

# Time Series Analysis Helpfile

*Anubhav Dikshit (anudi287)*

*2019-10-16*

<b>Library</b>	<b>3</b>
Generate two time series and use smoothing filter . . . . .	4
Casuality and Invertiblity. . . . .	6
ACF and Theortical ACF . . . . .	7
<b>Visualization, detrending and residual analysis of Rhine data.</b>	<b>8</b>
ACF Plot . . . . .	8
Detrending using linear regression . . . . .	14
Detrending using kernel smoother . . . . .	16
Detrending using seasonal means model . . . . .	20
Model tuning using SetpAIC . . . . .	22
<b>Analysis of oil and gas time series</b>	<b>23</b>
Checking Stationary . . . . .	23
Log transformation to fix stationary . . . . .	25
Detrending using difference method . . . . .	26
Detrending using smoother . . . . .	30
Detrending using linear regression . . . . .	33
<b>Linear Regressions on Necessarily Lagged Variables and Appropriate Correlation</b>	<b>35</b>
Generate AR and using PACF . . . . .	35
Methods of Moments, Conditional Least Squares and Maximum Likelihood . . . . .	36
<b>ARIMA</b>	<b>37</b>
Sample and Theoretical ACF and PACF . . . . .	37
Forecast and Prediction . . . . .	39
Prediction Band . . . . .	41
Finding a Suitable ARIMA Model and EACF . . . . .	42
<b>State Space Models</b>	<b>60</b>
Kalman Filter Code . . . . .	68

<b>Lecture Slides</b>	<b>72</b>
Lecture 1 . . . . .	72
Lecture 2 . . . . .	123
Lecture 3 . . . . .	158
Lecture 4 . . . . .	187
Lecture 5 . . . . .	209
Lecture 6 . . . . .	248
Lecture 7 . . . . .	280
Lecture 8 . . . . .	300
Lecture 9 . . . . .	313
Teaching Session II . . . . .	337
Teaching Session III . . . . .	353
Summary . . . . .	373

# Library

```
knitr::opts_chunk$set(echo = TRUE)

library("tidyverse") #ggplot and dplyr
library("gridExtra") # combine plots
library("knitr") # for pdf
library("fpp2") #timeseries with autoplot and stuff
library("reshape2") #reshape the data
library("MASS") #StepAIC
library("astsa") #dataset oil and gas is present here
library("zoo") #dataset oil and gas is present here
library("forecast") # for forecasting time series
library("kernlab") #gausspr function
library("TSA") #Q3
library("tseries")
library("matlib") # for inv and I

# The palette with black:
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
                 "#FOE442", "#0072B2", "#D55E00", "#CC79A7")

set.seed(12345)
options(scipen=999)
```

## Generate two time series and use smoothing filter

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.

```
set.seed(12345)

n = 100
x <- vector(length = n)
x2 <- vector(length = n)

x[1] <- 0
x[2] <- 0

#first series generation
for(i in 3:n){
  x[i] <- -0.8 * x[i-2] + rnorm(1,0,1)
}

#second series generation
for(i in 1:n){
  x2[i] <- cos(0.4*pi*i)
}

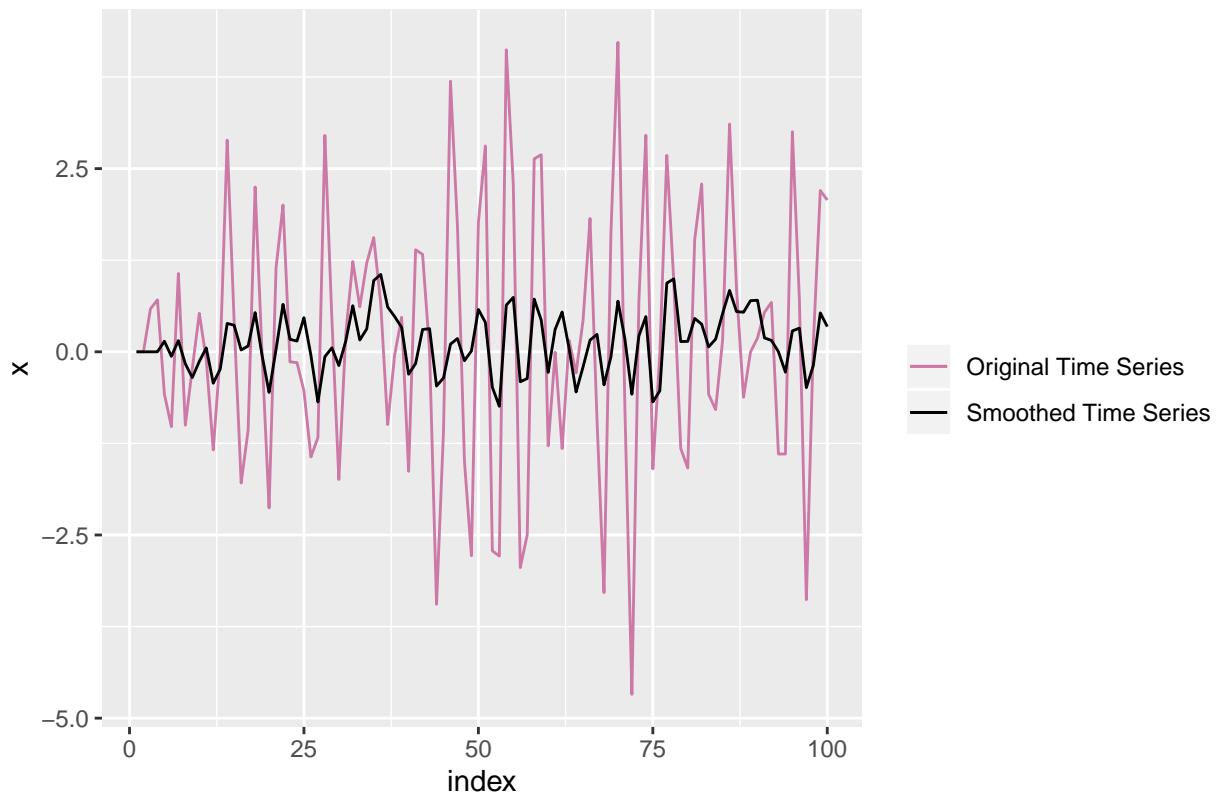
# smoothing filter function
smoothing_filter <- function(x){
  v <- vector(length = length(x))
  for(i in 5:length(x)){
    v[i] = 0.2 * (x[i] + x[i-1] + x[i-2] + x[i-3] + x[i-4])
  }
  return(v)
}

#generate smoothed series
smooth_x <- smoothing_filter(x)
smooth_x2 <- smoothing_filter(x2)

#adding everything to a dataframe
df <- cbind(x,x2,smooth_x,smooth_x2) %>% as.data.frame() %>% mutate(index=1:100)

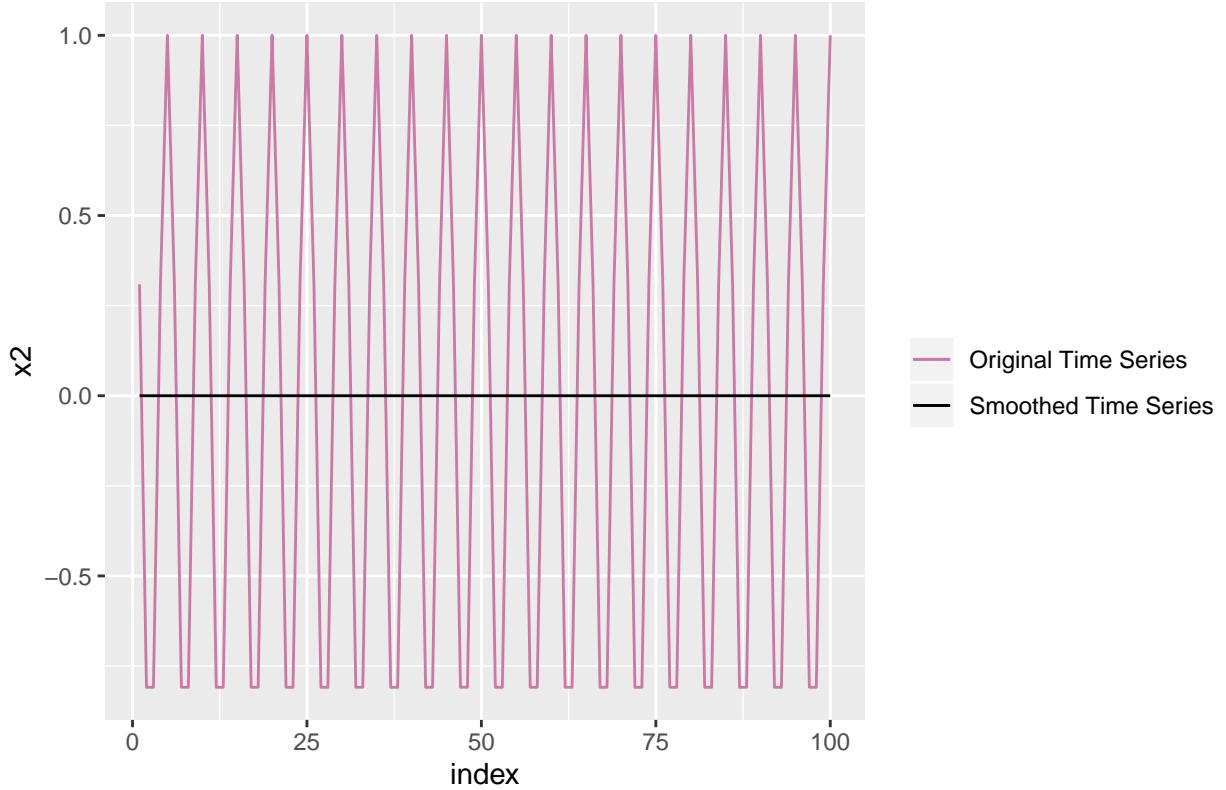
ggplot(df, aes(x=index)) +
  geom_line(aes(y=x, color="Original Time Series")) +
  geom_line(aes(y=smooth_x, color="Smoothed Time Series")) +
  ggtitle("Plot of 1st time series and its smoothed version") +
  scale_colour_manual("", breaks = c("Original Time Series", "Smoothed Time Series"),
                     values = c("#CC79A7", "#000000"))
```

Plot of 1st time series and its smoothed version



```
ggplot(df, aes(x=index)) +  
  geom_line(aes(y=x2, color="Original Time Series")) +  
  geom_line(aes(y=smooth_x2, color="Smoothed Time Series")) +  
  ggtitle("Plot of 2ND time series and its smoothed version") +  
  scale_colour_manual("", breaks = c("Original Time Series", "Smoothed Time Series"),  
                     values = c("#CC79A7", "#000000"))
```

## Plot of 2ND time series and its smoothed version



### Causality and Invertibility.

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.

Causality: ARMA(p,q) is causal iff roots  $\phi(z') = 0$  are outside unit circle. eg:  $x_t = 0.4x_{t-1} + 0.3x_{t-2} + w_t$ , roots are  $-> 1 - 0.4B + 0.3B^2$

equation is:  $\phi(Z) = 1 - 4B + 2B^2 + 0B^3 + 0B^4 + B^5$

```
z = c(1, -4, 2, 0, 0, 1)
polyroot(z)
```

```
## [1] 0.2936658+0.000000i -1.6793817+0.000000i 1.0000000-0.000000i
## [4] 0.1928579-1.410842i 0.1928579+1.410842i
```

```
any(Mod(polyroot(z)) <= 1)
```

```
## [1] TRUE
```

Invertible: ARMA(p,q) is causal iff roots  $\theta(z') = 0$  are outside unit circle.

equation is:  $\theta(Z) = 1 + 3B^2 + B^4 - 4B^6$

```

z = c(1,0,3,0,1,0,-4)
polyroot(z)

## [1] 0.1375513+0.6735351i -0.1375513+0.6735351i -0.1375513-0.6735351i
## [4] 0.1375513-0.6735351i 1.0580446+0.0000000i -1.0580446+0.0000000i

any(Mod(polyroot(z))<=1)

## [1] TRUE

```

## ACF and Theoretical ACF

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.

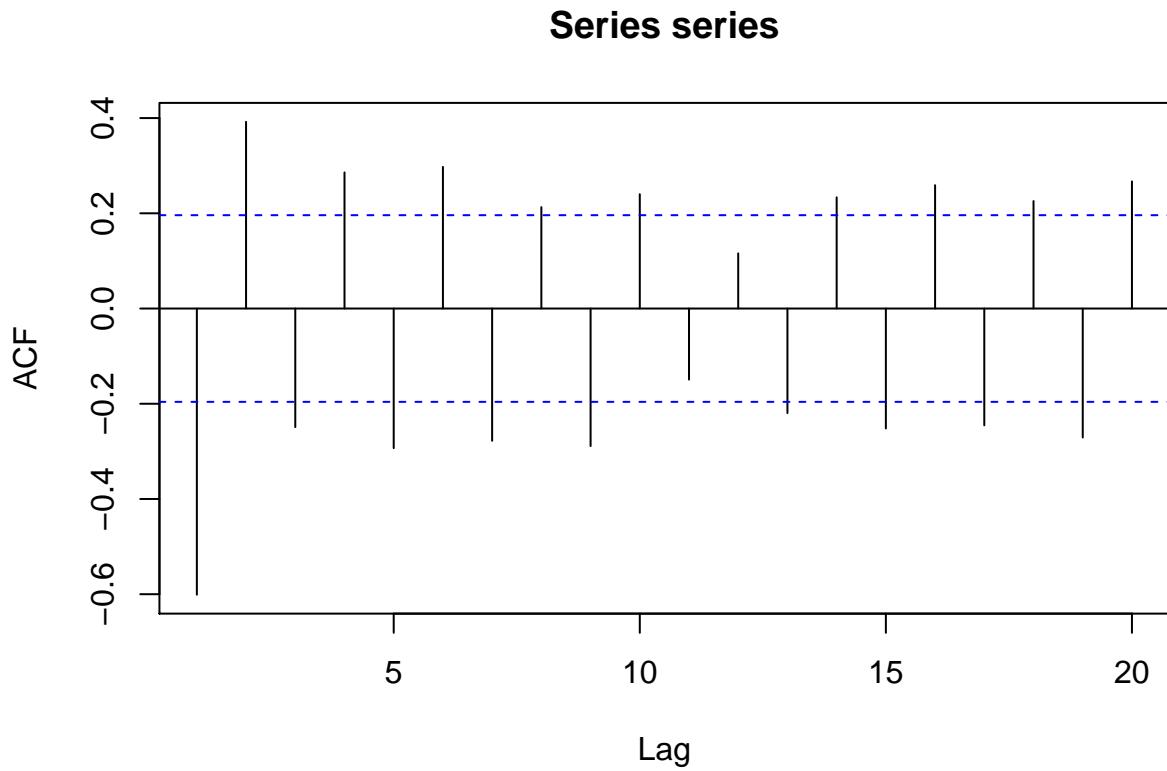
```

set.seed(54321)

series <- arima.sim(n = 100, list(ar = c(-3/4), ma = c(0,-1/9)))

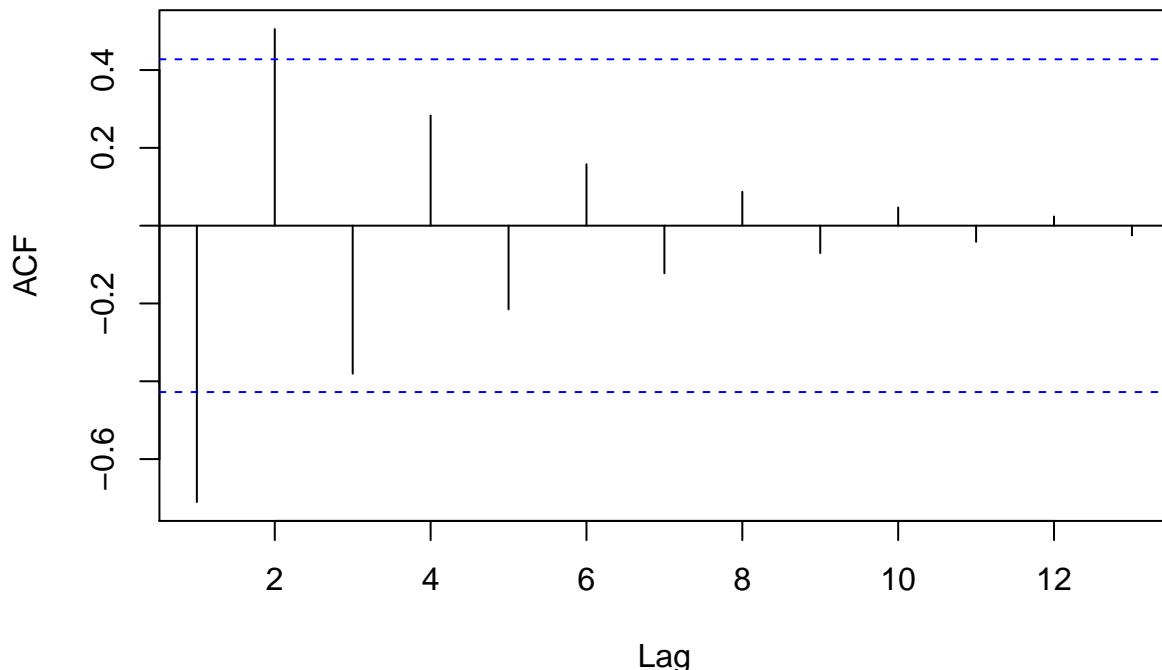
acf(series)

