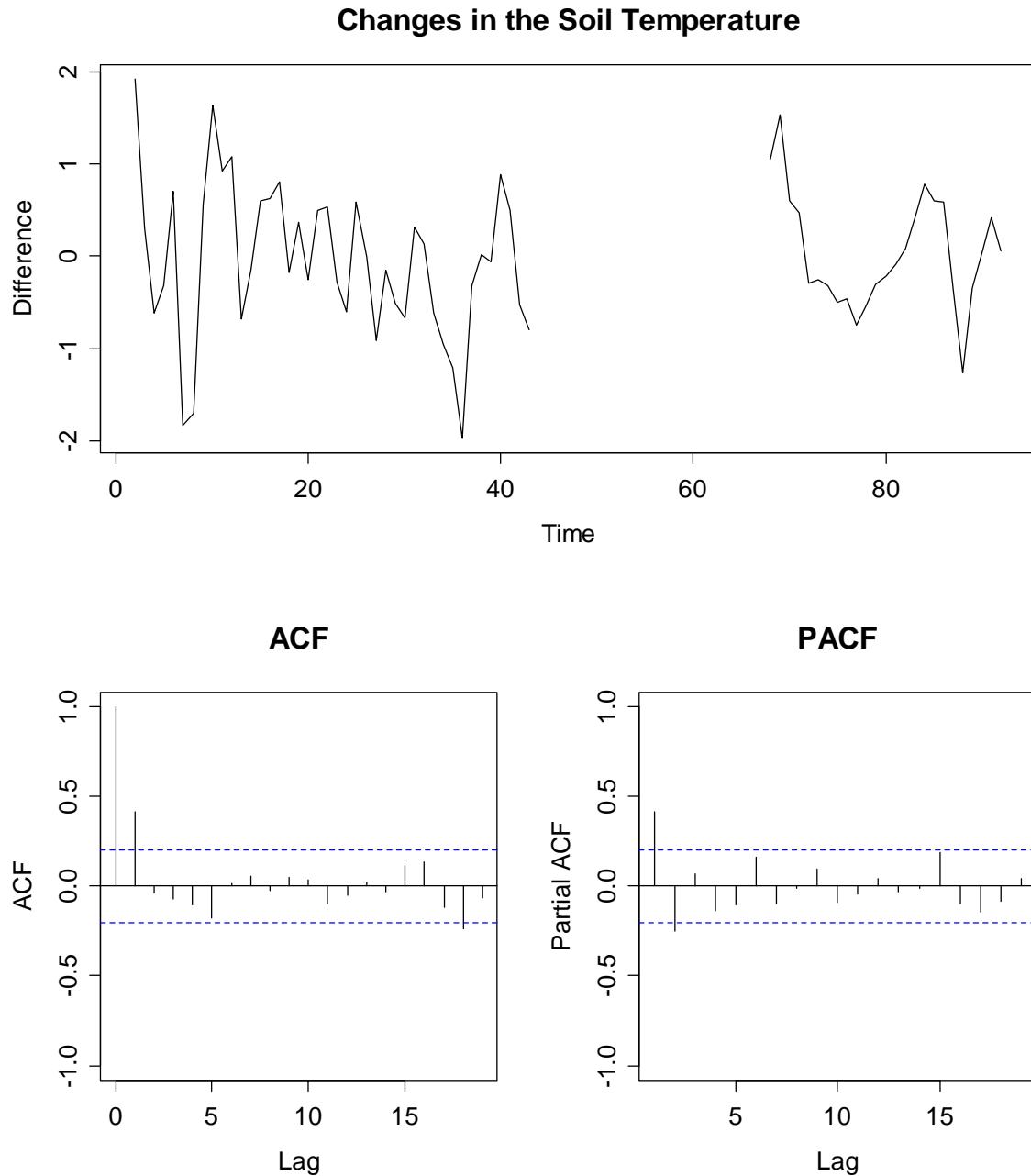
As we had stated above, multivariate time series analysis requires stationarity. Is this met with our series? The time series plot does not give a very clear answer. Science tells us that temperature has a seasonal pattern. Moreover, the correlogram of the two series is enlightening.



The ACF exhibits a slow decay, especially for the soil temperature. Thus, we decide to perform lag 1 differencing before analyzing the series. This has another advantage: we are then exploring how changes in the air temperature are associated with changes in the soil temperature and if so, what the time delay is. These results are easier to interpret than a direct analysis of air and soil temperatures. Next, we display the differenced series with their ACF and PACF. The observations during the snow cover period are now omitted.



The differenced air temperature series seems stationary, but is clearly not iid. There seems to be some strong negative correlation at lag 4. This may indicate the properties of the meteorological weather patterns at that time of year in that part of Switzerland. We now perform the same analysis for the changes in the soil temperature.



9.2 Cross Correlation

To begin with, we consider the (theoretical) cross covariance, the measure that describes the amount of linear dependence between the two time series processes. Firstly, we recall the definition of the within-series autocovariances, denoted by $\gamma_{11}(k)$ and $\gamma_{22}(k)$:

$$\gamma_{11}(k) = \text{Cov}(X_{1,t+k}, X_{1,t}), \quad \gamma_{22}(k) = \text{Cov}(X_{2,t+k}, X_{2,t})$$

The cross covariances between the two processes X_1 and X_2 are given by:

$$\gamma_{12}(k) = \text{Cov}(X_{1,t+k}, X_{2,t}), \quad \gamma_{21}(k) = \text{Cov}(X_{2,t+k}, X_{1,t})$$

Note that owing to the stationarity of the two series, the cross covariances $\gamma_{12}(k)$ and $\gamma_{21}(k)$ both do not depend on the time t . Moreover, there is some obvious symmetry in the cross covariance:

$$\gamma_{12}(-k) = \text{Cov}(X_{1,t-k}, X_{2,t}) = \text{Cov}(X_{1,t}, X_{2,t+k}) = \gamma_{21}(k)$$

Thus, for practical purposes, it suffices to consider $\gamma_{12}(k)$ for positive and negative values of k . Note that we will preferably work with correlations rather than covariances, because they are scale-free and thus easier to interpret. We can obtain the cross correlations by standardizing the cross covariances:

$$\rho_{12}(k) = \frac{\gamma_{12}(k)}{\sqrt{\gamma_{11}(0)\gamma_{22}(0)}}, \quad \rho_{21}(k) = \frac{\gamma_{21}(k)}{\sqrt{\gamma_{11}(0)\gamma_{22}(0)}}.$$

Not surprisingly, we also have symmetry here, i.e. $\rho_{12}(-k) = \rho_{21}(k)$. Additionally, the cross correlations are limited to the interval between -1 and +1, i.e. $|\rho_{12}(k)| \leq 1$. As for the interpretation, $\rho_{12}(k)$ measures the linear association between two values of X_1 and X_2 , if the value of the first time series is k steps ahead. Concerning estimation of cross covariances and cross correlations, we apply the usual sample estimators:

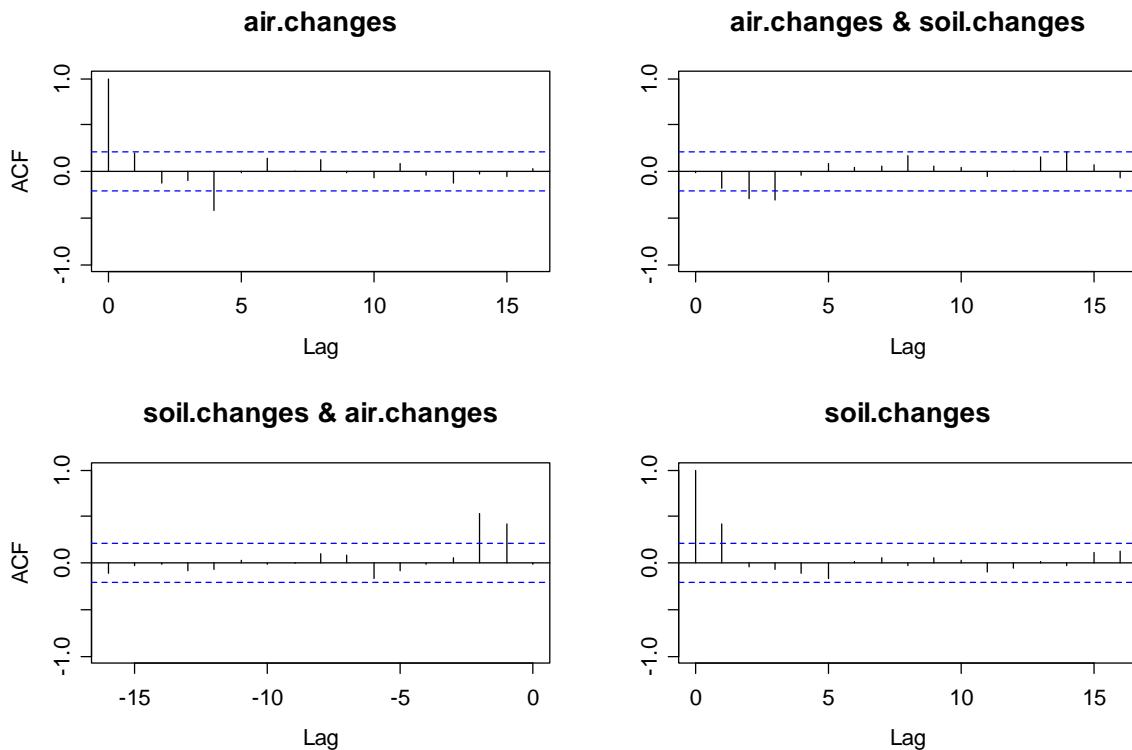
$$\hat{\gamma}_{12}(k) = \frac{1}{n} \sum_t (x_{1,t+k} - \bar{x}_1)(x_{2,t} - \bar{x}_2) \text{ and } \hat{\gamma}_{21}(k) = \frac{1}{n} \sum_t (x_{2,t+k} - \bar{x}_2)(x_{1,t} - \bar{x}_1),$$

where the summation index t for $k \geq 0$ goes from 1 to $n-k$ and for $k < 0$ goes from $1-k$ to n . With \bar{x}_1 and \bar{x}_2 we denote the mean values of $x_{1,t}$ and $x_{2,t}$, respectively. We define the estimation of the cross-correlations as

$$\hat{\rho}_{12}(k) = \frac{\hat{\gamma}_{12}(k)}{\sqrt{\hat{\gamma}_{11}(0)\hat{\gamma}_{22}(0)}}, \quad \hat{\rho}_{21}(k) = \frac{\hat{\gamma}_{21}(k)}{\sqrt{\hat{\gamma}_{11}(0)\hat{\gamma}_{22}(0)}}.$$

The plot of $\hat{\rho}_{12}(k)$ against k is called the cross-correlogram. Note that this must be viewed for both positive and negative k . In R, we the job is done by the acf() function, applied to a multiple time series object.

```
> both <- ts.union(diff(air.na), diff(soil.na))
> acf(both, na.action=na.pass, ylim=c(-1,1))
```



The top left panel shows the ACF of the differenced air temperature, the bottom right one holds the pure autocorrelations of the differenced soil temperature. The two off-diagonal plots contains estimates of the cross correlations: The top right panel has $\hat{\rho}_{12}(k)$ for positive values of k , and thus shows how changes in the air temperature depend on changes in the soil temperature.