```



```
acf(ARMAacf(ar = c(-3/4), ma = c(0, -1/9), lag.max = 20))
```

### Series ARMAacf(ar = c(-3/4), ma = c(0, -1/9), lag.max = 20)



## Visualization, detrending and residual analysis of Rhine data.

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

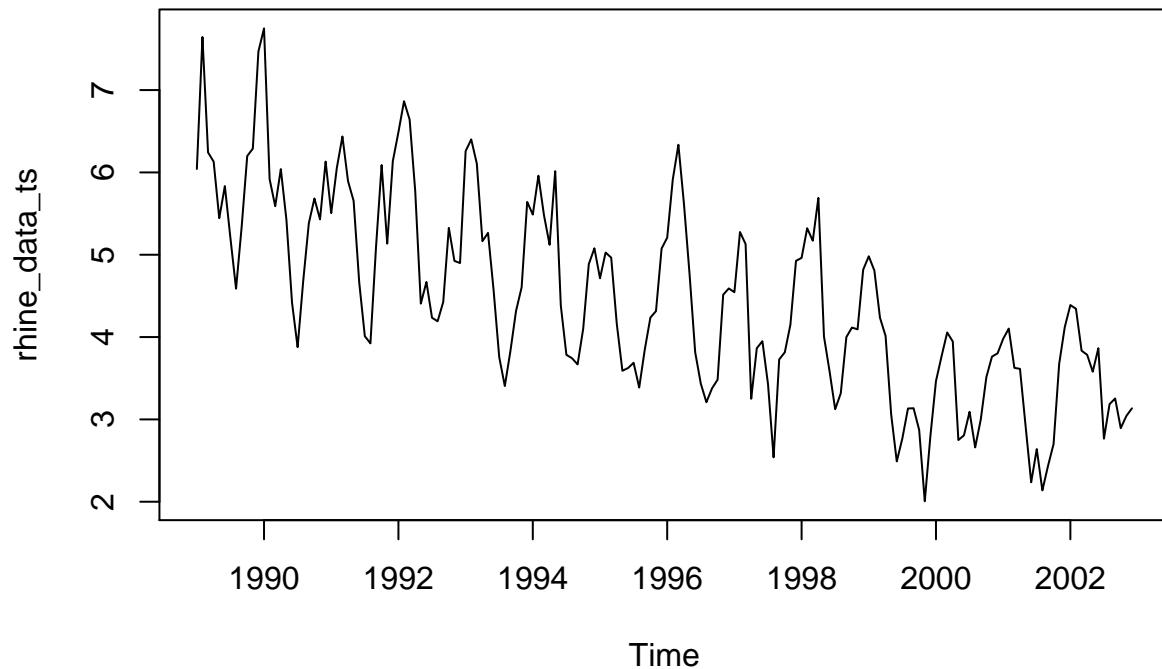
### ACF Plot

- a) Import the data to R, convert it appropriately to ts object (use function `ts()`) and explore it by plotting the time series, creating scatter plots of  $x_t$  against  $x_{t-1}, \dots, x_{t-12}$ . Analyze the time series plot and the 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 scatter plots? Does the variance seem to change over time? Which variables in the scatter plots seem to have a significant relation to each other?

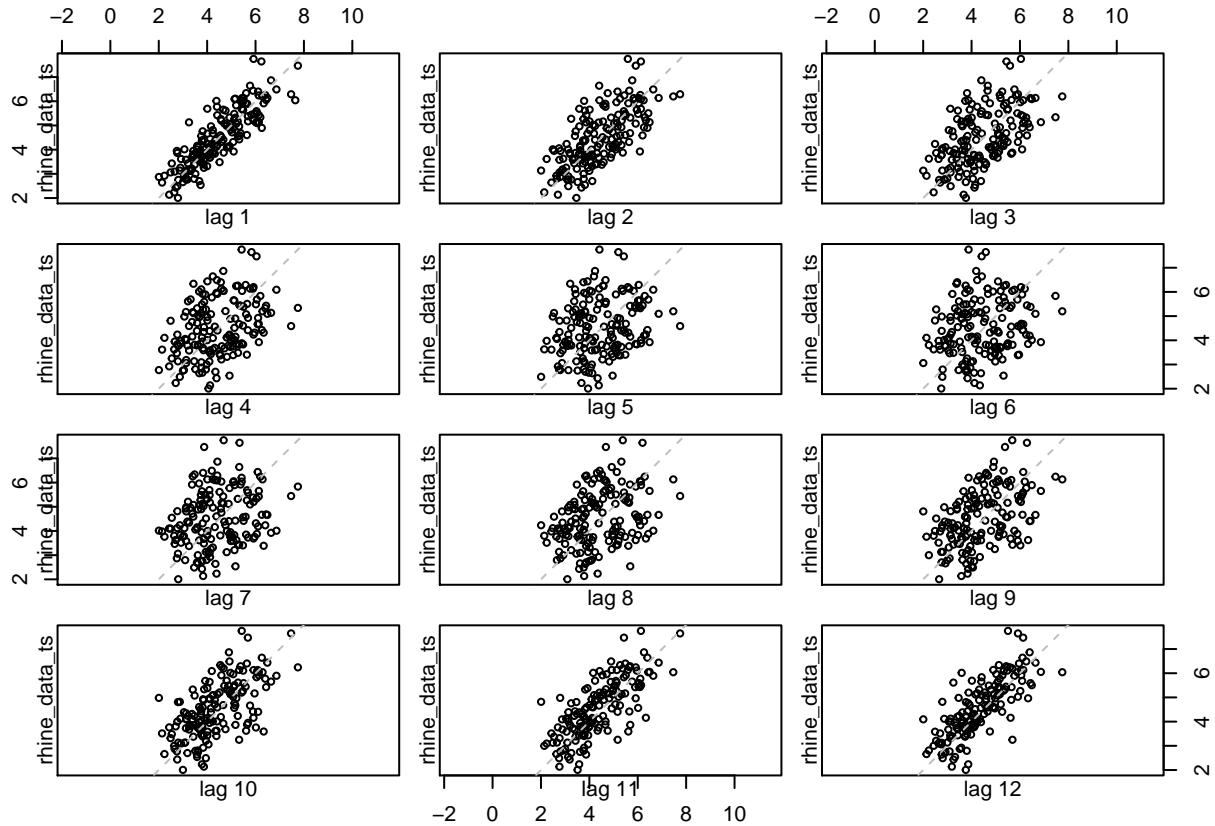
```
set.seed(12345)
rhine_data <- read.csv2("Rhine.csv")
rhine_data_ts <- ts(data = rhine_data$TotN_conc,
                     start = c(1989, 1),
                     frequency = 12)

plot.ts(rhine_data_ts, main="Time Series of Nitrogen Concentration in Rhine")
```

## Time Series of Nitrogen Concentration in Rhine

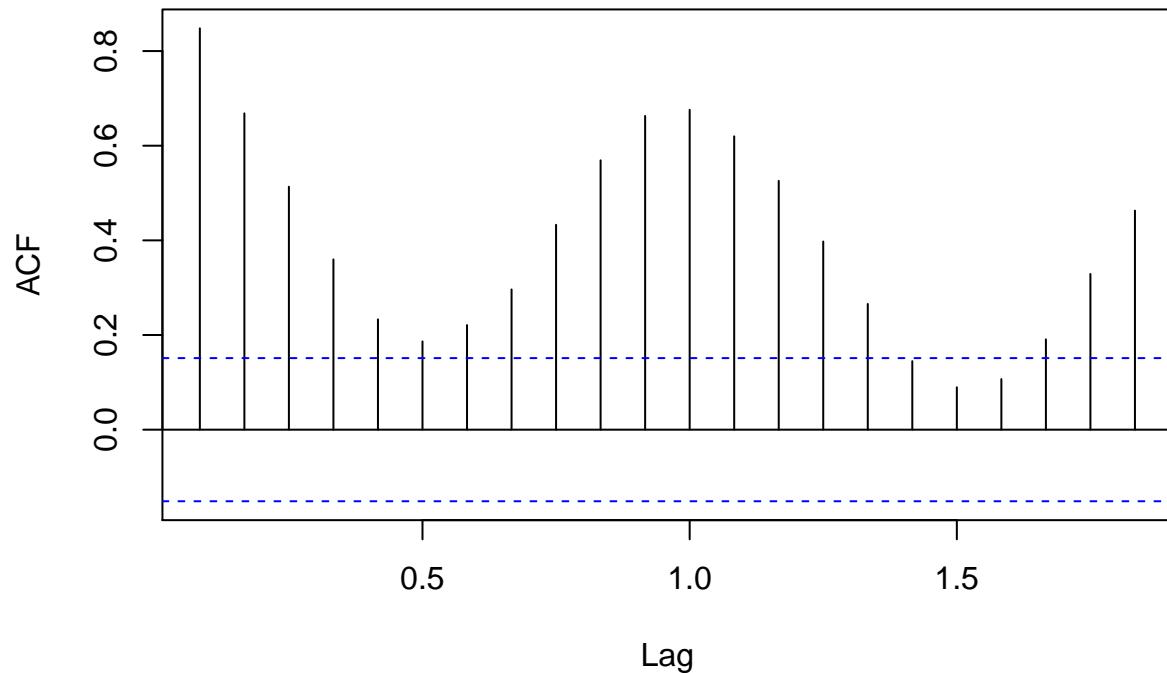


```
lag.plot(rhine_data_ts, lags = 12)
```