Note that we do not expect any significant correlation coefficients here, because the ground temperature has hardly any influence on the future air temperature at all. Conversely, the bottom left panel shows $\hat{\rho}_{12}(k)$ for negative values of k , and thus how the changes in the soil temperature depend on changes in the air temperature. Here, we expect to see significant correlation.

9.2.1 Interpreting the Cross Correlogram

Interpreting the cross correlogram is tricky, because the within-series dependency results in a mixing of the correlations. It is very important to note that the confidence bounds shown in the above plots are usually wrong and can thus be strongly misleading. If not the additional steps to be discussed below are taken, interpreting the raw cross correlograms will lead to false conclusions.

The reason for these problems is that the variances and covariances of the $\hat{\rho}_{12}(k)$ are very complicated functions of $\rho_{11}(j), \rho_{22}(j)$ and $\rho_{12}(j), j \in \mathbb{Z}$. For illustrative purposes, we will treat some special cases explicitly.

Case 1: No correlation between the two series for large lags

In the case where the cross correlation $\rho_{12}(j) = 0$ for $|j| \geq m$, we have for $|k| \geq m$:

$$\text{Var}(\hat{\rho}_{12}(k)) \approx \frac{1}{n} \sum_{j=-\infty}^{\infty} \{\rho_{11}(j)\rho_{22}(j) + \rho_{12}(j+k)\rho_{12}(j-k)\}.$$

Thus, the variance of the estimated cross correlation coefficients goes to zero for $n \rightarrow \infty$, but for a deeper understanding with finite sample size, we must know all true auto and cross-correlations, which is of course impossible in practice.

Case 2: No correlation between the series for all lags

If the two processes X_1 and X_2 are independent, i.e. $\rho_{12}(j) = 0$ for all j , then the variance of the cross correlation estimator simplifies to:

$$\text{Var}(\hat{\rho}_{12}(k)) \approx \frac{1}{n} \sum_{j=-\infty}^{\infty} \rho_{11}(j)\rho_{22}(j).$$

If, for example, X_1 and X_2 are two independent AR(1) processes with parameters α_1 and α_2 , then $\rho_{11}(j) = \alpha_1^{|j|}$, $\rho_{22}(j) = \alpha_2^{|j|}$ and $\rho_{12}(j) = 0$. For the variance of $\hat{\rho}_{12}(k)$ we have, because the autocorrelations form a geometric series:

$$\text{Var}(\hat{\rho}_{12}(k)) \approx \frac{1}{n} \sum_{j=-\infty}^{\infty} (\alpha_1 \alpha_2)^{|j|} = \frac{1}{n} \frac{1 + \alpha_1 \alpha_2}{1 - \alpha_1 \alpha_2}.$$

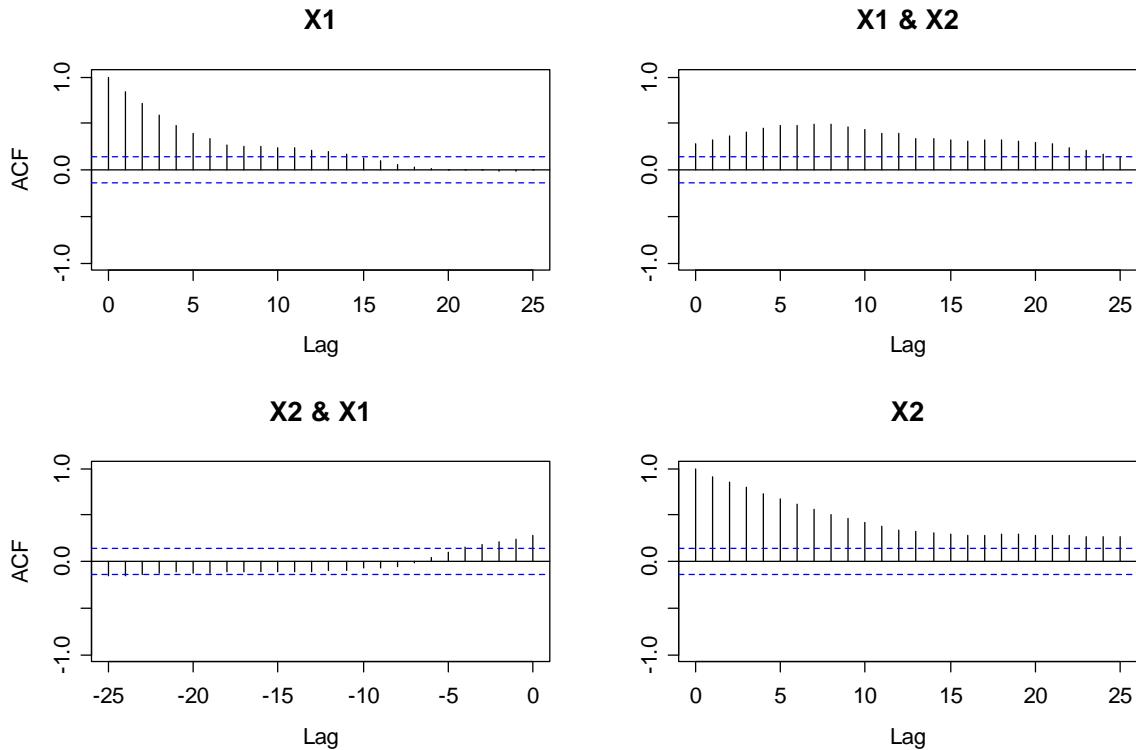
For $\alpha_1 \rightarrow 1$ and $\alpha_2 \rightarrow 1$ this expression goes to ∞ , i.e. the estimator $\hat{\rho}_{12}(k)$ can, for a finite time series, differ greatly from the true value 0. We would like to illustrate this with two simulated AR(1) processes with $\alpha_1 = \alpha_2 = 0.9$. According to theory all cross-correlations are 0. However, as we can see in the figure on the next page, the estimated cross correlations differ greatly from 0, even though the length of the estimated series is 200. In fact, $2\sqrt{\text{Var}(\hat{\rho}_{12}(k))} \approx 0.44$, i.e. the 95% confidence interval is ± 0.44 . Thus even with an estimated cross-correlation of 0.4 the null hypothesis "true cross-correlation is equal to 0" cannot be rejected.

Case 3: No cross correlations for all lags and one series uncorrelated

Only now, in this special case, the variance of the cross correlation estimator is significantly simplified. In particular, if X_1 is a White Noise process which is independent of X_2 , we have, for large n and small k :

$$\text{Var}(\hat{\rho}_{12}(k)) \approx \frac{1}{n}.$$

Thus, in this special case, the rule of thumb $\pm 2/\sqrt{n}$ yields a valid approximation to a 95% confidence interval for the cross correlations and can help to decide whether they are significantly or just randomly different from zero.



In most practical examples, however, the data will be auto- and also cross correlated. Thus, the question arises whether it is at all possible to do something here. Fortunately, the answer is yes: with the method of *prewhitening*, described in the next chapter, we do obtain a theoretically sound and practically useful cross correlation analysis.

9.3 Prewhitening

The idea behind *prewhitening* is to transform one of the two series such that it is uncorrelated, i.e. a White Noise series, which also explains the name of the approach. Formally, we assume that the two stationary processes X_1 and X_2 can be transformed as follows:

$$U_t = \sum_{i=0}^{\infty} a_i X_{1,t-i}$$

$$V_t = \sum_{i=0}^{\infty} b_i X_{2,t-i}$$

Thus, we are after coefficients a_i and b_i such that an infinite linear combination of past terms leads to White Noise. We know from previous theory that such a representation exists for all stationary and invertible $ARMA(p,q)$ processes, it is the $AR(\infty)$ representation. For the cross-correlations between U_t and V_t and between X_t and Y_t , the following relation holds:

$$\rho_{UV}(k) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_i b_j \rho_{X_1 X_2}(k+i-j)$$

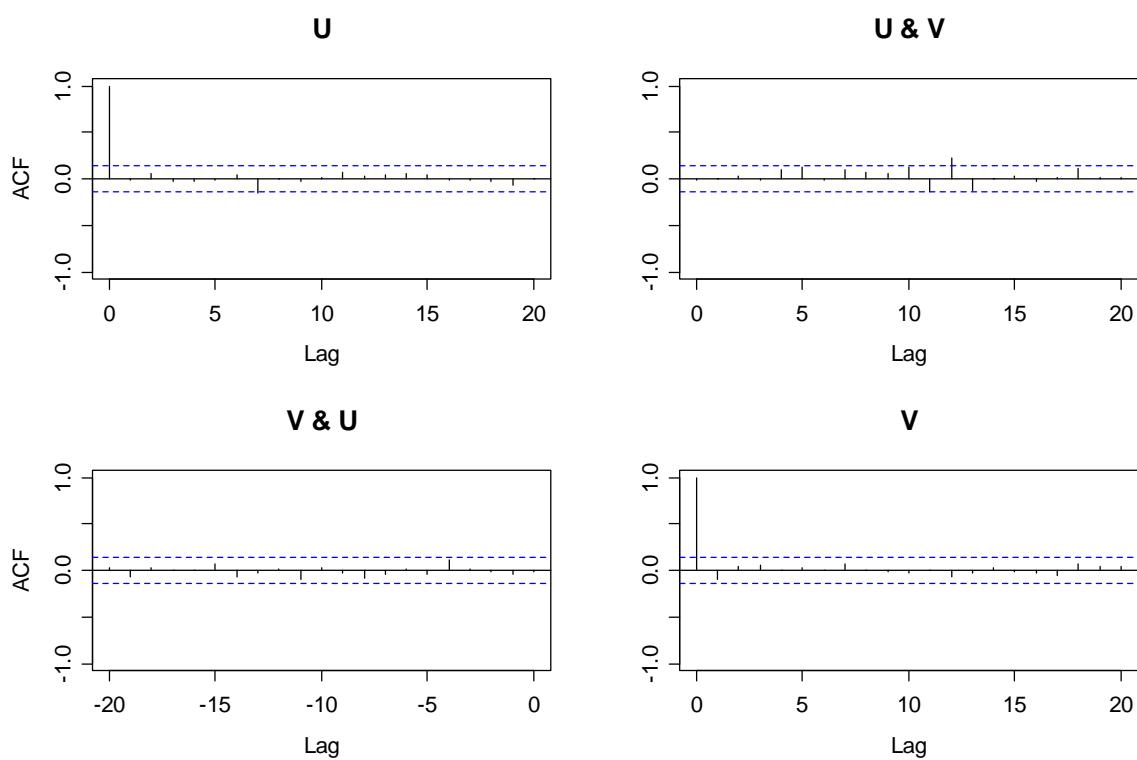
We conjecture that for two independent processes X_1 and X_2 , where all cross correlation coefficients $\rho_{X_1 X_2}(k)=0$, also all $\rho_{UV}(k)=0$. Additionally, the converse is also true, i.e. it follows from “ U_t and V_t uncorrelated” that the original processes X_1 and X_2 are uncorrelated, too. Since U_t and V_t are White Noise processes, we are in the above explained case 3, and thus the confidence bounds in the cross correlograms are valid. Hence, any cross correlation analysis on “real” time series starts with representing them in terms of u_t and v_t .

Example: AR(1) Simulations

For our example with the two simulated $AR(1)$ processes, we can estimate the AR model coefficients with the Burg method and plug them in for prewhitening the series. Note that this amounts considering the residuals from the two fitted models!

$$u_t = x_{1,t} - \hat{\alpha}_1 x_{1,t-1}, \text{ where } \hat{\alpha}_1 = 0.889, \text{ and}$$

$$v_t = x_{2,t} - \hat{\alpha}_2 x_{2,t-1}, \text{ where } \hat{\alpha}_2 = 0.917.$$



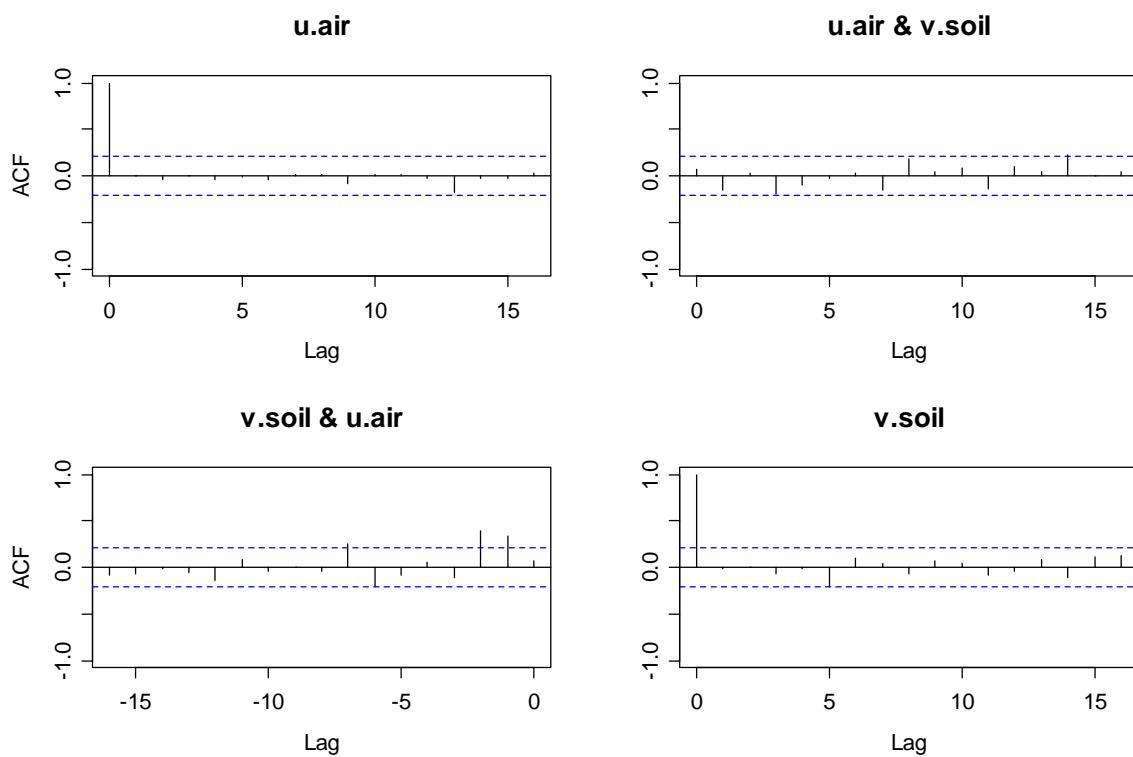
The figure on the previous page shows both the auto and cross correlations of the prewhitened series. We emphasize again that we here consider the residuals from the $AR(1)$ models that were fitted to series X_1 and X_2 . We observe that, as we expect, there are no significant autocorrelations, and there is just one cross correlation coefficient that exceeds the 95% confidence bounds. We can attribute this to random variation.

The theory suggests, because U_t and V_t are uncorrelated, that also X_1 and X_2 do not show any linear dependence. Well, owing to how we set up the simulation, we know this for a fact, and take the result as evidence that the prewhitening approach works in practice.

Example: Air and Soil Temperatures

For verifying whether there is any cross correlation between the changes in air and soil temperatures, we have to perform prewhitening also for the two differenced series. Previously, we had identified an $AR(5)$ and a $MA(1)$ model as. We can now just take their residuals and perform a cross correlation analysis:

```
> fit.air <- arima(diff(air.na), order=c(5,0,0))
> fit.soil <- arima(diff(soil.na), order=c(0,0,1))
> u.air <- resid(fit.air); v.soil <- resid(fit.soil)
> acf(ts.union(u.air, v.soil), na.action=na.pass)
```



The bottom left panel shows some significant cross correlations. A change in the air temperature seems to induce a change in the ground with a lag of 1 or 2 days.

9.4 Transfer Function Models

In the previous section we had observed significant cross correlations between the prewhitened air and soil temperature changes. This means that the cross correlations between the original air and soil temperature changes will also be different from zero. However, due to the prewhitening, inferring the magnitude of the linear association is different. The aim of this section is to clarify this issue.

The transfer function models are a possible way to capture the dependency between two time series. We must assume that the first series influences the second, but the second does not influence the first. Furthermore, the influence occurs only at simultaneously or in the future, but not on past values. Both assumptions are met in our example. The transfer function model is:

$$X_{2,t} - \mu_2 = \sum_{j=0}^{\infty} \nu_j (X_{1,t-j} - \mu_1) + E_t$$

We call X_1 the *input* and correspondingly, X_2 is named the *output*. For the error term E_t we require zero expectation and that they are independent from the input series, in particular:

$$E[E_t] = 0 \text{ and } \text{Cov}(E_t, X_{1,s}) = 0 \text{ for all } t \text{ and } s.$$

However, the errors E_t are usually autocorrelated. Note that this model is very similar to the time series regression model. However, here we have infinitely many unknown coefficients ν_j , i.e. we do not know (a priori) on which lags to regress the input for obtaining the output. For the following theory, we assume (w.l.o.g.) that $\mu_1 = \mu_2 = 0$, i.e. the two series were adjusted for their means. In this case the cross covariances $\gamma_{21}(k)$ are given by:

$$\gamma_{21}(k) = \text{Cov}(X_{2,t+k}, X_{1,t}) = \text{Cov}\left(\sum_{j=0}^{\infty} \nu_j X_{1,t+k-j}, X_{1,t}\right) = \sum_{j=0}^{\infty} \nu_j \gamma_{11}(k-j).$$

In cases where the transfer function model has a finite number of coefficients ν_j only, i.e. $\nu_j = 0$ for $j > K$, then the above formula turns into a linear system of $K+1$ equations that we could theoretically solve for the unknowns $\nu_j, j = 0, \dots, K$.

If we replaced the theoretical γ_{11} and γ_{21} by the empirical covariances $\hat{\gamma}_{11}$ and $\hat{\gamma}_{21}$, this would yield, estimates $\hat{\nu}_j$. However, this method is statistically inefficient and the choice of K proves to be difficult in practice. We again resort to some special case, for which the relation between cross covariance and transfer function model coefficients simplifies drastically.

Special Case: Uncorrelated input series X_1

In this case, $\gamma_{11}(k) = 0$ for $k \neq 0$ and we have $\gamma_{21}(k) = \nu_k \gamma_{11}(0)$. For the coefficients ν_k this results in the simplified transfer function model:

$$\nu_k = \frac{\gamma_{21}(k)}{\gamma_{11}(0)} = \rho_{21} \sqrt{\frac{\gamma_{22}(0)}{\gamma_{11}(0)}}, \text{ for } k \geq 0.$$

However, X_1 generally is not a White Noise process. We can resort to prewhitening the input series. As we will show below, we can obtain an equivalent transfer function model with identical coefficients if a smart transformation is

applied to the output series. Namely, we have to filter the output with the model coefficients from the input series.

$$X_{1,t} = 0.296 \cdot X_{1,t-1} - 0.242 \cdot X_{1,t-2} - 0.119 \cdot X_{1,t-3} - 0.497 \cdot X_{1,t-4} + 0.216 \cdot X_{1,t-5} + D_t,$$

where D_t is the innovation, i.e. a White Noise process, for which we estimate the variance to be $\hat{\sigma}_D^2 = 2.392$. We now solve this equation for D_t and get:

$$\begin{aligned} D_t &= X_{1,t} - 0.296 \cdot X_{1,t-1} + 0.242 \cdot X_{1,t-2} + 0.119 \cdot X_{1,t-3} + 0.497 \cdot X_{1,t-4} - 0.216 \cdot X_{1,t-5} \\ &= (1 - 0.296B + 0.242B^2 + 0.119B^3 + 0.497B^4 - 0.216B^5)X_{1,t} \end{aligned}$$

We now apply this same transformation, i.e. the characteristic polynomial of the AR(5) also on the output series X_2 and the transfer function model errors E_t :

$$Z_t = (1 - 0.296B + 0.242B^2 + 0.119B^3 + 0.497B^4 - 0.216B^5)X_{2,t}$$

$$U_t = (1 - 0.296B + 0.242B^2 + 0.119B^3 + 0.497B^4 - 0.216B^5)E_t.$$

We can now equivalently write the transfer function model with the new processes D_t , Z_t and U_t . It takes the form:

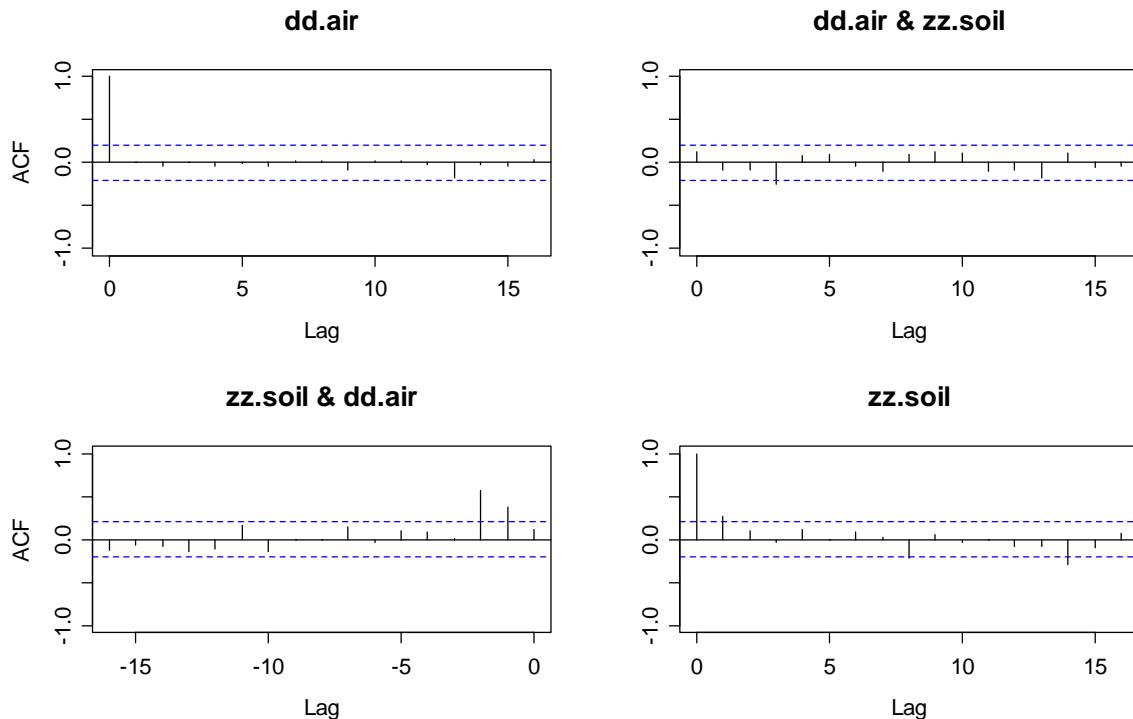
$$Z_t = \sum_{j=0}^{\infty} \nu_j D_{t-j} + U_t,$$

where the coefficients ν_j are identical than for the previous formulation of the model. The advantage of this latest formulation, however, is that the input series D_t is now White Noise, such that the above special case applies, and the transfer function model coefficients can be obtained by a straightforward computation from the cross correlations:

$$\hat{\nu}_k = \frac{\hat{\gamma}_{21}(k)}{\hat{\sigma}_D^2} = \frac{\hat{\sigma}_Z}{\hat{\sigma}_D} \hat{\rho}_{21}(k), \text{ where } k \geq 0.$$

where $\hat{\gamma}_{21}$ and $\hat{\rho}_{21}$ denote the empirical cross covariances and cross correlations of D_t and Z_t . However, keep in mind that Z_t and U_t are generally correlated. Thus, the outlined method is not a statistically efficient estimator either. While efficient approaches exist, we will not discuss them in this course and scriptum. Furthermore, for practical application the outlined procedure usually yields reliable results. We conclude this section by showing the results for the permafrost example: the transfer function model coefficients in the example are based on the cross correlation between the AR(5) residuals of the air changes and the ground changes that had been filtered with these AR(5) coefficients.

```
> dd.air <- resid(fit.air)
> coefs <- coef(fit.air)[1:5]
> zz.soil <- filter(diff(soil.na), c(1, -coefs, sides=1))
> as.int <- ts.intersect(dd.air, zz.soil)
> acf.val <- acf(as.int, na.action=na.pass)
```



Again, in all except for the bottom left panel, the correlation coefficients are mostly zero, respectively only insignificantly or by chance different from that value. This is different in the bottom left panel. Here, we have substantial cross correlation at lags 1 and 2. Also, these values are proportional to the transfer function model coefficients. We can extract these as follows:

```
> multip <- sd(zz.soil, na.rm=TRUE)/sd(dd.air, na.rm=TRUE)
> multip*acf.val$acf[,2,1]
[1]  0.054305137  0.165729551  0.250648114  0.008416697
[5]  0.036091971  0.042582917 -0.014780751  0.065008411
[9] -0.002900099 -0.001487220 -0.062670672  0.073479065
[13] -0.049352348 -0.060899602 -0.032943583 -0.025975790
```

Thus, the soil temperature in the permafrost boreholes reacts to air temperature changes with a delay of 1-2 days. An analysis of further boreholes has suggested that the delay depends on the type of terrain in which the measurements were made. Fastest response times are found for a very coarse-blocky rock glacier site, whereas slower response times are revealed for blocky scree slopes with smaller grain sizes.

10 Spectral Analysis

During this course, we have encountered several time series which show periodic behavior. Prominent examples include the number of shot lynx in the Mackenzie River district in Canada, as well as the wave tank data from section 4.4. In these series, the periodicity is not deterministic, but stochastic. An important goal is to understand the cycles at which highs and lows in the data appear.

In this chapter, we will introduce *spectral analysis* as a descriptive means for showing the character of, and the dependency structure within a time series. This will be based on interpreting the series as a superposition of cyclic components, namely as a *linear combination of harmonic oscillations*. We will introduce the *periodogram*, where the aim is to show which frequencies contribute most importantly to the variation in the series.

In spirit, such an analysis is related to the correlogram. In fact, one can show that the information in the correlogram and the periodogram are mathematically equivalent. However, the two approaches still provide different, complementary views on a series and it is thus often worthwhile to pursue both approaches. Finally, we here also mention that in some areas time series are preferably analyzed in the time domain, whereas in other applied fields, e.g. electrical engineering, geophysics and econometrics, the frequency approach predominates.

10.1 Decomposing in the Frequency Domain

We will here first introduce some background and theory on how to decompose time series into cyclic components and then lay the focus on the efficient estimation of these.

10.1.1 Harmonic Oscillations

The simplest and best known periodic functions are sine and cosine. It is thus appealing to use these as a basis for decomposing time series. A harmonic oscillation is of the form

$$y(t) = a \cdot \cos(2\pi\nu t - \phi).$$

Here, we call a the amplitude, ν is the frequency and ϕ is the phase. Apparently, the function $y(t)$ is periodic, and the period is $T = 1/\nu$. It is common to write the above harmonic oscillation in a different form, i.e.:

$$y(t) = \alpha \cdot \cos(2\pi\nu t) + \beta \cdot \sin(2\pi\nu t),$$

where in fact $\alpha = a \cos(\phi)$ and $\beta = a \sin(\phi)$. The advantage of this latter form is that if we want to fit a harmonic oscillation with fixed frequency to data, which means estimating amplitude and phase, we face a linear problem instead of a non-linear one, as it was the case in the previous formulation. The time can be either continuous or discrete. In the context of our analysis of discrete time series, only the latter will be relevant.

Now, if fitting a harmonic oscillation to discrete data, we face an identification problem: If frequency ν fits, then all higher frequencies such as $\nu+1, \nu+2, \dots$ will fit as well. This phenomenon is known as aliasing. The plot below shows harmonics where $a=1$ and $\phi=0$. As frequencies, we choose both $\nu=1/6$ and $\nu=1+1/6$. We observe that we cannot decide upon which of the two frequencies generated our discrete time observations. Naturally, the time resolution of our series determines which frequencies we can identify. Or more clearly: we take the point that our data do not allow to identify periodicities with frequency $\nu > 1/2$, i.e. that harmonics which oscillate more than once between two observations.

10.1.2 Superposition of Harmonics

In a real-world stationary time series, it is rare to nonexistent that only one single periodicity that can be attributed to a single frequency makes up for all the variation that is observed. Thus, for a decomposition of the series into a number of periodicities with different frequency, we choose the regression approach:

$$X_t = \alpha_0 + \sum_{k=1}^m (\alpha_k \cos(2\pi\nu_k t) + \beta_k \sin(2\pi\nu_k t)) + E_t,$$

where α_k, β_k are interpreted as the unknown parameters, E_t is an *iid* error term with expectation zero and ν_1, \dots, ν_m is a set of pre-defined frequencies. Under these assumptions, we can obtain estimates $\hat{\alpha}_k, \hat{\beta}_k$ with the ordinary least squares algorithm. As for the frequencies, we choose multiples of $1/n$, i.e.

$$\nu_k = k/n, \text{ for } k=1, \dots, m \text{ with } m = \lfloor n/2 \rfloor.$$

These are called the *Fourier frequencies*. Using some mathematics, one can prove that the above regression problem has an orthogonal design. Thus, the estimated coefficients $\hat{\alpha}_k, \hat{\beta}_k$ are uncorrelated and (for $k > 0$) have variance $2\sigma_E^2/2$. Because we are also spending n parameters for the n observations, the frequency decomposition model fits perfectly, i.e. all residuals are zero. Another very important result is that the

$$\text{sum of squared residuals } \sum_{i=1}^n r_i^2 \text{ increases by } \frac{n}{2}(\hat{\alpha}_k^2 + \hat{\beta}_k^2)$$

if the frequency ν_k is omitted from the model. We can use this property to gauge the prominence of a particular frequency in the decomposition model, and exactly that is the aim with the periodogram which will be discussed below.