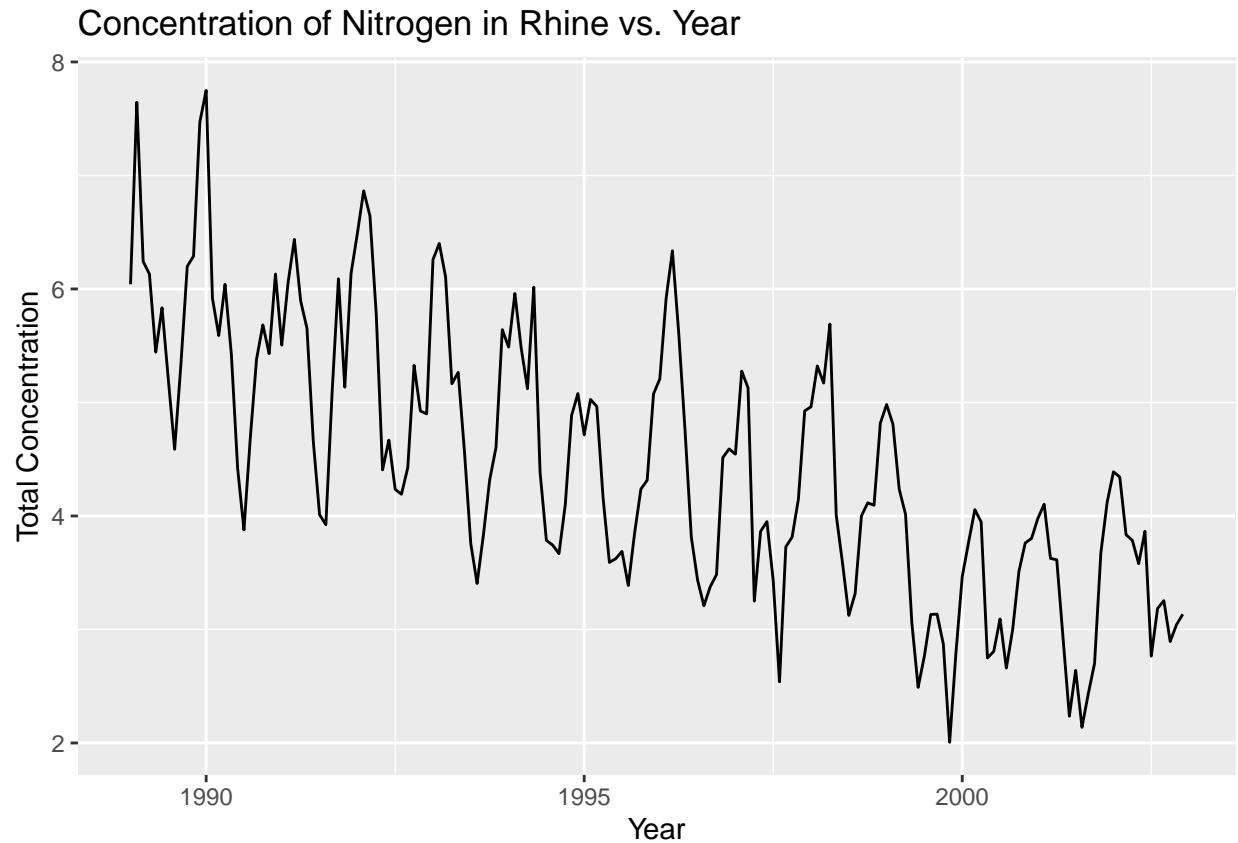


```
acf(rhine_data_ts)
```

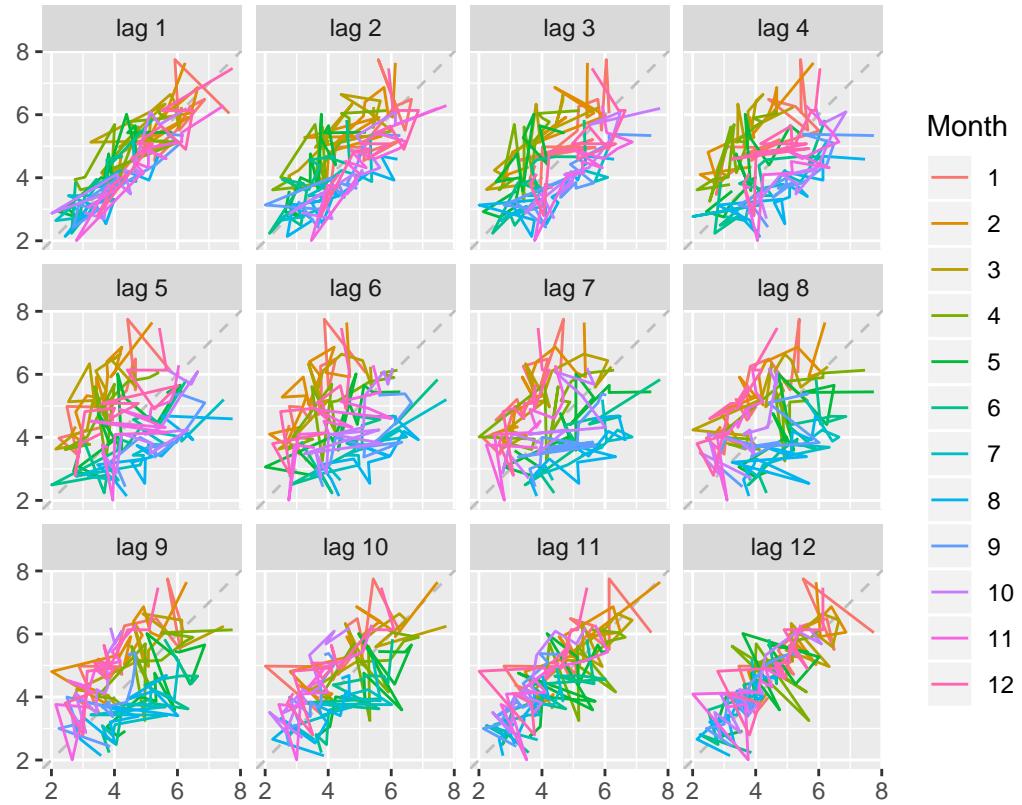
### Series rhine\_data\_ts



```
#alternative  
autoplot(rhine_data_ts) + ylab("Total Concentration") + xlab("Year") +  
  ggtitle("Concentration of Nitrogen in Rhine vs. Year")
```

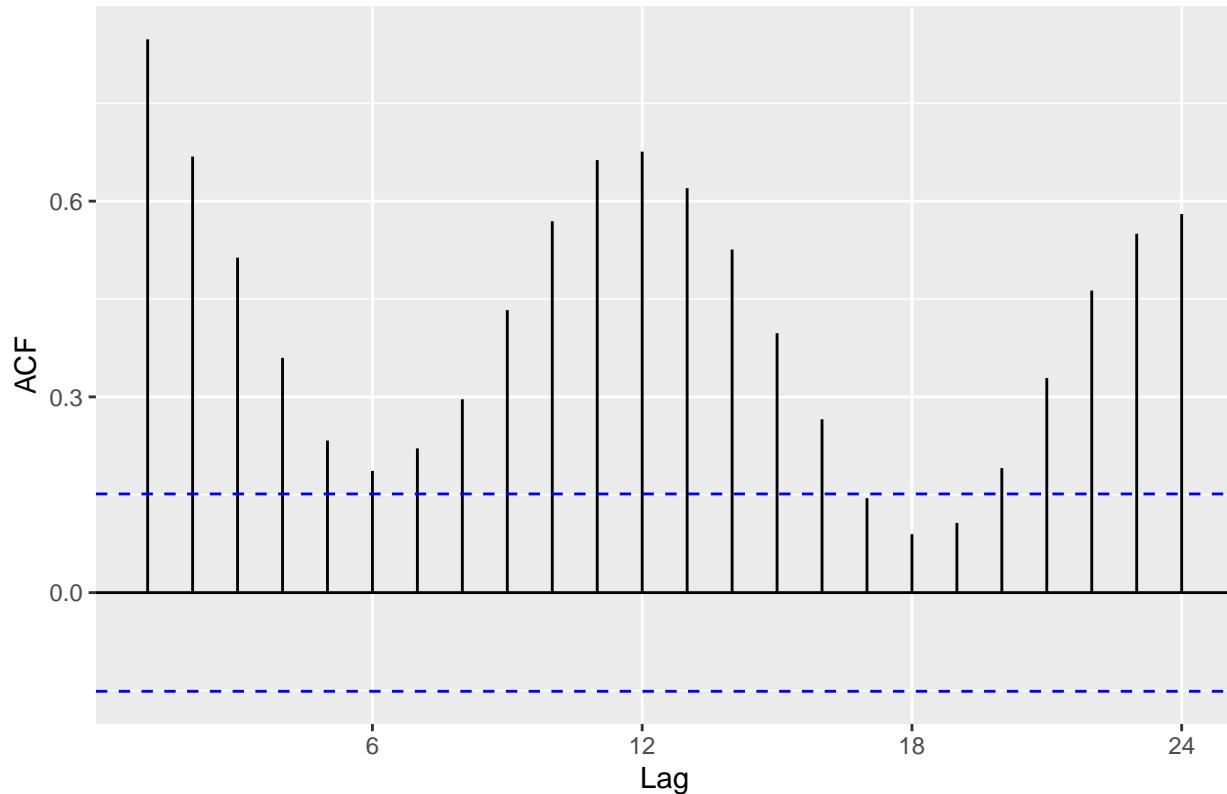


```
gglagplot(rhine_data_ts, lags = 1, set.lags = 1:12, color=FALSE)
```



```
ggAcf(rhine_data_ts) + ggttitle("ACF for Total Nitrogen Concentration")
```

## ACF for Total Nitrogen Concentration



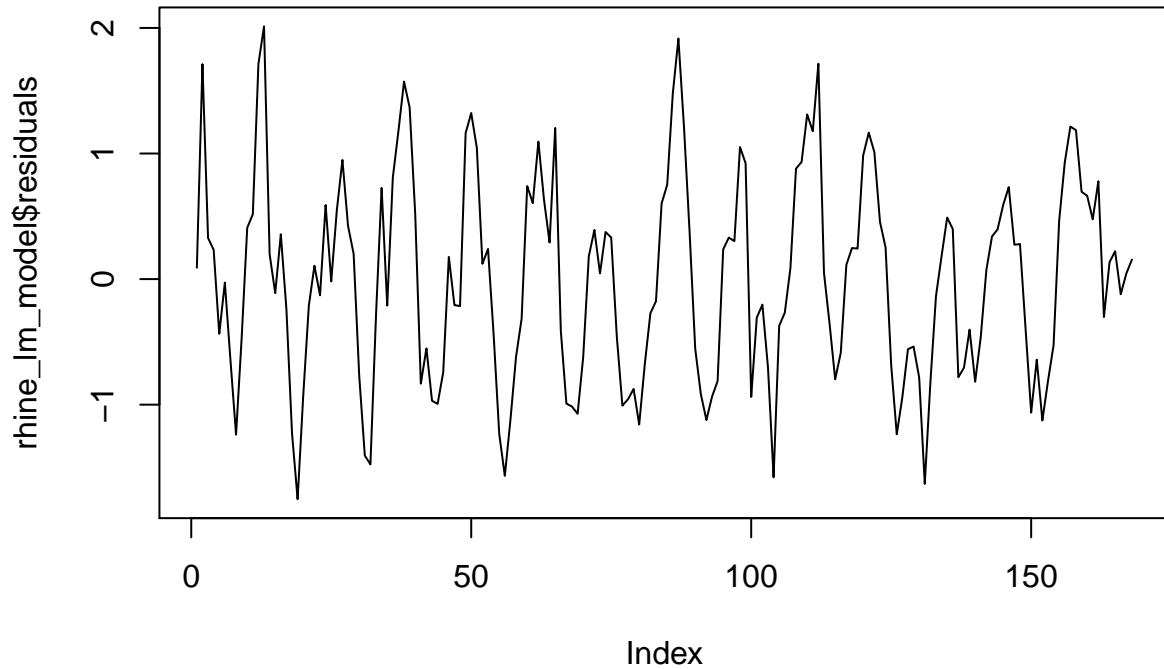
## Detrending using linear regression

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 how this pattern might be related to seasonality of the series.

```
set.seed(12345)

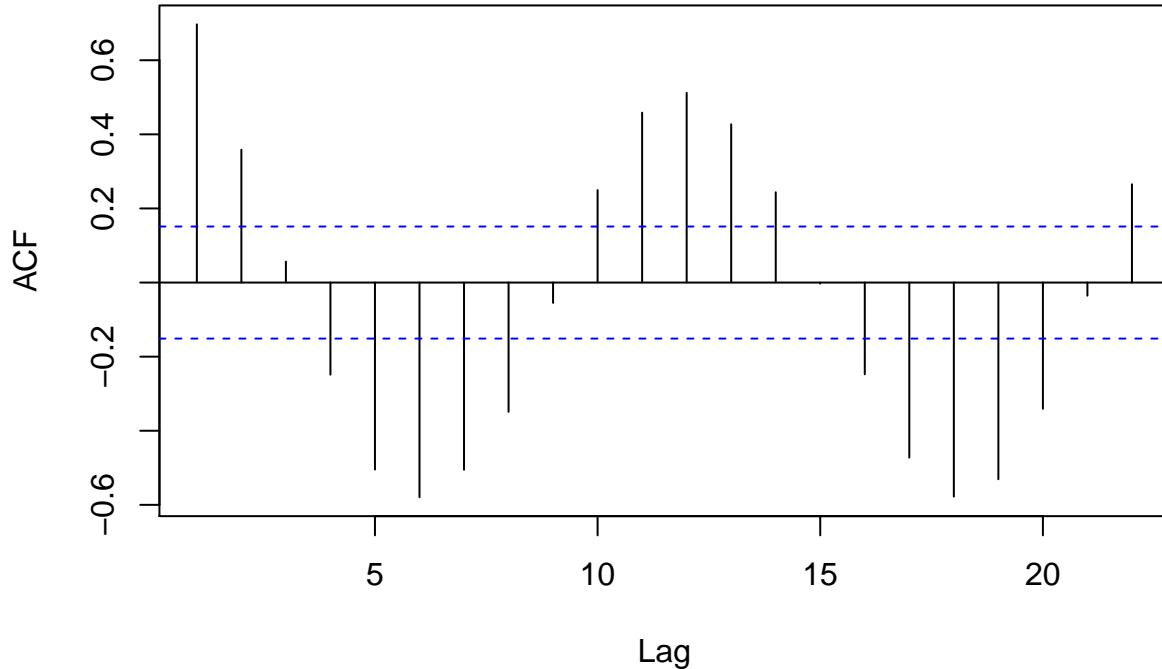
rhine_lm_model <- lm(TotN_conc~Time, data=rhine_data)
plot(rhine_lm_model$residuals, type = 'l', main="Plot of Residual from the
linear model of Nitrogen Concentration")
```

## Plot of Residual from the linear model of Nitrogen Concentration



```
acf(rhine_lm_model$residuals)
```

## Series rhine\_lm\_model\$residuals



### Detrending using kernel smoother

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?

```
set.seed(12345)

model_smooth_lag_5 <- ksmooth(x = rhine_data$Time, y = rhine_data$TotN_conc,
                                bandwidth=5)
model_smooth_lag_10 <- ksmooth(x = rhine_data$Time, y = rhine_data$TotN_conc,
                                 bandwidth=10)
model_smooth_lag_20 <- ksmooth(x = rhine_data$Time, y = rhine_data$TotN_conc,
                                 bandwidth=20)

model_smooth_lag_5_residual <- rhine_data$TotN_conc - model_smooth_lag_5$y
model_smooth_lag_10_residual <- rhine_data$TotN_conc - model_smooth_lag_10$y
model_smooth_lag_20_residual <- rhine_data$TotN_conc - model_smooth_lag_20$y

residual_df <- cbind(model_smooth_lag_5_residual, model_smooth_lag_10_residual,
                      model_smooth_lag_20_residual, rhine_data$Time) %>%
  as.data.frame()

colnames(residual_df) <- c("lag_5_residual", "lag_10_residual",
```

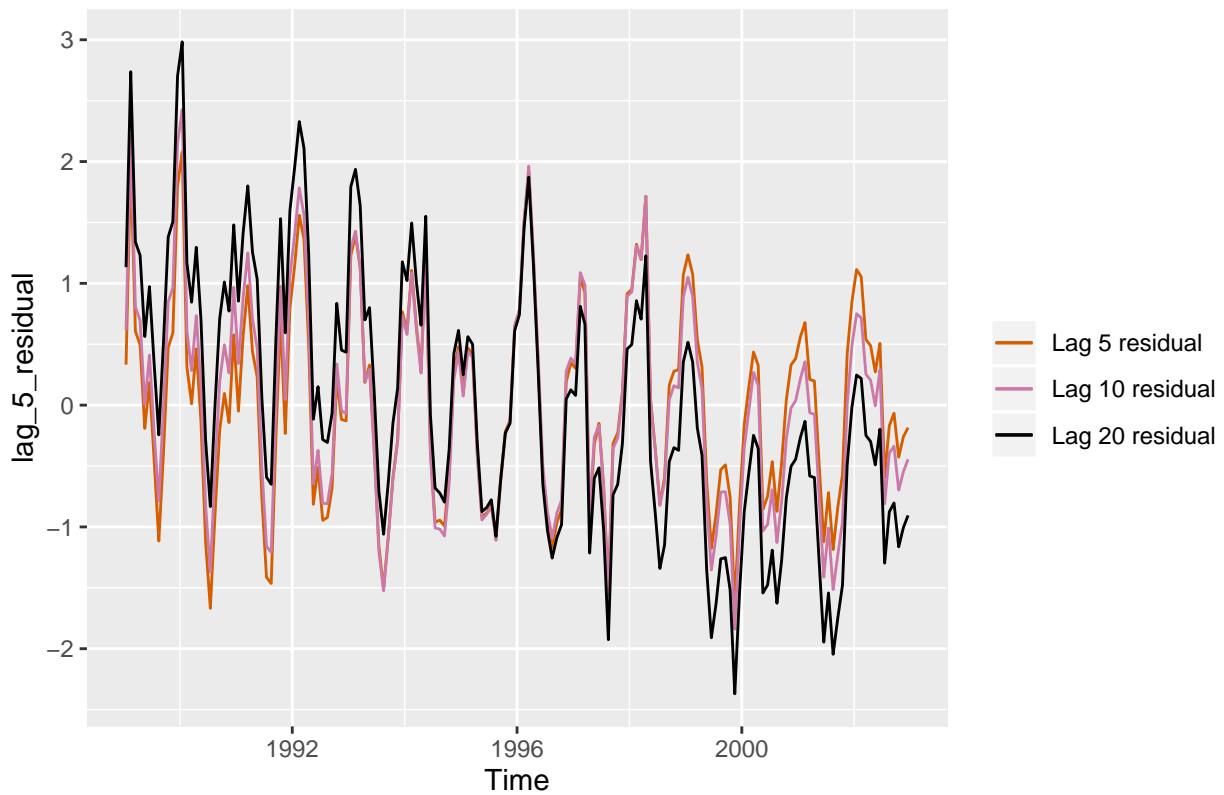
```

    "lag_20_residual", "Time"))

ggplot(residual_df, aes(x=Time)) +
  geom_line(aes(y=lag_5_residual, color="Lag 5 residual")) +
  geom_line(aes(y=lag_10_residual, color="Lag 10 residual")) +
  geom_line(aes(y=lag_20_residual, color="Lag 20 residual")) +
  ggtitle("Residual vs. Time by Lag") +
  scale_colour_manual("", breaks = c("Lag 5 residual", "Lag 10 residual",
                                    "Lag 20 residual"),
                      values = c("#CC79A7", "#000000", "#D55E00"))

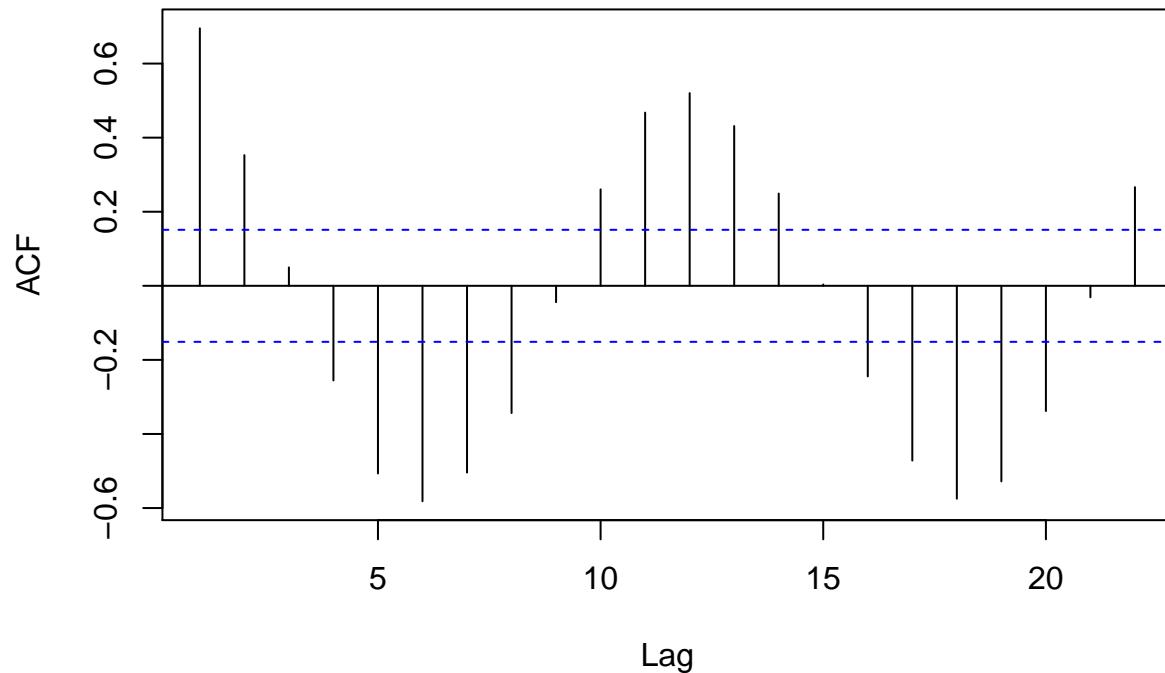
```

Residual vs. Time by Lag



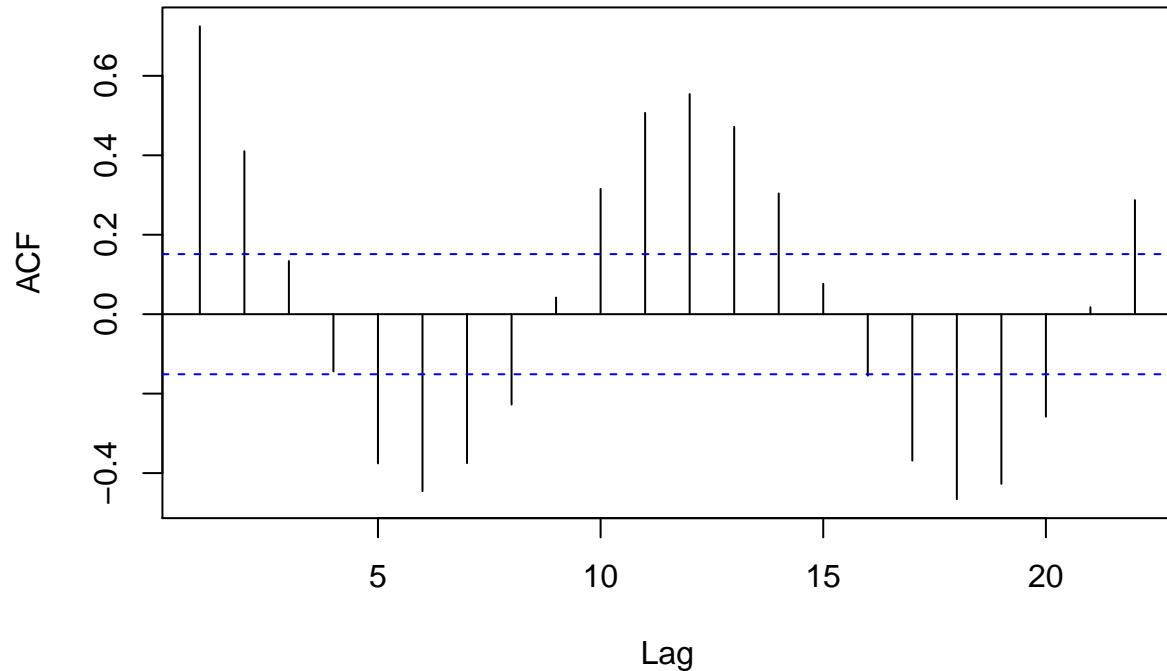
```
acf(model_smooth_lag_5_residual)
```

### Series model\_smooth\_lag\_5\_residual



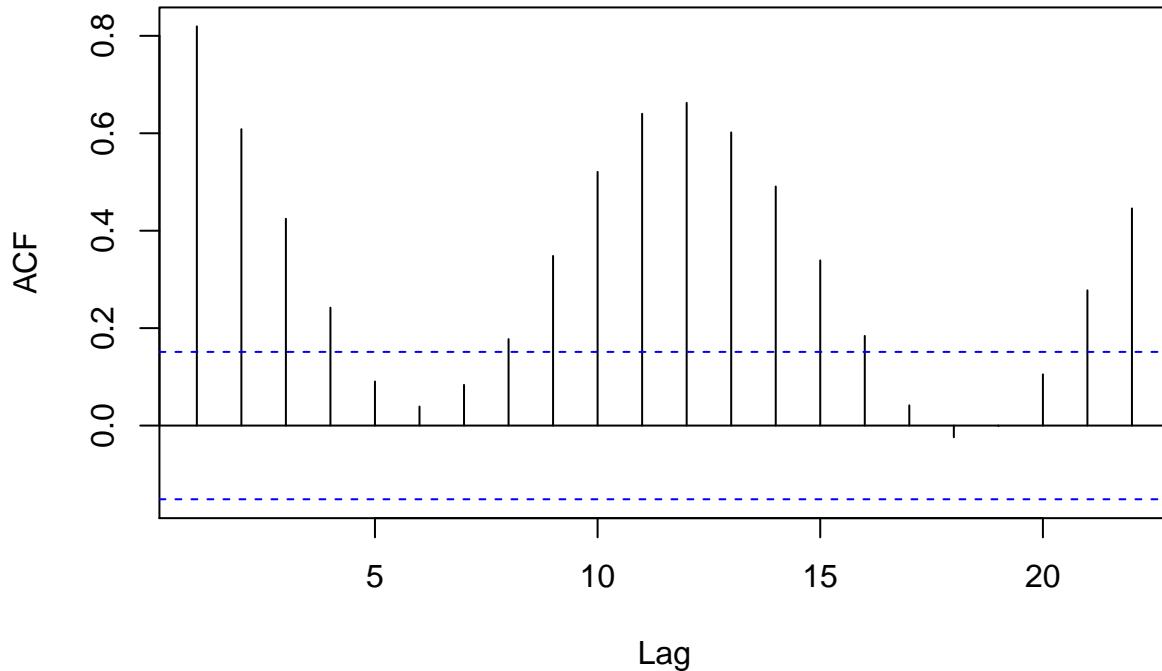
```
acf(model_smooth_lag_10_residual)
```

### **Series model\_smooth\_lag\_10\_residual**



```
acf(model_smooth_lag_20_residual)
```

## Series model\_smooth\_lag\_20\_residual



### Detrending using seasonal means model

d) Eliminate the trend by fitting the following so-called seasonal means model:  $x_t = \alpha_0 + \alpha_1 t + \beta_1 I(month = 2) + \dots + \beta_{12} I(month = 12) + w_t$ , where  $I(x)=1$  is an identity function. 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.

```

set.seed(12345)

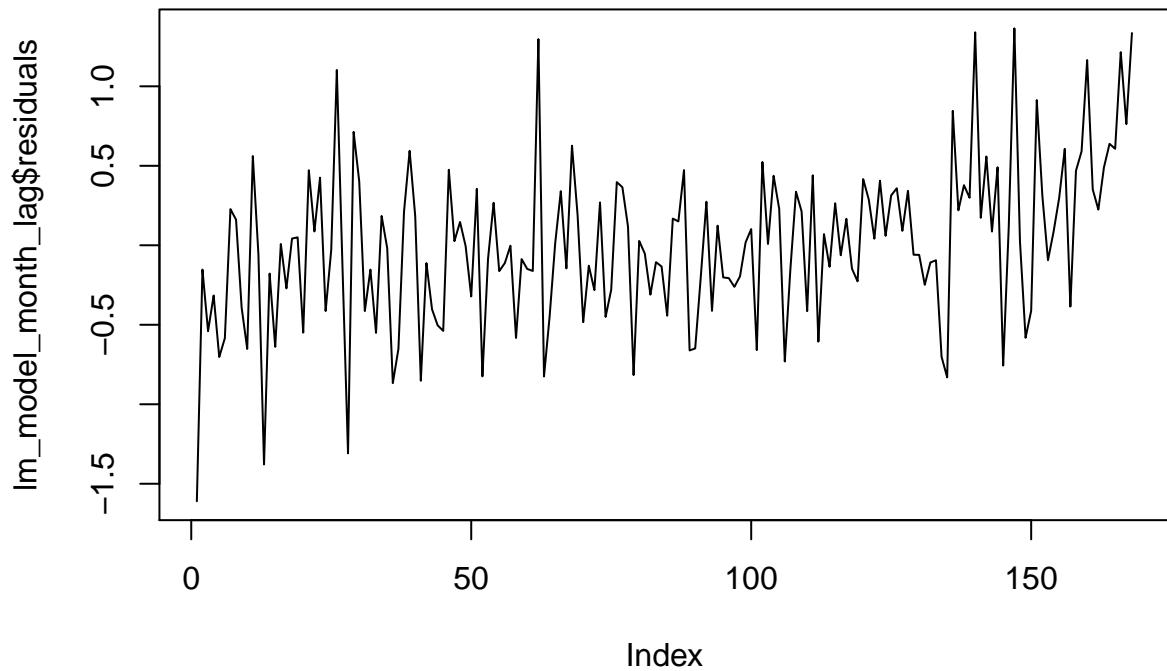
rhine_data_wide <- rhine_data
rhine_data_wide$dummy <- "1"
rhine_data_wide$Month <- paste0("Month_",rhine_data_wide$Month)
rhine_data_wide <- dcast(rhine_data_wide,
                         formula = TotN_conc+Year+Time~Month,
                         value.var = "dummy", fill = "0")

lm_model_month_lag <- lm(data=rhine_data_wide,
                           TotN_conc~Time+Month_1+Month_2+Month_3+Month_4+Month_5+
                           Month_6+
                           Month_7+
                           Month_8+Month_9+Month_10+Month_11+Month_12)

plot(lm_model_month_lag$residuals, type = 'l', main="Plot of the Residuals vs. Time")

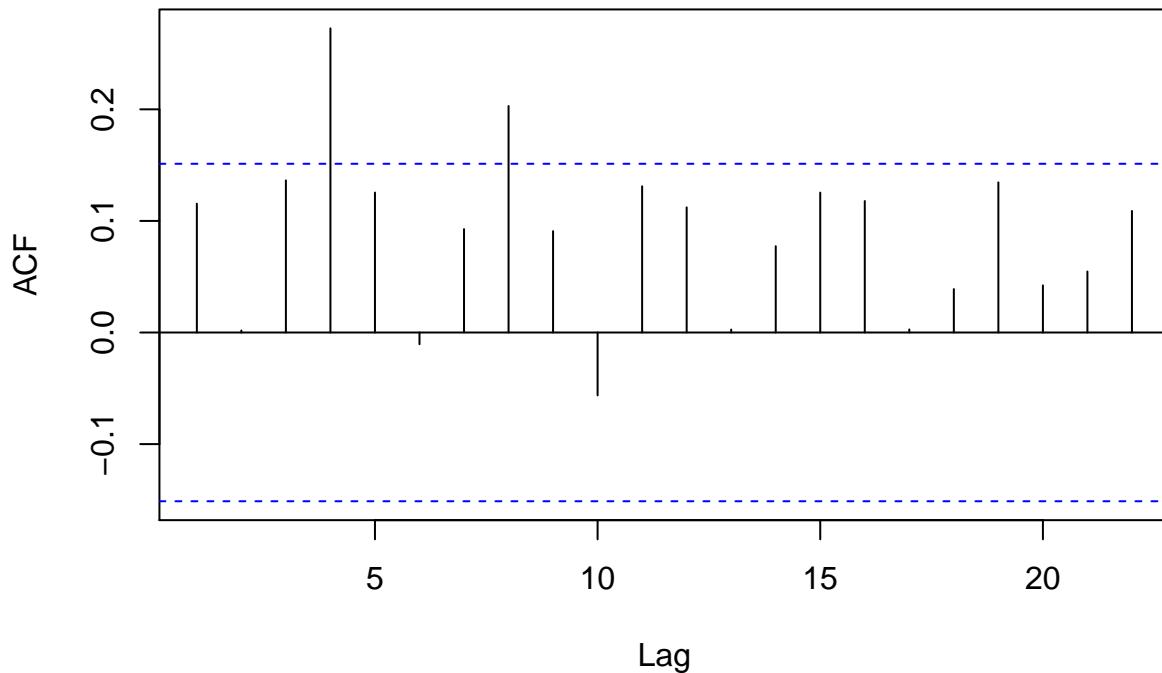
```