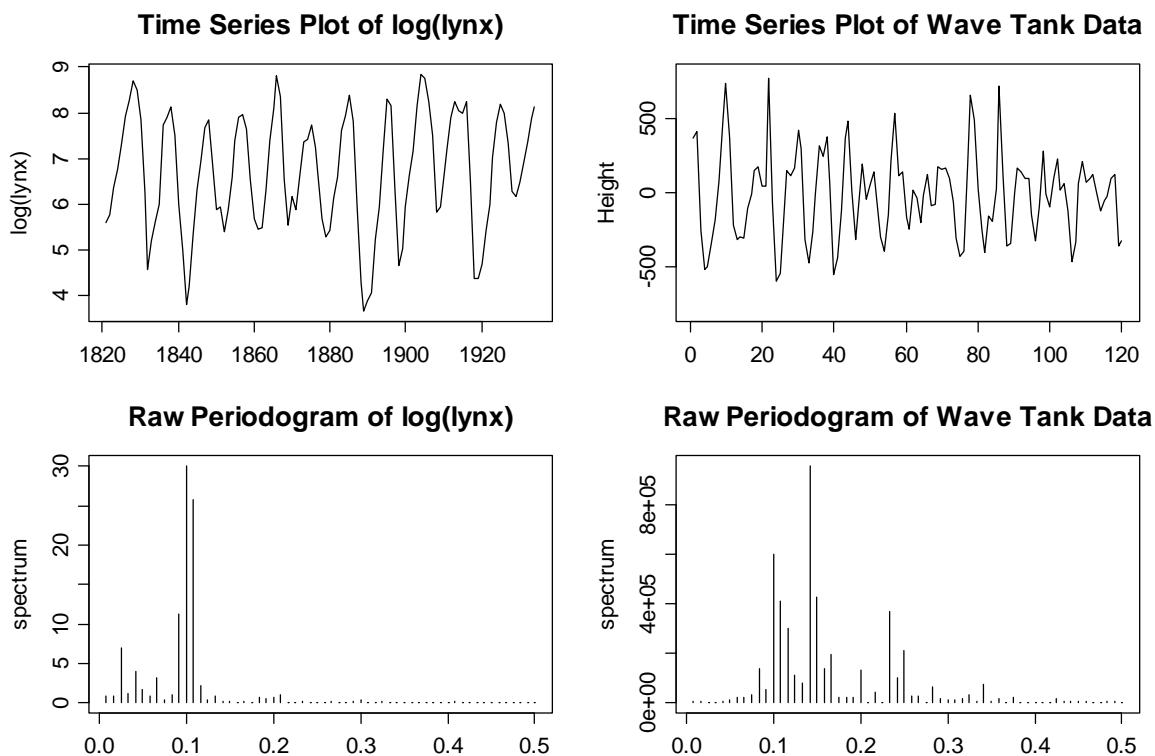
10.1.3 The Periodogram

The periodogram quantifies the presence of periodicities in a time series. It is based on half of the increase in sum of squared residuals in the decomposition model if a particular frequency is omitted. We can rewrite that directly as a function of the observations:

$$\begin{aligned} I_n(\nu_k) &= \frac{n}{4}(\hat{\alpha}_k^2 + \hat{\beta}_k^2) \\ &= \frac{1}{n} \left(\sum_{t=1}^n x_t \cos(2\pi\nu_k t) \right)^2 + \frac{1}{n} \left(\sum_{t=1}^n x_t \sin(2\pi\nu_k t) \right)^2 \end{aligned}$$

The result is then plotted versus the frequency ν_k , and this is known as the raw periodogram. In R, we can use the convenient function `spec.pgram()`. We illustrate its use with the lynx and the wave tank data:

```
> spec.pgram(log(lynx), log="no", type="h")
> spec.pgram(wave, log="no", type="h")
```



The periodogram of the logged lynx data is easy to read: the most prominent frequencies in this series with 114 observations are the ones near 0.1, more exactly, these are $\nu_{11} = 11/114 = 0.096$ and $\nu_{12} = 12/114 = 0.105$. The period of these frequencies is $1/\nu_k$ and thus, $114/11 = 10.36$ and $114/12 = 9.50$. This suggests that the series shows a peak at around every 10th observation which is clearly the case in practice. We can also say that the highs/lows appear between 11 and 12 times in the series. Also this can easily be verified in the time series plot.

Then, there is a secondary peak at $\nu_3 = 3/114$. This must be a cyclic component that appears three times in our data, and the period is $114/3 = 38$. Thus, these are the 40-year-superhighs and -lows that we had identified already earlier.

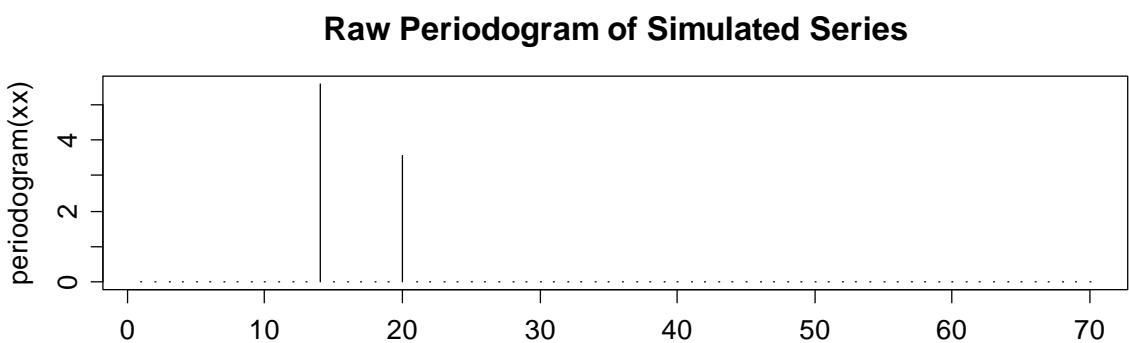
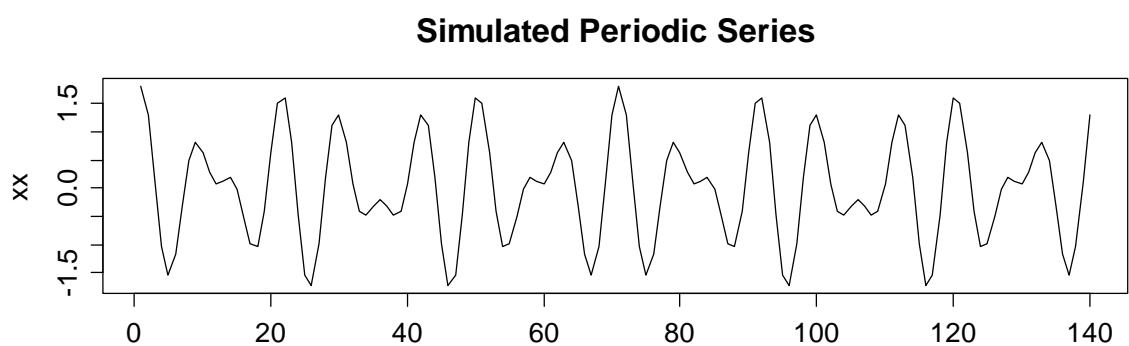
For the wave tank data, we here consider the first 120 observations only. The periodogram is not as clean as for the logged lynx data, but we will try with an interpretation, too. The most prominent peaks are at $k = 12, 17$ and 30 . Thus we have a superposition of cycles which last 4, 7 and 10 observations. The verification is left to you.

10.1.4 Leakage

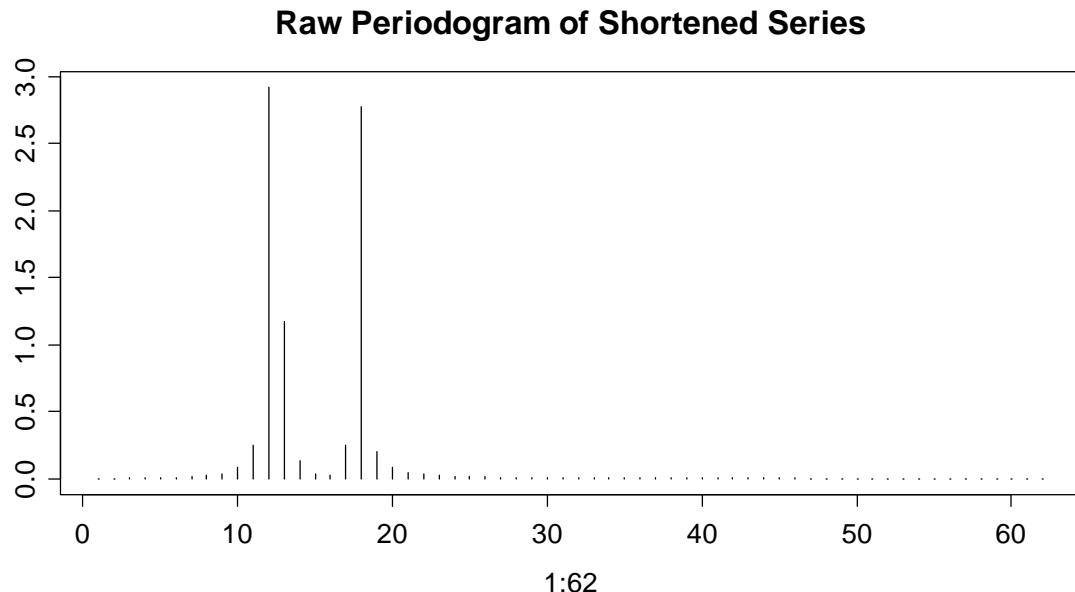
While some basic inspections of the periodogram can and sometimes do already provide valuable insight, there are a few issues which need to be taken care of. The first one which is discussed here is the phenomenon called *leakage*. It appears if there is no Fourier frequency that corresponds to the true periodicity in the data. Usually, the periodogram then shows higher values in the vicinity of the true frequency. The following simulation example is enlightening:

$$X_t = \cos\left(\frac{2\pi \cdot 13 \cdot t}{140}\right) + 0.8 \cdot \cos\left(\frac{2\pi \cdot 20 \cdot t}{140}\right), \text{ for } t = 0, \dots, 139$$

We have a series of 140 observations which is made up as the superposition of two harmonic oscillations with frequencies $13/140$ and $20/140$. These correspond to periods of 7.00 and 10.77, and both are Fourier frequencies. We display the time series plot, as well as the periodogram:



Now if we shorten this very same series by 16 data points so that 124 observations remain, the true frequencies $20/140$ and $13/140$ do no longer appear in the decomposition model, i.e. are not Fourier frequencies anymore. The periodogram now shows leakage:



If not all of the true frequencies in the data generating model are Fourier frequencies, then, $\hat{\alpha}_k, \hat{\beta}_k$ from the decomposition model are only approximations to the true contribution of a particular frequency for the variation in the series.

11 State Space Models

State space modeling is a very flexible tool that from which one can benefit in almost all applied fields. It would certainly merit an own full course, exclusively focusing on its applied aspects. Due to restrictions in time, we here provide only a small overview on the potential and a few examples. While one can write most time series models in state space formulation, it is usually simpler to do without, and use their own genuine notation. The real benefits of state space models only come into play when one has to deal with observations that are blurred with additional measurement noise, or in situations, where some parameters are required to adapt over time.

We will here first introduce the general formulation of state space models, and then illustrate with a number of examples. The first two concern AR processes with additional observation noise, then there are two regressions with time-varying coefficients, and finally we consider a linear growth model. The conditional means and variances of the state vectors are usually estimated with the Kalman Filter. Because this is mathematically rather complex topic, and not in the primary focus of the applied user, this scriptum only provides the coarse rationale for it.

11.1 State Space Formulation

State space models are built on two equations. One is the state equation, and the other is the observation equation. We here introduce the general notation; their meaning will become clearer with the examples discussed below.