## Plot of the Residuals vs. Time



```
acf(lm_model_month_lag$residuals)
```

## Series lm\_model\_month\_lag\$residuals



## Model tuning using SetpAIC

e) Perform step-wise variable selection in model from step d). Which model gives you the lowest AIC value? Which variables are left in the model?

```
set.seed(12345)

lm_model_month_lag_step <- stepAIC(lm_model_month_lag,
                                      scope = list(upper = ~Time+Month_1+Month_2+
                                                    Month_3+Month_4+Month_5+
                                                    Month_6+Month_7+
                                                    Month_8+Month_9+Month_10+
                                                    Month_11+Month_12,
                                                    lower = ~1),
                                      trace = TRUE,
                                      direction="backward")

## Start:  AIC=-202.02
## TotN_conc ~ Time + Month_1 + Month_2 + Month_3 + Month_4 + Month_5 +
##               Month_6 + Month_7 + Month_8 + Month_9 + Month_10 + Month_11 +
##               Month_12
##
## Step:  AIC=-202.02
```

```

## TotN_conc ~ Time + Month_1 + Month_2 + Month_3 + Month_4 + Month_5 +
##   Month_6 + Month_7 + Month_8 + Month_9 + Month_10 + Month_11
##
##          Df Sum of Sq      RSS      AIC
## - Month_4    1     0.200  43.436 -203.249
## - Month_1    1     0.220  43.456 -203.170
## - Month_3    1     0.331  43.567 -202.743
## <none>           43.237 -202.023
## - Month_2    1     1.440  44.677 -198.517
## - Month_11   1     2.305  45.541 -195.297
## - Month_5    1     3.274  46.511 -191.760
## - Month_10   1     3.401  46.637 -191.303
## - Month_9    1     7.853  51.089 -175.986
## - Month_6    1     8.215  51.452 -174.797
## - Month_7    1    14.321  57.557 -155.959
## - Month_8    1    16.488  59.725 -149.749
## - Time       1   118.387 161.624    17.499
##
## Step:  AIC=-203.25
## TotN_conc ~ Time + Month_1 + Month_2 + Month_3 + Month_5 + Month_6 +
##   Month_7 + Month_8 + Month_9 + Month_10 + Month_11
##
##          Df Sum of Sq      RSS      AIC
## <none>           43.436 -203.249
## - Month_1    1     0.640  44.077 -202.790
## - Month_3    1     0.851  44.288 -201.988
## - Month_11   1     2.235  45.671 -196.819
## - Month_2    1     2.706  46.142 -195.096
## - Month_5    1     3.355  46.791 -192.748
## - Month_10   1     3.502  46.938 -192.223
## - Month_9    1     8.868  52.304 -174.036
## - Month_6    1     9.317  52.753 -172.602
## - Month_7    1    16.912  60.348 -150.004
## - Month_8    1    19.636  63.072 -142.586
## - Time       1   118.194 161.630    15.506

colnames(lm_model_month_lag_step$model)

```

```

## [1] "TotN_conc" "Time"      "Month_1"    "Month_2"    "Month_3"
## [6] "Month_5"   "Month_6"    "Month_7"    "Month_8"    "Month_9"
## [11] "Month_10"  "Month_11"

```

## Analysis 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.

### Checking Stationary

- 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.

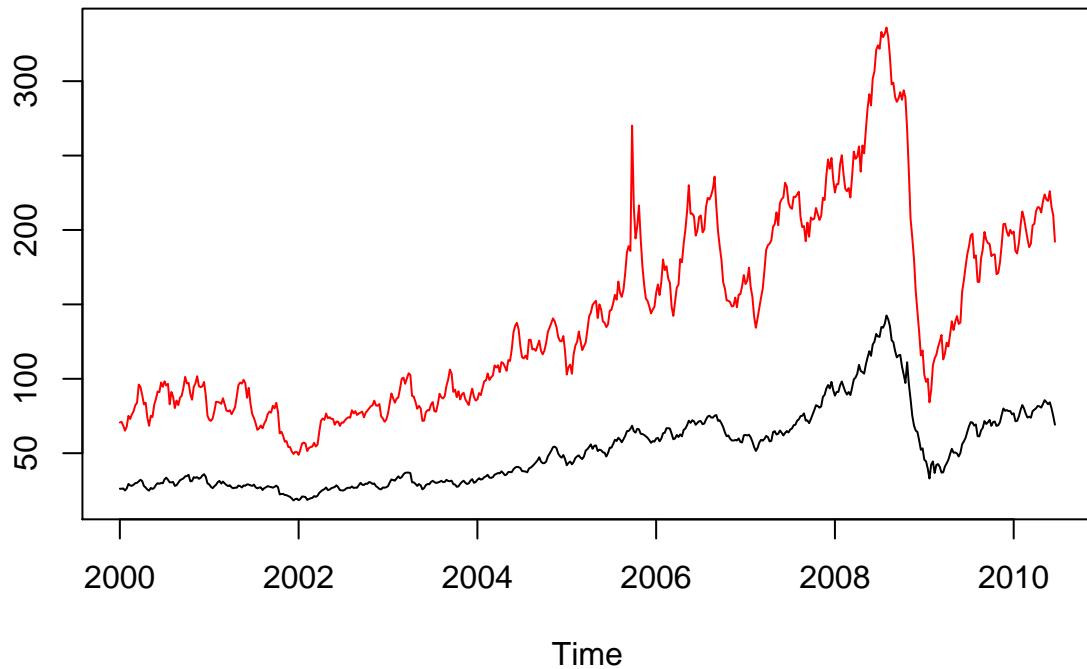
```

set.seed(12345)

data_oil <- astsa::oil
data_gas <- astsa::gas

ts.plot(data_oil, data_gas, gpars = list(col = c("black", "red")))

```



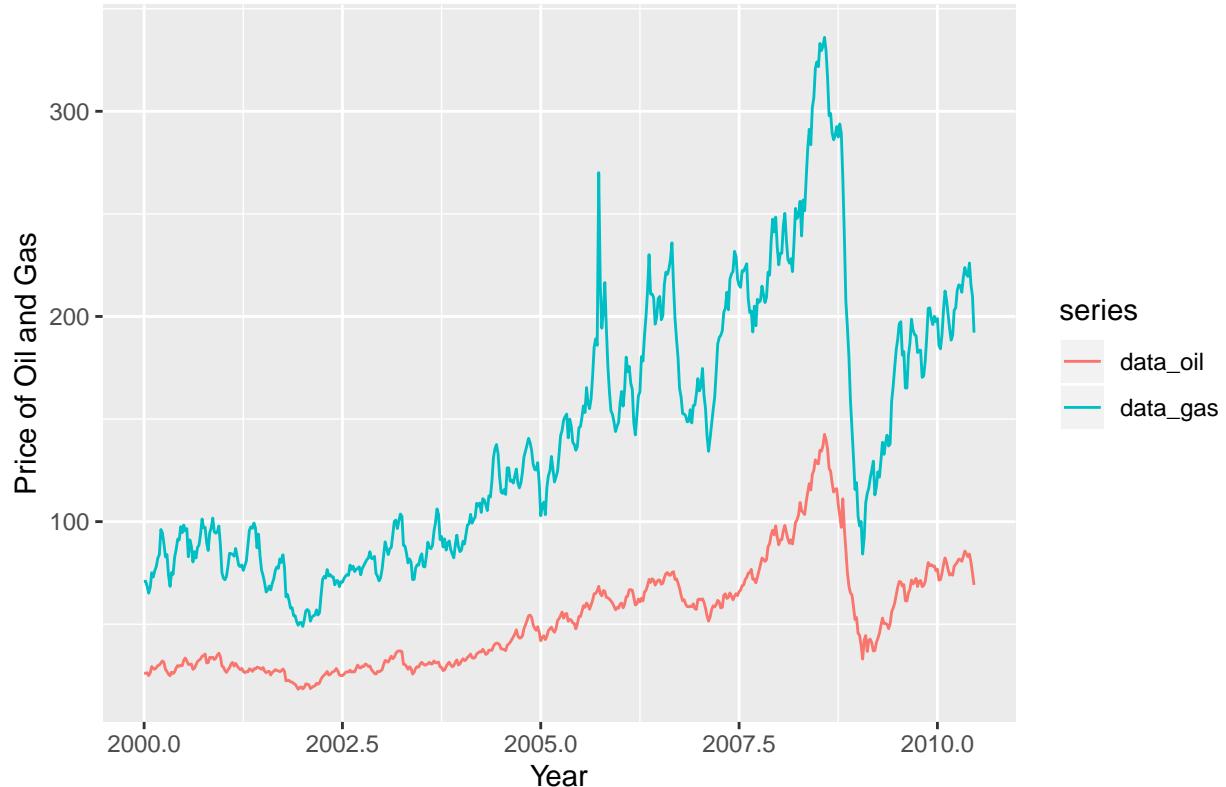
```

#alternative

autoplot(ts(cbind(data_oil, data_gas), start = 2000, frequency = 52)) +
  ylab("Price of Oil and Gas") + xlab("Year") +
  ggtitle("Price of Oil and Gas vs. Years")

```

## Price of Oil and Gas vs. Years



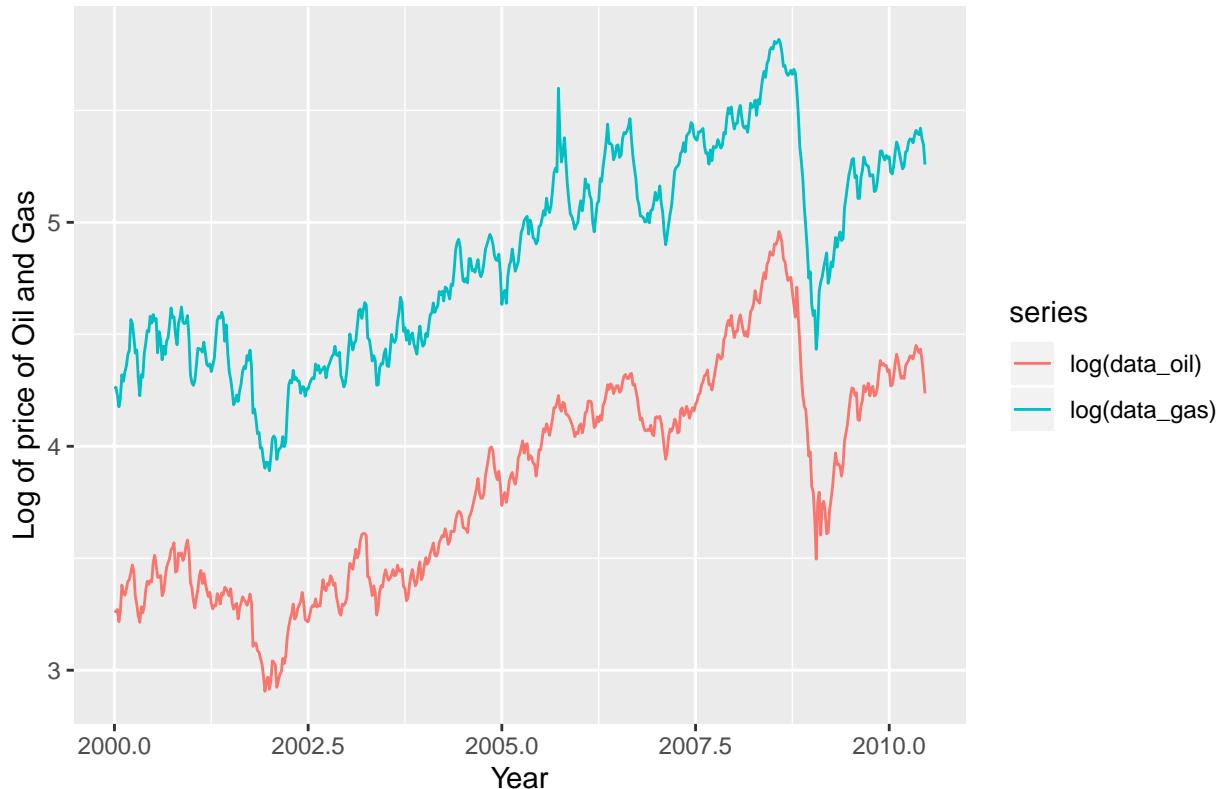
### Log transformation to fix stationary

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?

```
set.seed(12345)

autoplot(ts(cbind(log(data_oil), log(data_gas)), start = 2000, frequency = 52)) +
  ylab("Log of price of Oil and Gas") + xlab("Year") +
  ggtitle("Log of price of Oil and Gas vs. Years")
```

## Log of price of Oil and Gas vs. Years



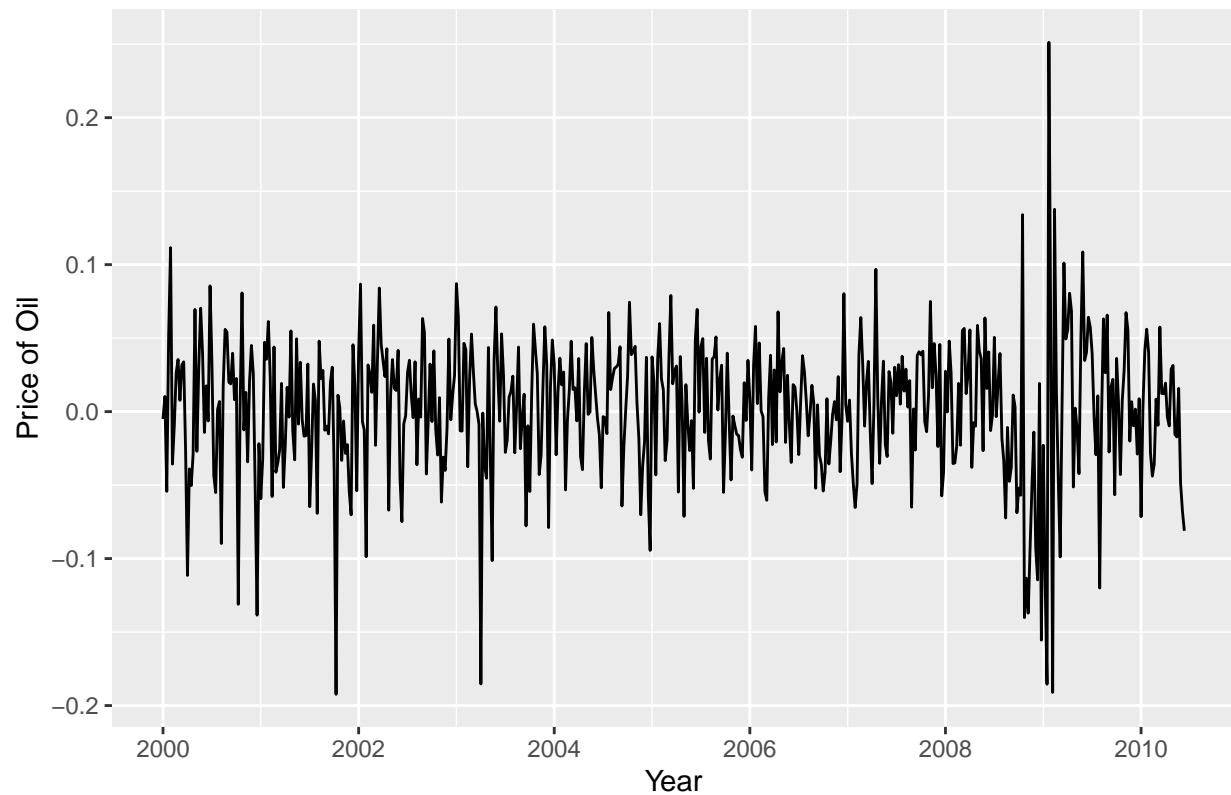
## Detrending using difference method

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

```
set.seed(12345)

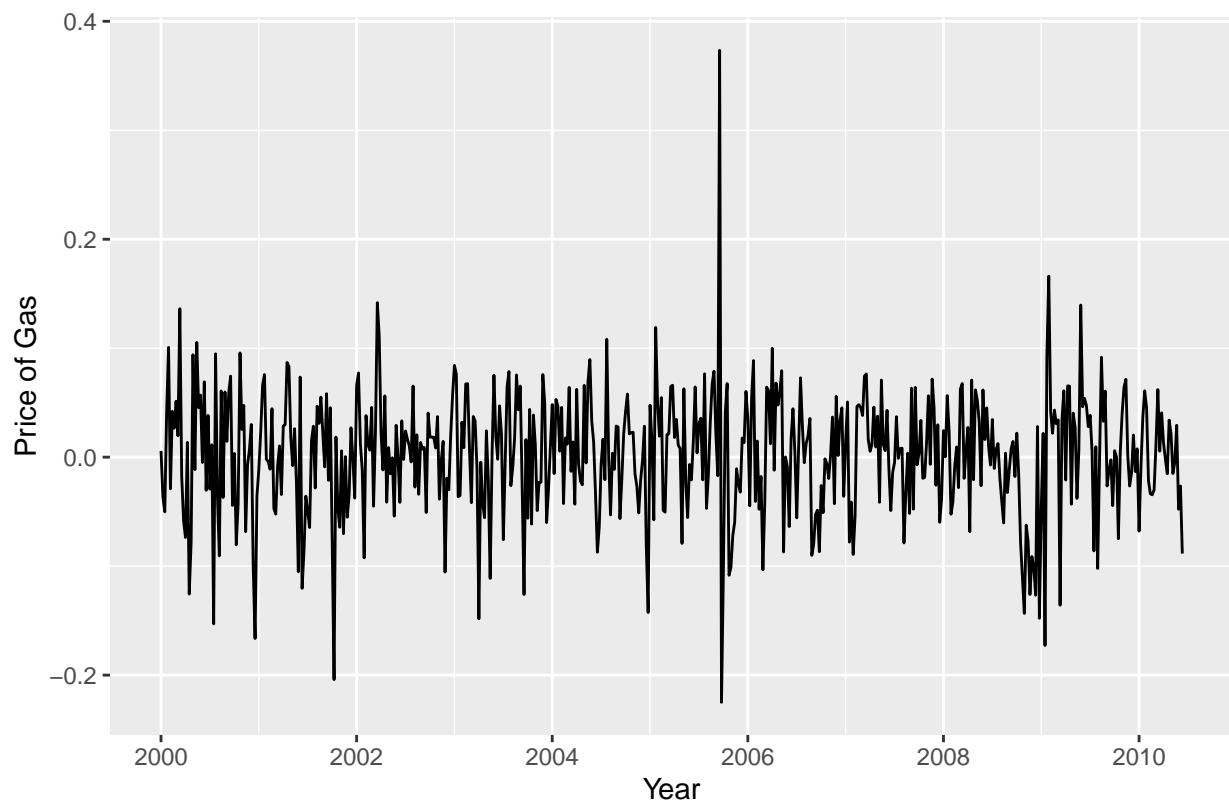
autoplot(ts(diff(log(data_oil)), differences = 1), start = 2000, frequency = 52)) +
  ylab("Price of Oil") + xlab("Year") +
  ggtitle("Price of Oil with Diff 1 vs. Years")
```

Price of Oil with Diff 1 vs. Years



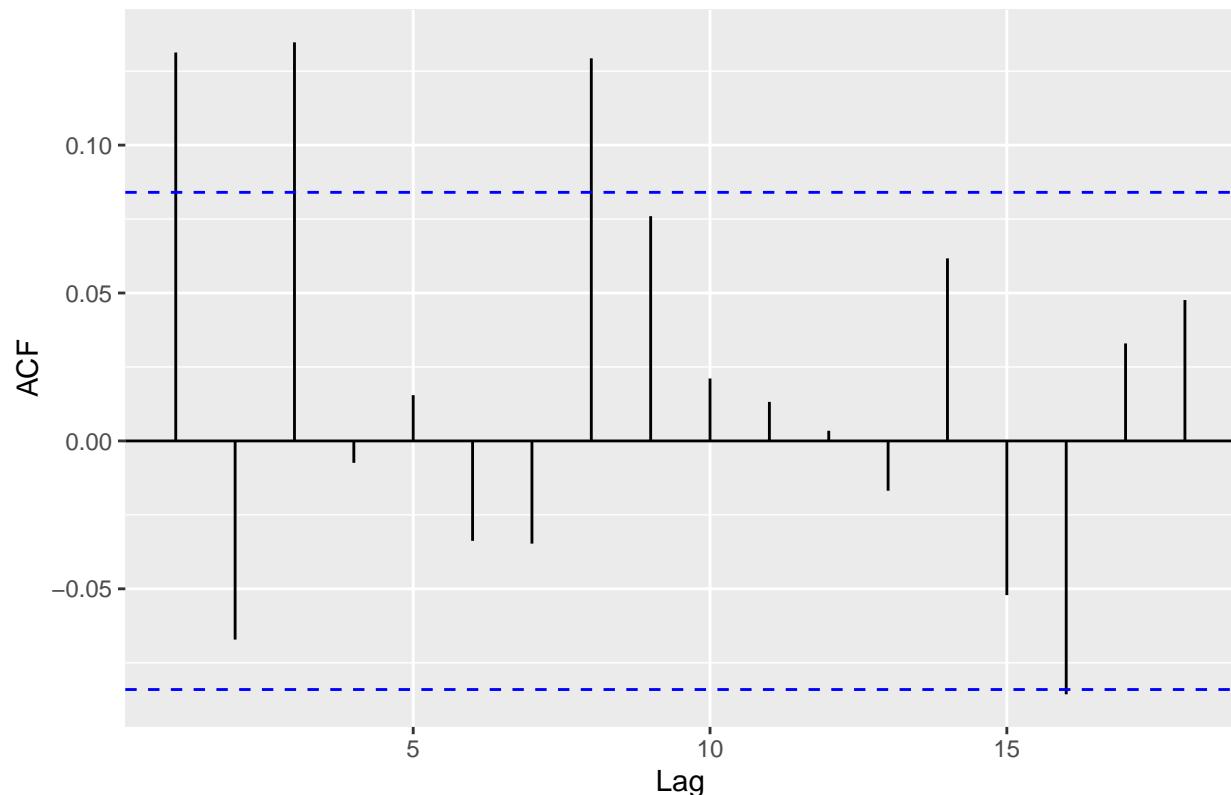
```
autoplot(ts(diff(log(data_gas), differences = 1), start = 2000, frequency = 52)) +  
  ylab("Price of Gas") + xlab("Year") +  
  ggtitle("Price of Gas with Diff 1 vs. Years")
```

Price of Gas with Diff 1 vs. Years



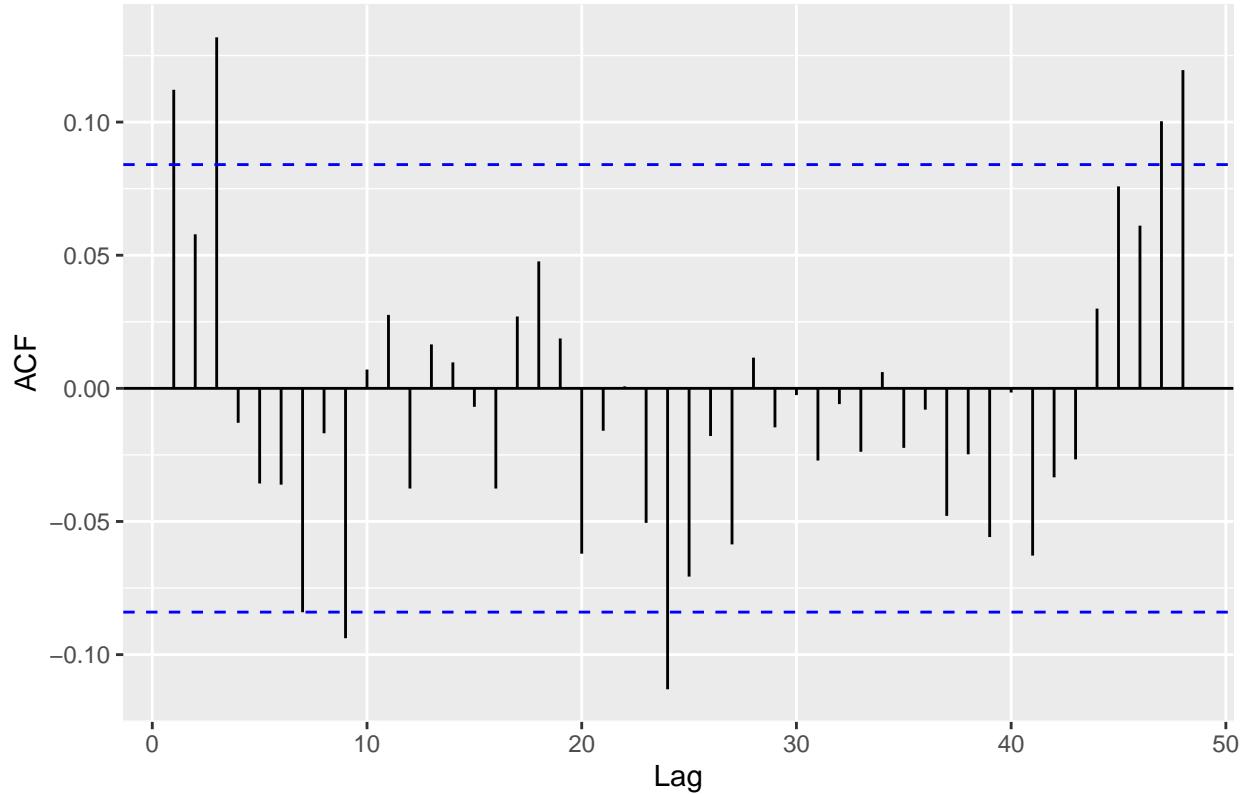
```
ggAcf(diff(log(data_oil), differences = 1), data_oil)
```

Series: diff(log(data\_oil), differences = 1)



```
ggAcf(diff(log(data_gas), differences = 1), data_gas)
```

Series: diff(log(data\_gas), differences = 1)



### Detrending using smoother

d) Exhibit scatter plots of  $x_t$  and  $y_t$  for up to three weeks of lead time of  $x_t$  include a non-parametric smoother in each plot and comment the results: are there outliers? Are the relationships linear? Are there changes in the trend?

```
set.seed(12345)

oil_price_one_diff <- diff(log(data_oil), differences = 1)
gas_price_one_diff <- diff(log(data_gas), differences = 1)

df <- data.frame(oil_price_one_diff=as.matrix(oil_price_one_diff),
                  gas_price_one_diff = as.matrix(gas_price_one_diff),
                  time=time(oil_price_one_diff))

df <- na.omit(df)

df$gas_price_one_diff = lag(df$gas_price_one_diff,1)
df$gas_price_two_diff = lag(df$gas_price_one_diff,2)
df$gas_price_three_diff = lag(df$gas_price_one_diff,3)

df <- na.omit(df)

df$smooth_one_week_lag <- ksmooth(x = df$oil_price_one_diff,
```