State Equation

The state of the system at time t is represented by a column vector X_t , and it is a linear transformation of the state at time $t-1$, plus some random vector W_t (system noise) from a multivariate normal distribution. The linear transformation of the state at time $t-1$ is defined with a matrix G_t , and the covariance matrix of the multivariate normal is denoted with w_t .

$$X_t = G_t X_{t-1} + W_t, \text{ where } W_t \sim N(0, w_t)$$

Observation Equation

The observation at time t is denoted by a column vector Y_t that is a linear combination of the states, determined by a matrix F_t , and random variation (measurement noise) from a normal distribution with covariance matrix v_t .

$$Y_t = F_t X_t + V_t, \text{ where } V_t \sim N(0, v_t)$$

Note that in this general formulation, all matrices can be time varying, but in most of our examples, they will be constant. Also, the nomenclature is different depending on the source, but we here adopt the notation of \mathbb{R} .

11.2 AR Processes with Measurement Noise

We are interested in a stochastic process X_t (which may be an AR process). Then, one usually makes measurements to obtain observations, i.e. acquires a realization of the process. So far, we operated under the assumption that the measurements were error-free, i.e. that there was no measurement noise and our data were exact. In many cases where the data are measured using physical devices, this is hardly realistic, and we may rather have realizations of some random variable

$$Y_t = X_t + V_t, \text{ where } V_t \sim N(0, \sigma_V^2).$$

Thus, the realizations of the process of interest, X_t , are latent, i.e. hidden under some random noise. We will now discuss how this issue can be solved in practice.

Example: $AR(1)$

As the simplest example of a state space model, we consider an $AR(1)$ process which is superposed with some additional measurement noise. The *state equation* is as follows:

$$X_t = \alpha_1 X_{t-1} + W_t.$$

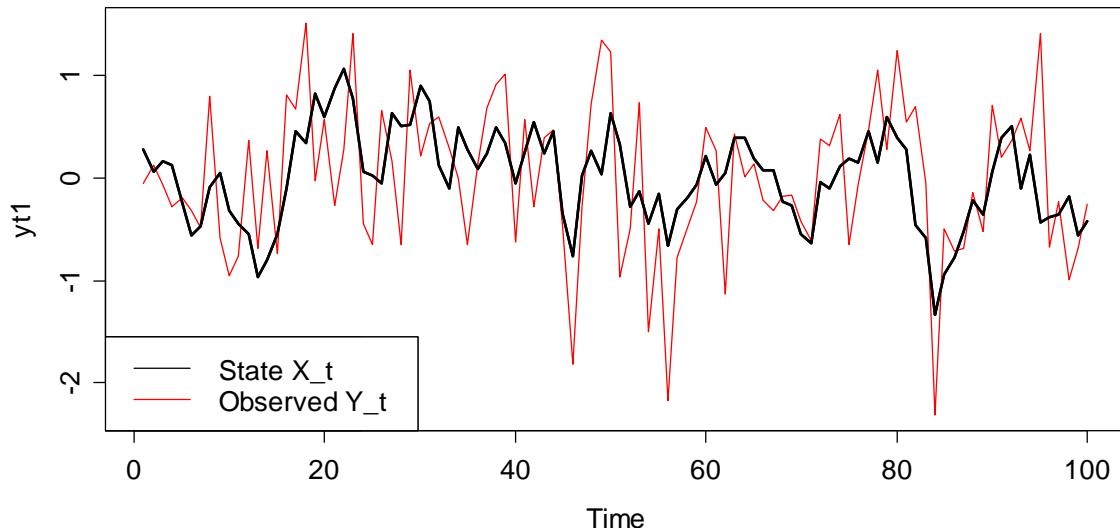
We assume that W_t is an iid innovation with Gaussian distribution, i.e. $W_t \sim N(0, \sigma_W^2)$. Also note that matrix G_t has dimension 1×1 , is time-constant and equal to α_1 . If there is additional measurement noise, our observations can be perceived as realizations of the random variable Y_t :

$$Y_t = X_t + V_t, \text{ where } V_t \sim N(0, \sigma_V^2).$$

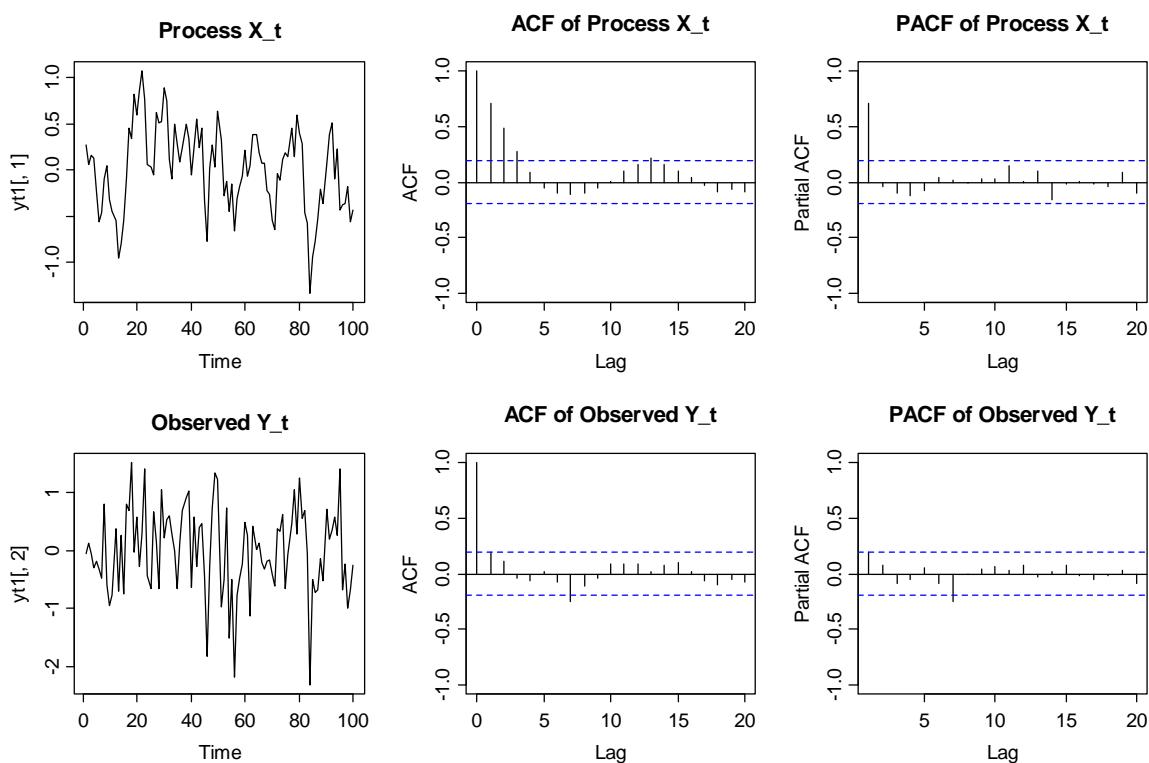
This is the *observation equation*, note that F_t is also time-constant and equal to the 1×1 identity matrix. We here assume that the errors V_t are iid, and also independent of X_s and W_s for all t and s . It is important to note that W_t is the process innovation that impacts future instances X_{t+k} . In contrast, V_t is pure measurement noise with no influence on the future of process X_t . For illustration, we consider a simulation example. We use $\alpha_1 = 0.7$, the innovation variance σ_W^2 is 0.1 and the measurement error variance σ_V^2 is 0.5. The length of the simulated series is 100 observations. On the next page, we show a number of plots. They include a time series plot with both series X_t and Y_t , and the individual plots of X_t with its ACF/PACF, and of Y_t with ACF/PACF. We clearly observe that the appearance of the two processes is very different. While X_t looks like an autoregressive process, and has ACF and PACF showing the stylized facts very prominently, Y_t almost appears to be White Noise. We here know that this is not

true. There is some dependency also in Y_t , but it is blurred by strong noise component, and it seems like a difficult task to recover the weak underlying signal.

AR(1) Simulation Example



We here emphasize that the state space formulation allowed to write a model comprised of a true signal plus additional noise. However, if we face an observed series of this type, we are not any further yet. We need some means to separate the two components. *Kalman filtering*, which is implemented in R package `sspir`, allows doing so: it recovers the (expected value of) the states X_t by some recursive algorithm. For details see below.



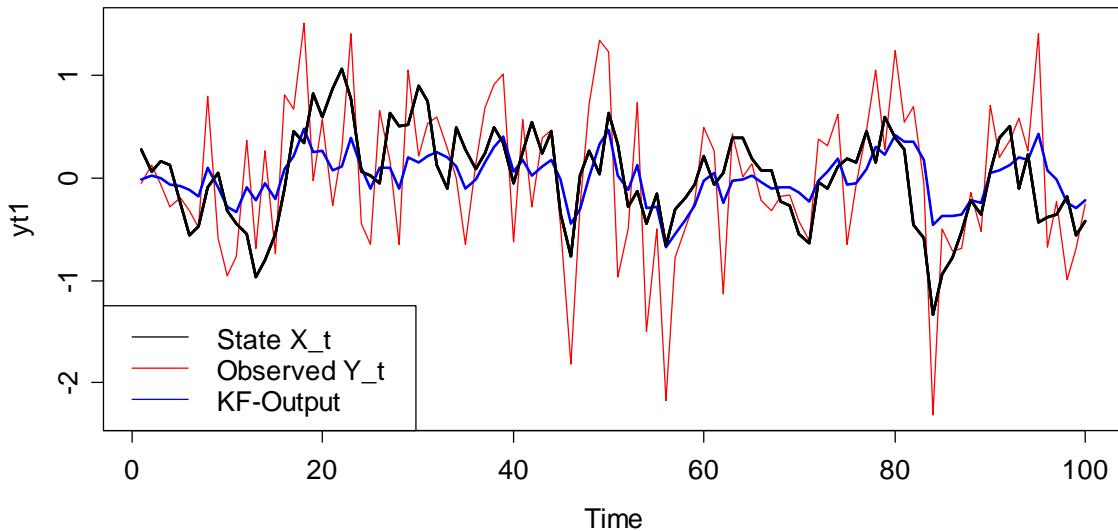
```

> ## Load the package for Kalman filtering
> library(sspir)
>
> ## State Space Formulation
> ssf <- SS(y = as.matrix(obs),
>             Fmat = function(tt,x,phi) {return(matrix(1))},
>             Gmat = function(tt,x,phi) {return(matrix(0.7))},
>             Vmat = function(tt,x,phi) {return(matrix(0.5))},
>             Wmat = function(tt,x,phi) {return(matrix(0.1))},
>             m0 = matrix(0),
>             C0 = matrix(0.1))
>
> ## Kalman Filtering
> fit <- kfilter(ssf)
> plot(fit$m, col="blue", lwd=2, ...)