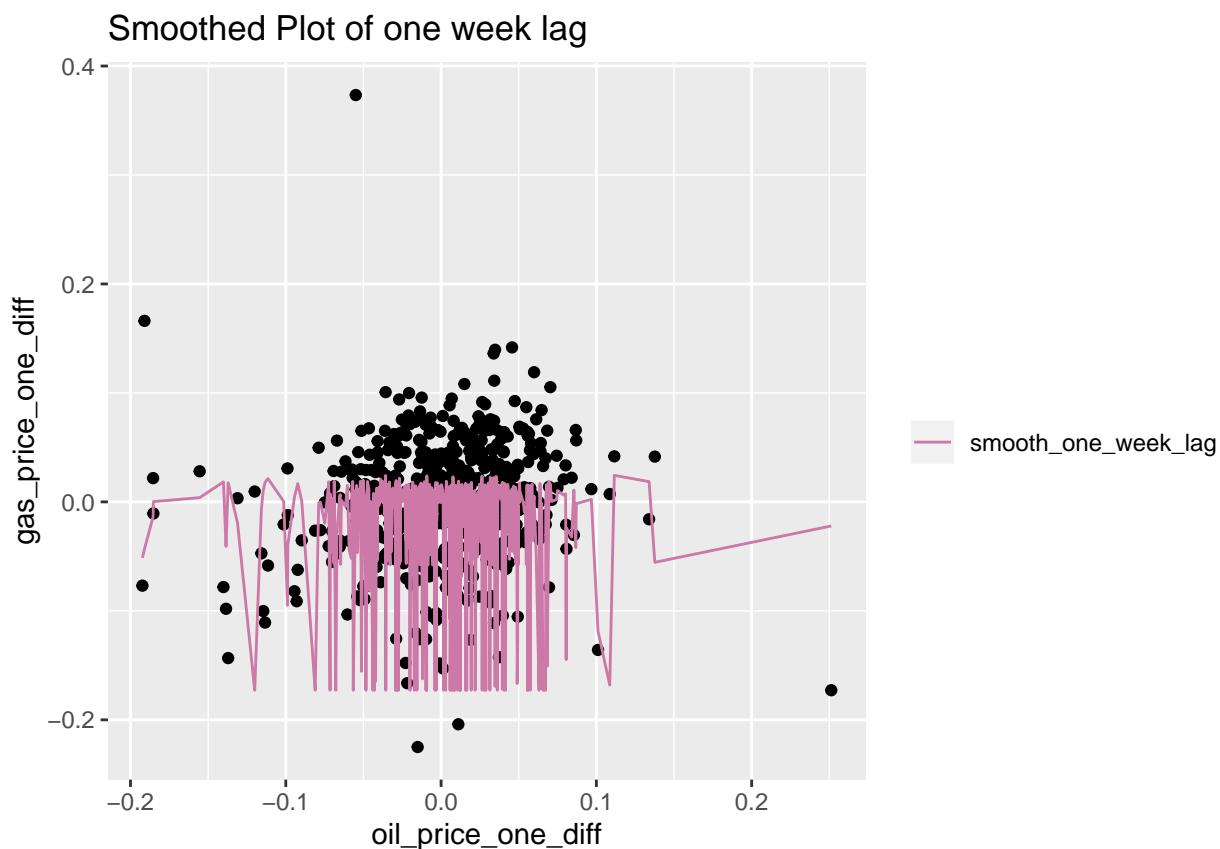
```

y = df$gas_price_one_diff,
bandwidth = 0.05, kernel = "normal")$y
df$smooth_two_week_lag <- ksmooth(x = df$oil_price_one_diff,
y = df$gas_price_two_diff,
bandwidth = 0.05, kernel = "normal")$y
df$smooth_three_week_lag <- ksmooth(x = df$oil_price_one_diff,
y = df$gas_price_three_diff,
bandwidth = 0.05, kernel = "normal")$y

df <- na.omit(df)

ggplot(data=df, aes(x=oil_price_one_diff, y = gas_price_one_diff)) + geom_point() +
  geom_line(aes(y= smooth_one_week_lag, color= "smooth_one_week_lag")) +
  scale_colour_manual("", breaks = c("smooth_one_week_lag"),
  values = c("#CC79A7")) +
  ggtitle("Smoothed Plot of one week lag")

```

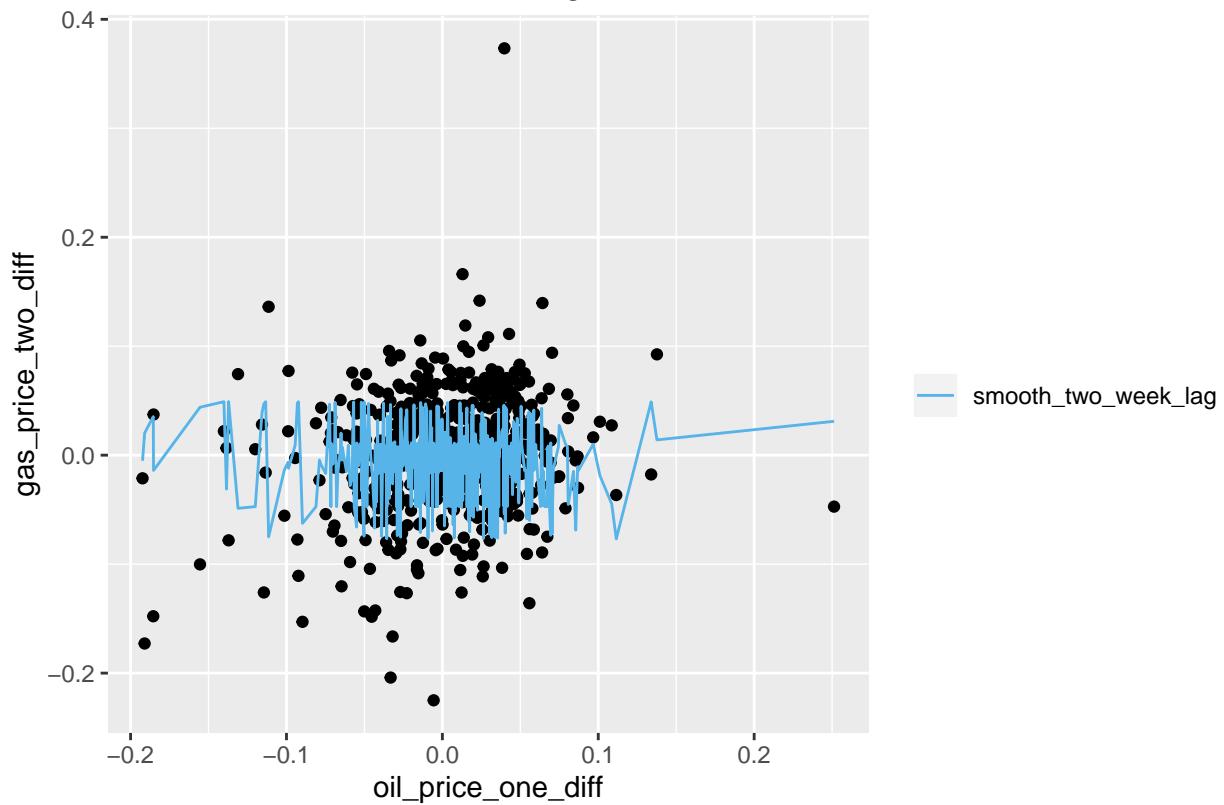


```

ggplot(data=df, aes(x=oil_price_one_diff, y = gas_price_two_diff)) + geom_point() +
  geom_line(aes(y= smooth_two_week_lag, color= "smooth_two_week_lag")) +
  scale_colour_manual("", breaks = c("smooth_two_week_lag"),
  values = c("#56B4E9")) +
  ggtitle("Smoothed Plot of two week lag")

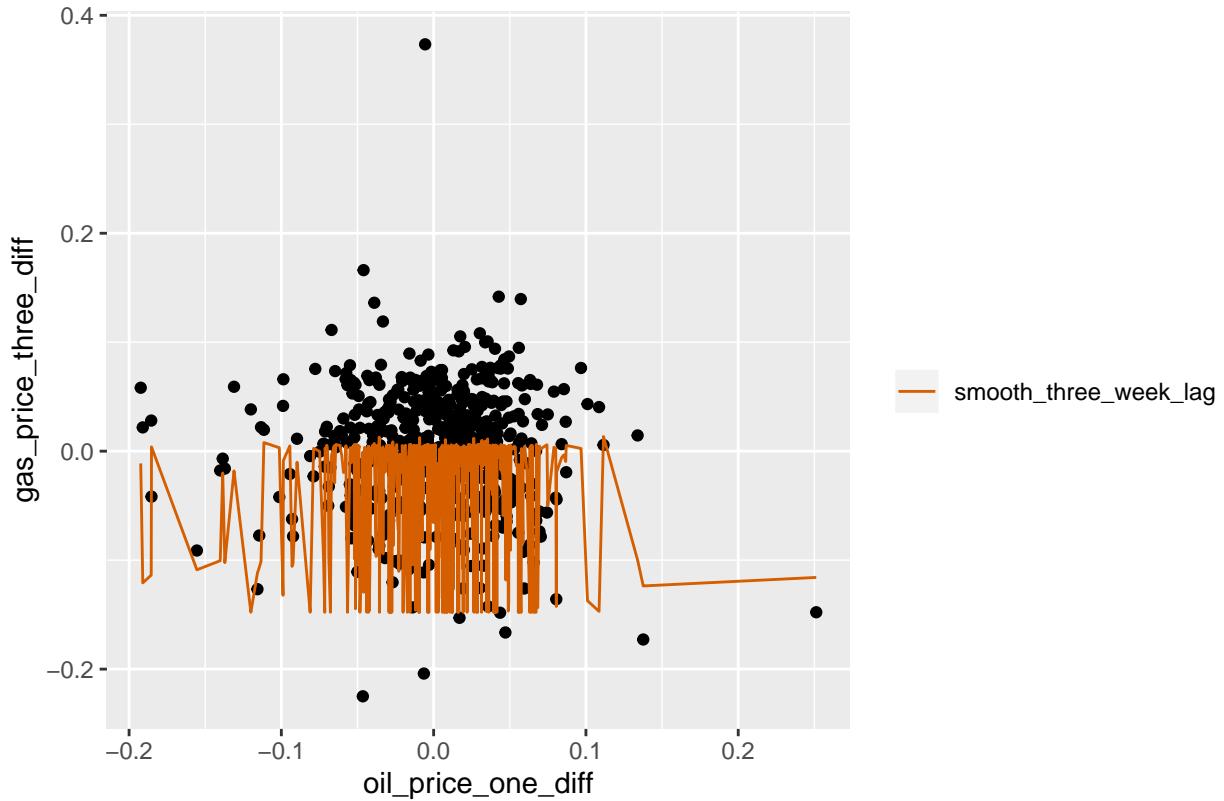
```

Smoothed Plot of two week lag



```
ggplot(data=df, aes(x=oil_price_one_diff, y = gas_price_three_diff)) + geom_point() +  
  geom_line(aes(y= smooth_three_week_lag, color= "smooth_three_week_lag")) +  
  scale_colour_manual("", breaks = c("smooth_three_week_lag"),  
                     values = c("#D55E00")) +  
  ggtitle("Smoothed Plot of three week lag")
```

Smoothed Plot of three week lag



### Detrending using linear regression

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.

```
set.seed(12345)

df$oil_price_two_diff = lag(df$oil_price_one_diff, 2)
df$x_t_more_zero <- ifelse(df$oil_price_one_diff > 0, "1", "0")
lm_model_lag <- lm(data=df, formula = gas_price_one_diff ~ x_t_more_zero +
                      oil_price_one_diff + oil_price_two_diff)
summary(lm_model_lag)
```

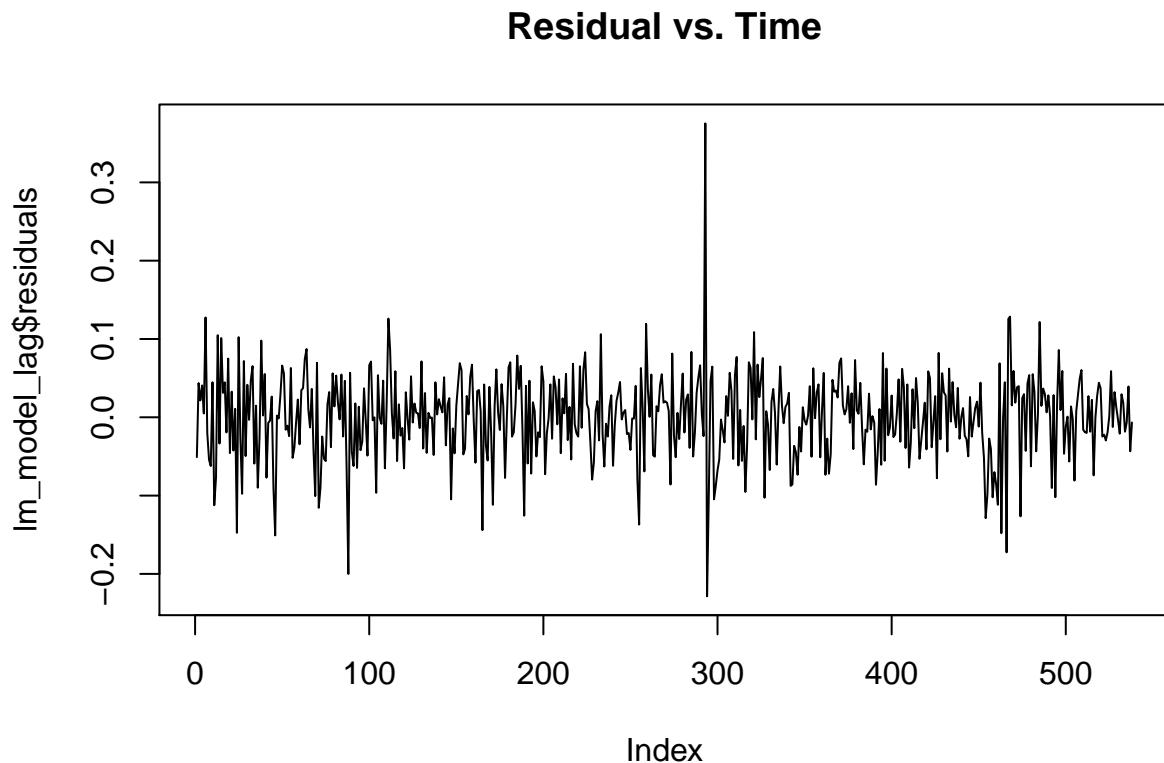
```
##
## Call:
## lm(formula = gas_price_one_diff ~ x_t_more_zero + oil_price_one_diff +
##     oil_price_two_diff, data = df)
##
## Residuals:
##      Min        1Q    Median        3Q       Max 
## -0.22855 -0.03191  0.00366  0.03692  0.37535 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  0.00366   0.03692   0.099    0.921    
## x_t_more_zero  0.00366   0.03692   0.099    0.921    
## oil_price_one_diff  0.03692   0.03692   1.000    0.314    
## oil_price_two_diff  0.03692   0.03692   1.000    0.314    
```

```

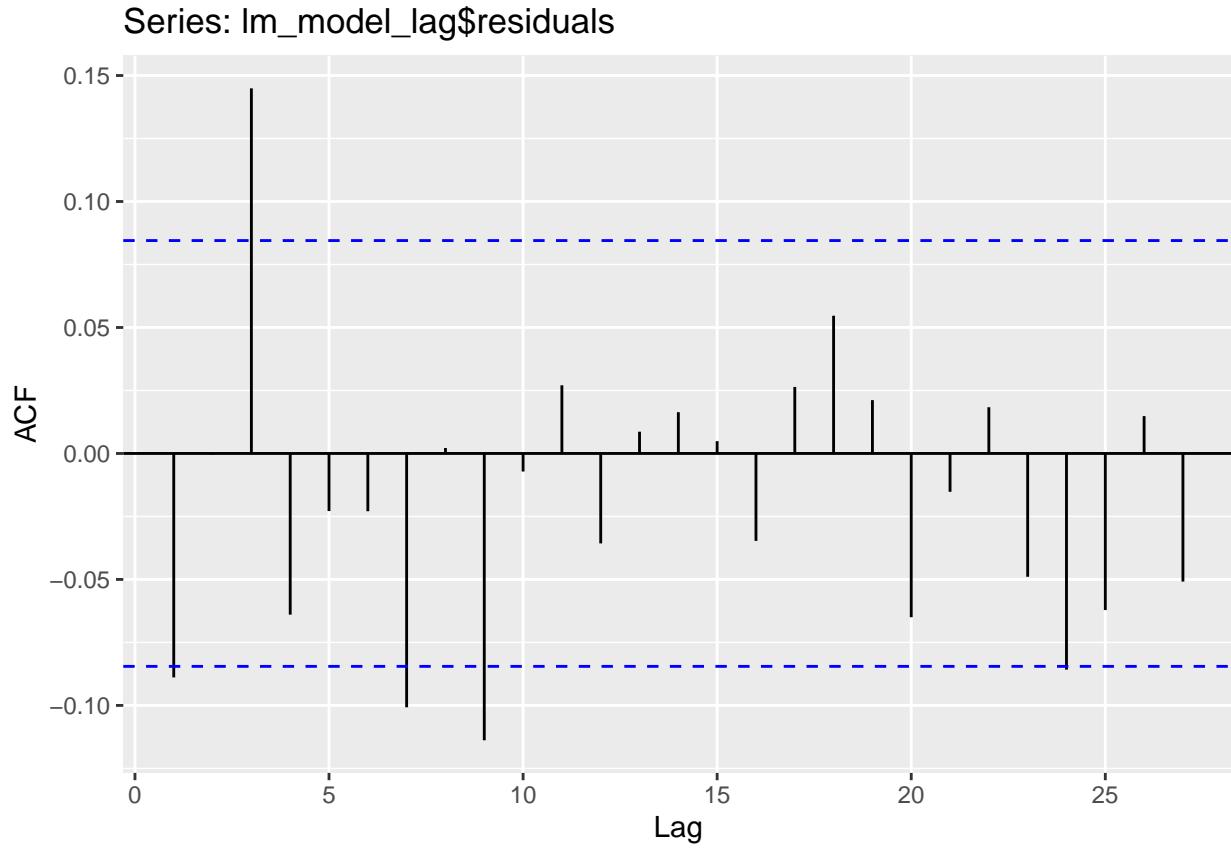
## (Intercept)      -0.002139   0.004588  -0.466     0.641
## x_t_more_zero1   0.006265   0.007304   0.858     0.391
## oil_price_one_diff  0.084851   0.077622   1.093     0.275
## oil_price_two_diff  0.222373   0.050788   4.378  0.0000144 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05511 on 534 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.04571,    Adjusted R-squared:  0.04035
## F-statistic: 8.526 on 3 and 534 DF,  p-value: 0.00001538