```

Kalman filtering in **R** requires to specify the state space model first. We need to supply argument y which stands for the observed time series data. They have to come in form of a matrix. Moreover, we have to specify the matrices F_t, G_t , as well as the covariance structures v_t, w_t . In our case, these are all simple 1×1 matrices. Finally, we have to provide m_0 , the starting value of the initial state, and C_0 , the variance of the initial state.

AR(1) Simulation Example with Kalman Filter Output



We can then employ the Kalman filter to recover the original signal X_t . It was added as the blue line in the above plot. While it is not 100% accurate, it still does a very good job of filtering the noise out. However, note that with this simulation example, we have some advantage over the real-life situation. Here, we could specify the correct state space formulation. In practice, we might have problems to identify appropriate values for G_t (the true AR(1) parameter) and the variances in v_t, w_t . On the other hand, in reality the precision of many measurement devices is more or less known, and thus some educated guess is possible.

Example: AR(2)

Here, we demonstrate the formulation of a state space model for an $AR(2)$ process that is superimposed with some measurement error. This example is important, because now, we need to use matrices in the state equation. It is as follows:

$$\begin{pmatrix} X_t \\ X_{t-1} \end{pmatrix} = \begin{pmatrix} \alpha_1 & \alpha_2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} X_{t-1} \\ X_{t-2} \end{pmatrix} + \begin{pmatrix} W_t \\ 0 \end{pmatrix}$$

Apparently, this is a two-dimensional model. The observation equation is:

$$Y_t = (1 \ 0) \cdot \begin{pmatrix} X_t \\ X_{t-1} \end{pmatrix} + V_t$$

Once the equations are set up, it is straightforward to derive the matrices:

$$G_t = G = \begin{pmatrix} \alpha_1 & \alpha_2 \\ 1 & 0 \end{pmatrix}, \quad H_t = H = (1 \ 0), \quad w_t = \begin{pmatrix} \sigma_w^2 & 0 \\ 0 & 0 \end{pmatrix}, \quad v_t = \sigma_v^2$$

Similar to the example above, we could now simulate from an $AR(2)$ process, add some artificial measurement noise and then try to uncover the signal using the Kalman filter. This is left as an exercise.

11.3 Dynamic Linear Models

A specific, but very useful application of state space models is to generalize linear regression such that the coefficients can vary over time. We consider a very simple example where the sales manager in a house building company uses the following model: the company's house sales at time t , denoted as S_t , depends on the general levels of sales in that area L_t and the company's pricing policy P_t .

$$S_t = L_t + \beta_t P_t + V_t$$

This is a linear regression model with price as the predictor, and the general level as the intercept. The assumption is that their influence varies over time, but generally only in small increments. We can use the following notation:

$$L_t = L_{t-1} + \Delta L_t$$

$$\beta_t = \beta_{t-1} + \Delta \beta_t$$

In this model, we assume that v_t , ΔL_t and $\Delta \beta_t$ are random deviations with mean zero that are independent over time. While we assume independence of ΔL_t and $\Delta \beta_t$, we could also allow for correlation among the two. The relative magnitudes of these perturbations are accounted for with the variances in the matrices V_t and W_t of the state space formulation. Note that if we set $W_t = 0$, then we are in the case

of plain OLS regression with constant parameters. Hence, we can also formulate any regression models in state space form. Here, we have:

$$Y_t = S_t, \quad X_t = \begin{pmatrix} L_t \\ \beta_t \end{pmatrix}, \quad W_t = \begin{pmatrix} \Delta L_t \\ \Delta \beta_t \end{pmatrix}, \quad F_t = \begin{pmatrix} 1 \\ P_t \end{pmatrix}, \quad G = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

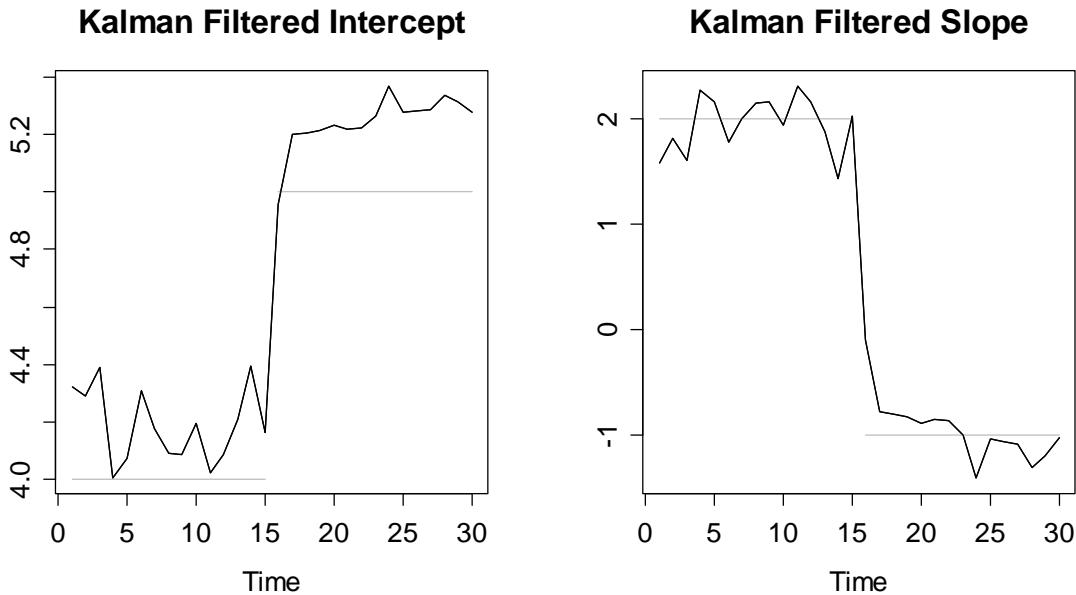
Because we do not have any data for this sales example, we again rely on a simulation. Evidently, this has the advantage that we can evaluate the Kalman filter output versus the truth. Thus, we let

$$y_t = a + bx_t + z_t$$

$$x_t = 2 + t/10$$

We simulate 30 data points from $t=1, \dots, 30$ and assume errors which are standard normally distributed, i.e. $z_t \sim N(0,1)$. The regression coefficients are $a=4$ and $b=2$ for $t=1, \dots, 15$ and $a=5$ and $b=-1$ for $t=16, \dots, 30$. We will fit a straight line with time-varying coefficients, as this is the model that matches what we had found for the sales example above.

```
> ## Simulation
> set.seed(1)
> x1      <- 1:30
> x1      <- x1/10+2
> aa      <- c(rep(4,15), rep( 5,15))
> bb      <- c(rep(2,15), rep(-1,15))
> nn      <- length(x1)
> y1      <- aa+bb*x1+rnorm(nn)
> x0      <- rep(1,nn)
> xx      <- cbind(x0,x1)
> x.mat <- matrix(xx, nrow=nn, ncol=2)
> y.mat <- matrix(y1, nrow=nn, ncol=1)
>
> ## State Space Formulation
> ssf <- SS(y=y.mat, x=x.mat,
>             Fmat=function(tt,x,phi)
>               return(matrix(c(x[tt,1],x[tt,2]),2,1)),
>             Gmat=function(tt,x,phi) return(diag(2)),
>             Wmat=function(tt,x,phi) return(0.1*diag(2)),
>             Vmat=function(tt,x,phi) return(matrix(1)),
>             m0=matrix(c(5,3),1,2),
>             C0=10*diag(2))
>
> ## Kalman-Filtering
> fit <- kfilter(ssf)
> par(mfrow=c(1,2))
> plot(fit$m[,1], type="l", xlab="Time", ylab=" ")
> title("Kalman Filtered Intercept")
> plot(fit$m[,2], type="l", xlab="Time", ylab=" ")
> title("Kalman Filtered Slope")
```



The plots show the Kalman filter output for intercept and slope. The estimates pick up the true values very quickly, even after the change in the regime. It is worth noting that in this example, we had a very clear signal with relatively little noise, and we favored recovering the truth by specifying the state space formulation with the true error variances that are generally unknown in practice.

Example: Batmobile

We here consider another regression problem where time-varying coefficients may be necessary. The description of the practical situation is as follows: *In April 1979 the Albuquerque Police Department began a special enforcement program aimed at countering driving while intoxicated accidents. The program was composed of a squad of police officers, breath alcohol testing (BAT) devices, and a van named batmobile, which housed a BAT device and was used as a mobile station. The data were collected by the Division of Governmental Research of the University of New Mexico under a contract with the National Highway Traffic Safety Administration of the Department of Transportation to evaluate the batmobile program.* Source: <http://lib.stat.cmu.edu/DASL/Datafiles/batdat.html>

The data comprise of quarterly figures of traffic accidents and the fuel consumption in the Albuquerque area as a proxy of the driven mileage. The first 29 quarters are the control period, and observations 30 to 52 were recorded during the experimental (batmobile) period. We would naturally assume a time series regression model for the number of accidents:

$$ACC_t = \beta_0 + \beta_1 \cdot FUEL_t + \beta_2 \cdot 1_{Q_2(t)} + \beta_3 \cdot 1_{Q_3(t)} + \beta_4 \cdot 1_{Q_4(t)} + E_t$$

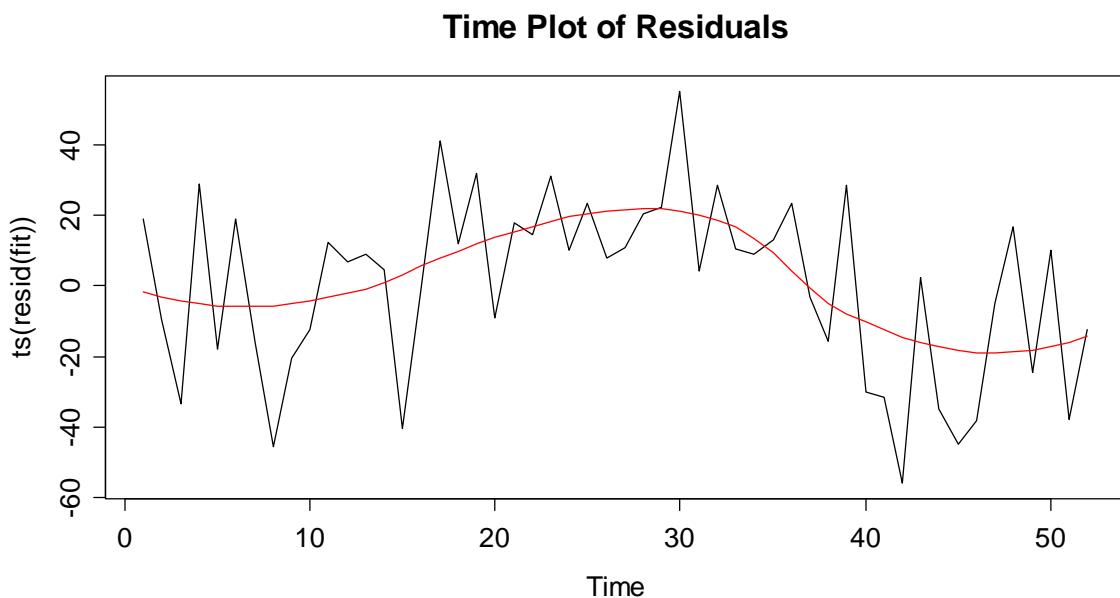
The accidents depend on the mileage driven and there is a seasonal effect. In the above model the intercept β_0 is assumed to be constant. In the light of the BAT program, we might well replace it by $\beta_0 = L_t$, i.e. some general level of accidents that is time-dependent. Let us first perform regression and check residuals.

```

> ## Regression and Time Plot of Residuals
> fit <- lm(ACC ~ FUEL + season, data=regdat)
> plot(ts(resid(fit)), main="Time Plot of Residuals")
>
> ## Fitting a Loess Smoother to the Residuals
> times      <- 1:52
> fit.loess <- loess(resid(fit) ~ times, span=0.65, degree=2)
> lines(times, fitted(fit.loess), col="red")

```

The time series plot of the residuals shows very clear evidence that there is timely dependence. In contrast to what the regression model with constant coefficients suggests, the level of accidents seems to rise in the control period, then drops markedly after the BAT program was introduced. The conclusion is that our above regression model is not adequate and needs to be enhanced. However, just adding an indicator variable that codes for the times before and after the introduction of the BAT program will not solve the issue. It is evident that the level of the residuals is not constant before the program started, and it does not suddenly drop to a constant lower level thereafter.



The alternative is to formulate a state space model and estimate it with the Kalman filter. We (conceptually) assume that all regression parameters are time dependent, and rewrite the model as:

$$ACC_t = L_t + \beta_{1t} \cdot FUEL_t + \beta_{2t} \cdot 1_{Q_2(t)} + \beta_{3t} \cdot 1_{Q_3(t)} + \beta_{4t} \cdot 1_{Q_4(t)} + E_t$$

Our main interest lies in the estimation of the modified intercept term L_t , which we now call the level. We expect it to drop after the introduction of the BAT program, but let's see if this materializes. The state vector X_t we are using contains the regression coefficients, and the state equation which describes their evolution over time is as follows:

$$X_t = GX_{t-1} + W_t, \text{ where } X_t = \begin{pmatrix} L_t \\ \beta_{1t} \\ \beta_{2t} \\ \beta_{3t} \\ \beta_{4t} \end{pmatrix}, G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, w_t = \begin{pmatrix} 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

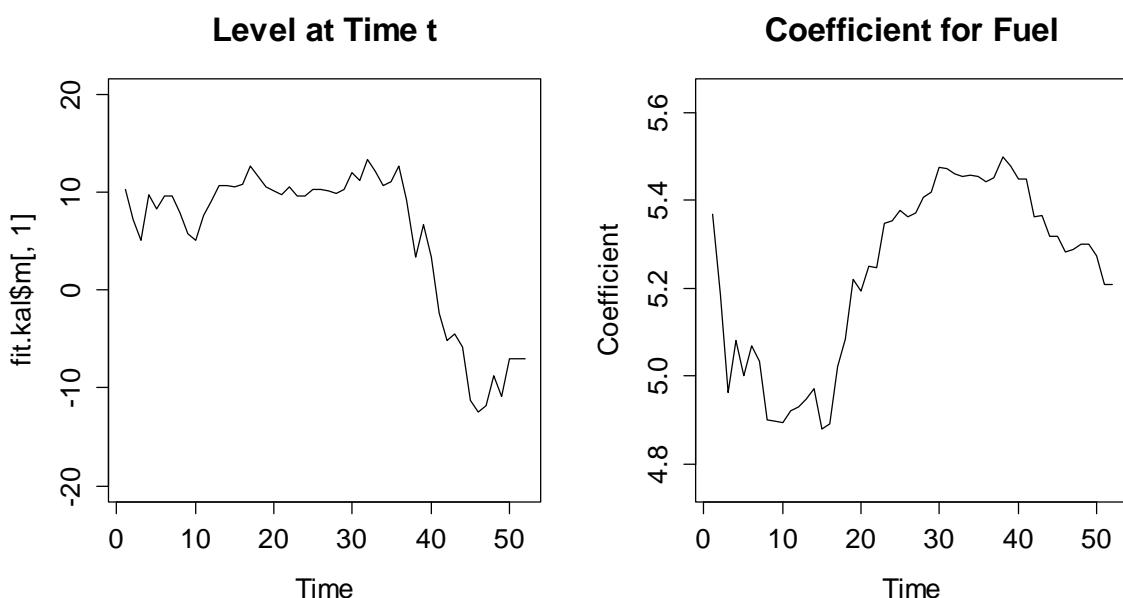
As we can see, we only allow for some small, random permutations in the level L_t , but not in the other regression coefficients. The observation equation then describes the regression problem, i.e.

$$Y_t = F_t X_t + V_t, \text{ where } Y_t = ACC_t, F_t = (1, Fuel_t, 1_{Q_2(t)}, 1_{Q_3(t)}, 1_{Q_4(t)}), V_t \sim N(0, \sigma_V^2)$$

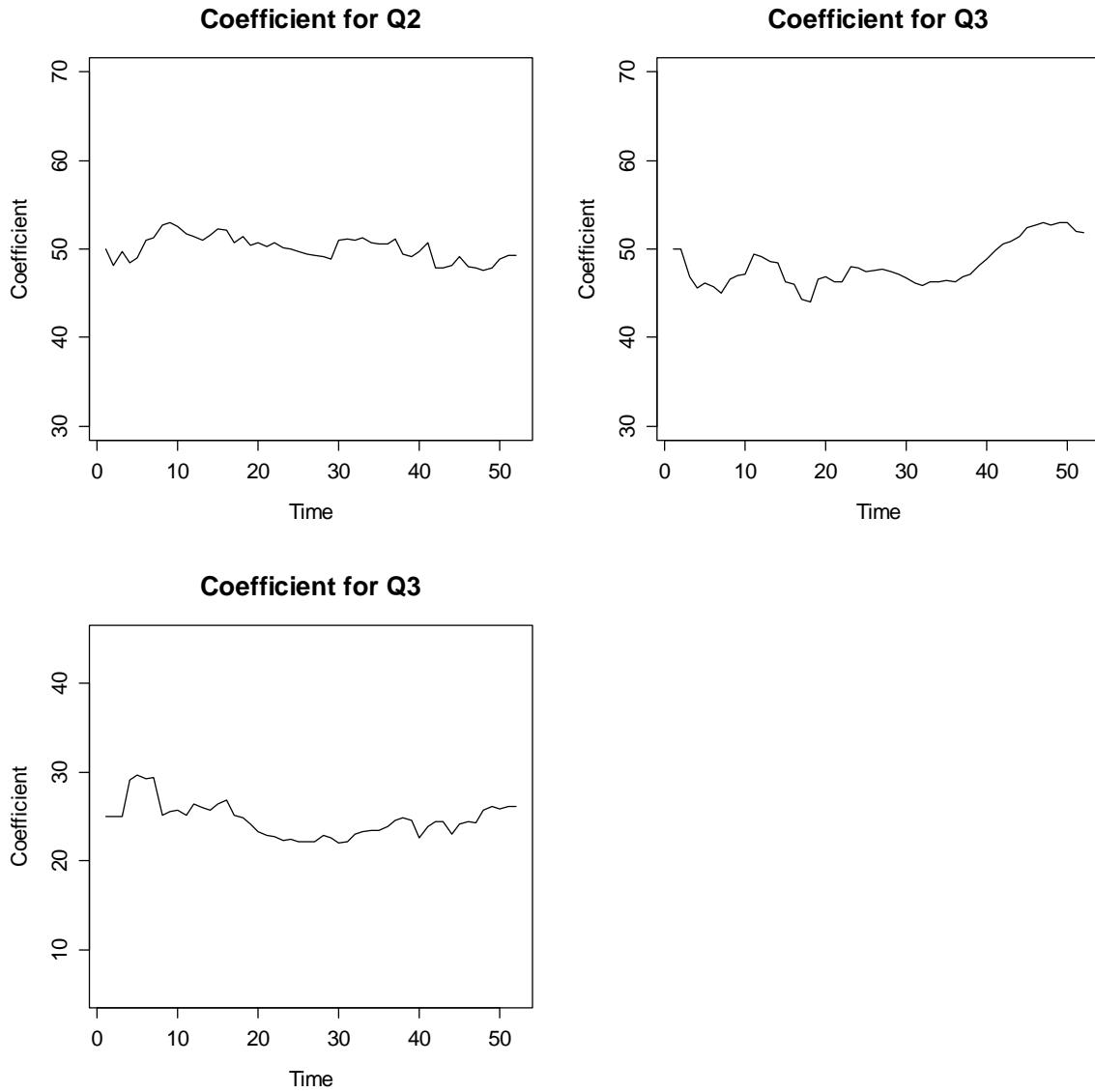
The variance of the noise term σ_V^2 is set to 600, which is the error variance in the canonical regression fit. Also the starting values for Kalman filtering, as well as the variances of these initial states are taken from there. Hence, the code for the state space formulation, as well as for Kalman filtering is as follows:

```
> y.mat      <- as.matrix(regdat$ACC)
> x.mat      <- model.matrix(fit)
> ssf        <- SS(y=y.mat, x=x.mat,
+   Fmat=function(tt,x,phi) return(t(x[tt,,drop=FALSE])),
+   Gmat=function(tt,x,phi) return(diag(5)),
+   Wmat=function(tt,x,phi) return(diag(c(10,0,0,0,0))),
+   Vmat=function(tt,x,phi) return(matrix(600)),
+   m0=matrix(c(0,5,50,50,25),1,5),
+   C0=diag(c(900,1,100,100,100)))
> fit.kal <- kfilter(ssf)
```

The filtered output is in object `m` in the output list. We can extract the estimates for the mean of the state vector at time t , which we display versus time.



Indeed, the level L_t shows a pronounced drop from $t = 35$ to $t = 45$. Hence, the BAT program shows an effect, but there is some delay after the intervention and it takes some time until it has full effect. Finally, we track the estimates for the seasonal coefficients.



The estimates for these coefficients slightly wiggle around over time. However, they do not seem to change systematically, as we had previously guessed.