```

```
plot(lm_model_lag$residuals, type = 'l', main="Residual vs. Time")
```



```
ggAcf(lm_model_lag$residuals)
```



## Linear Regressions on Necessarily Lagged Variables and Appropriate Correlation

Generate AR and using PACF

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  $\phi_{33}$  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  $\phi_{33}$ .

$$\phi_{33} = \text{corr}(X_{t-3} - f_p, X_t - f_p) \text{ where } f_p = \sum_{j=1}^p \phi_j X_{\tau-j}$$

```
set.seed(12345)
x_t <- arima.sim(model = list(ar = c(0.8, -0.2, 0.1)), n=1000)
actual_pacf_value <- pacf(x_t, plot = FALSE)$acf[3]
df <- data.frame(x_t = as.vector(x_t))
df$x_t_lag_1 <- lag(df$x_t, 1)
df$x_t_lag_2 <- lag(df$x_t, 2)
df$x_t_lag_3 <- lag(df$x_t, 3)
df <- na.omit(df)

# building models and getting their residuals
model_1_res <- lm(x_t ~ x_t_lag_1 + x_t_lag_2, data = df)$residuals
```

```

model_2_res <- lm(x_t_lag_3 ~ x_t_lag_1 + x_t_lag_2, data = df)$residuals

# theoretical pacf values
theotical_pacf_value <- cor(x = model_1_res, y = model_2_res,
                             use = "na.or.complete")

cat("The theoretical and actual value of PACF are: ", theoretical_pacf_value,
    actual_pacf_value)

```

## The theoretical and actual value of PACF are: 0.1146076 0.1170643

## Methods of Moments, Conditional Least Squares and Maximum Likelihood

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  $\phi_2$  fall within confidence interval for ML estimate?

```

set.seed(12345)
x_t <- arima.sim(model = list(ar = c(0.8,0.1)), n=100)

method_yule_walker <- ar(x_t, order = 2, method = "yule-walker", aic = FALSE)$ar
method_cls <- ar(x_t, order = 2, method = "ols", aic = FALSE)$ar
method_mle <- ar(x_t, order = 2, method = "mle", aic = FALSE)$ar

df <- data.frame(rbind(method_yule_walker, method_cls, method_mle))

kable(df, caption = "Comparison of parameters using different methods")

```

Table 1: Comparison of parameters using different methods

	ar1	ar2
method_yule_walker	0.8029146	0.1037053
method_cls	0.8066782	0.1205352
method_mle	0.7968774	0.1189369

```

# Since variance is not given by ar we use arima function
ML_Model_CI = arima(x_t, order = c(2,0,0), method = "ML")
sigma = ML_Model_CI$var.coef[2, 2]
phi_2 = ML_Model_CI$coef[2]
CI = c(phi_2 - 1.96 * sigma, phi_2 + 1.96 * sigma)
CI

##          ar2          ar2
## 0.09924714 0.13846032

```

# ARIMA

## Sample and Theoretical ACF and PACF

c) Generate 200 observations of a seasonal ARIMA(0,0,1)  $\times$  (0,0,1)<sub>12</sub> 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?

Now  $ARIMA(1,1,1)(1,1,1)_4$  can be written as  $(1 - \phi_1 B)(1 - B)(1 - B^4)(1 - \Phi_1 B^4)x_t = w_t(1 + \theta B)(1 + \Theta B^4)$   
Similarly  $ARIMA(0,0,1)(0,0,1)_{12}$  can be written as  $x_t = w_t(1 + \Theta B^{12})(1 + \theta B)$  which can be simplified as  $x_t = w_t(1 + \Theta B^{12} + \theta B + \Theta \theta B^{13})$  given that  $\theta = 0.3$  and  $\Theta = 0.6$  we get  $x_t = w_t(1 + 0.3B + 0.6B^{12} + 0.18B^{13})$

```
set.seed(12345)
x_t <- arima.sim(model = list(ma = c(0.3,rep(0,10),0.6,0.18)), n=200)

df <- data.frame(sample_acf = acf(x_t, plot = FALSE, lag.max = 14)$acf,
                  sample_pacf = pacf(x_t, plot = FALSE, lag.max = 14)$acf,
                  theoritical_acf = ARMAacf(ma = c(0.3,rep(0,10),0.6,0.18),
                                              pacf = FALSE, lag.max = 13),
                  theoritical_pacf = ARMAacf(ma = c(0.3,rep(0,10),0.6,0.18),
                                              pacf = TRUE, lag.max = 14))

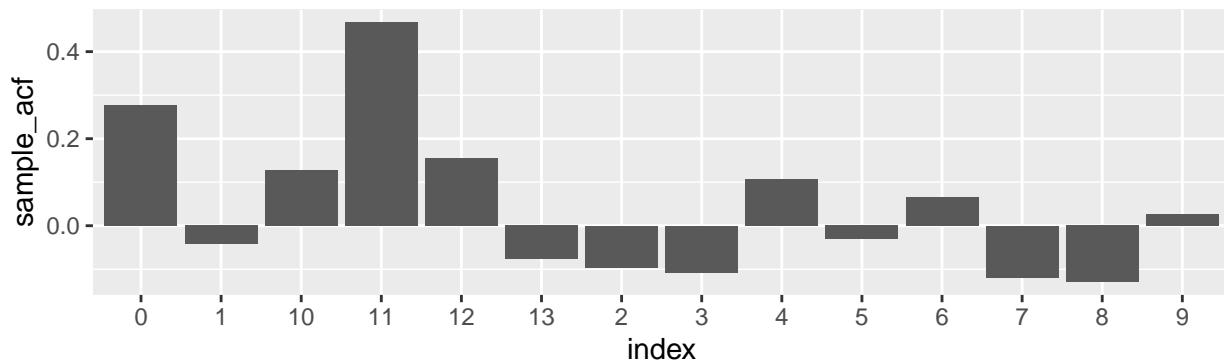
df$index <- rownames(df)

plot1 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=sample_acf)) +
  ggtitle("Sample ACF")

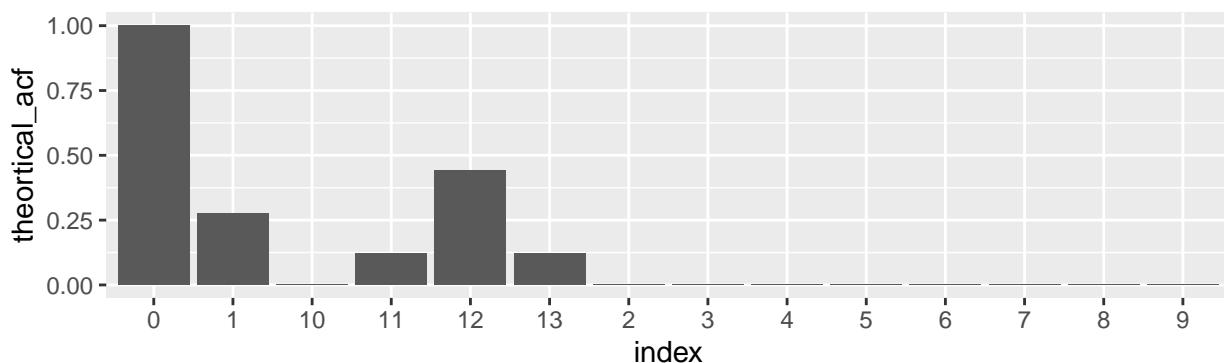
plot2 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=theoritical_acf)) +
  ggtitle("Theoretical ACF")

grid.arrange(plot1, plot2, ncol = 1)
```

Sample ACF



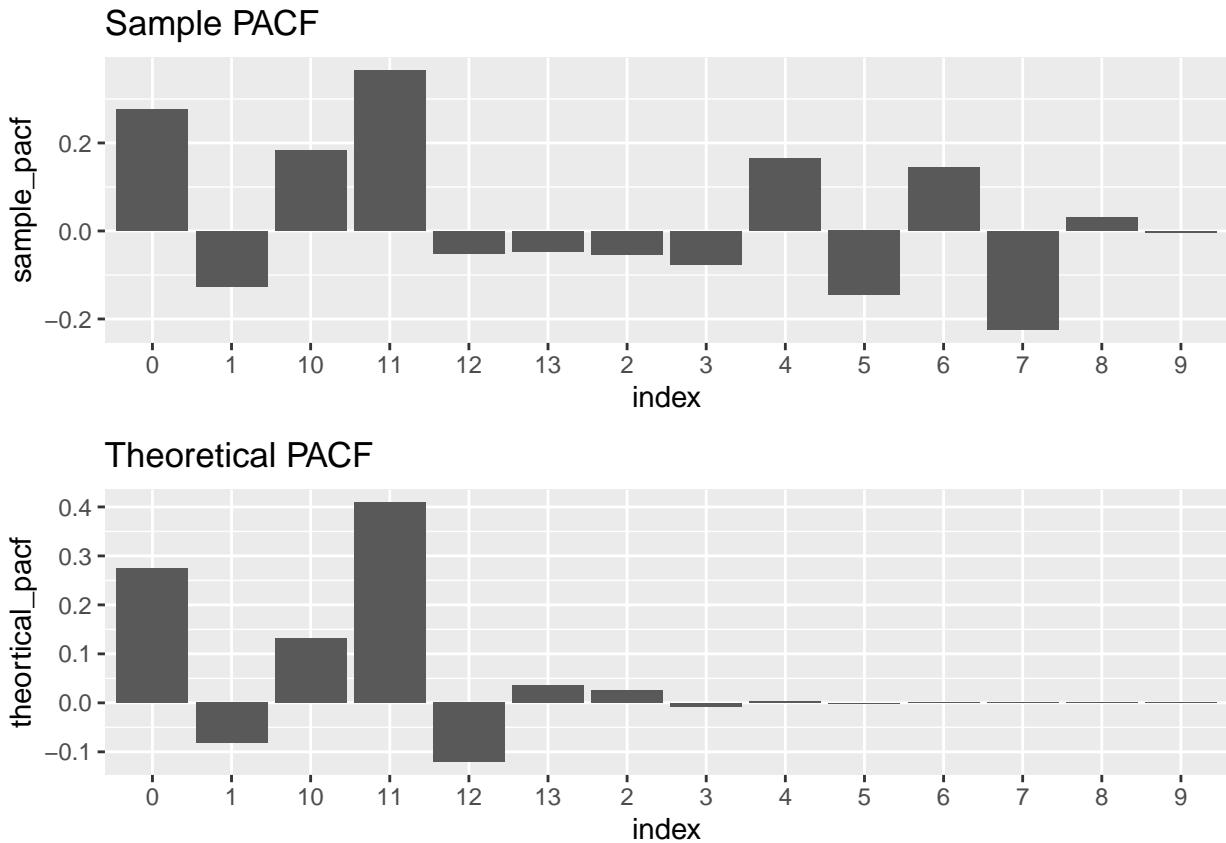
Theoretical ACF



```
plot3 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=sample_pacf)) +
  ggtitle("Sample PACF")

plot4 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=theoretical_pacf)) +
  ggtitle("Theoretical PACF")

grid.arrange(plot3, plot4, ncol = 1)
```



## Forecast and Prediction

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.

```

set.seed(12345)
x_t <- arima.sim(model = list(ma = c(0.3, rep(0, 10), 0.6, 0.18)), n=200)
fit_x_t <- arima(x_t, order = c(0, 0, 1), seasonal = list(order = c(0, 0, 1),
                                                               period = 12))
predicted_x_t <- predict(fit_x_t, n.ahead=30)
predicted_x_t_upper_band <- predicted_x_t$pred + 1.96 * predicted_x_t$se
predicted_x_t_lower_band <- predicted_x_t$pred - 1.96 * predicted_x_t$se

#kernlab

df <- data.frame(y = x_t)
df$x <- as.numeric(rownames(df))
gausspr_model <- gausspr(x=df$x, y=df$y)

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

```

```

predicted_x_t_kernlab <- predict(gausspr_model, newdata=data.frame(x=201:230))

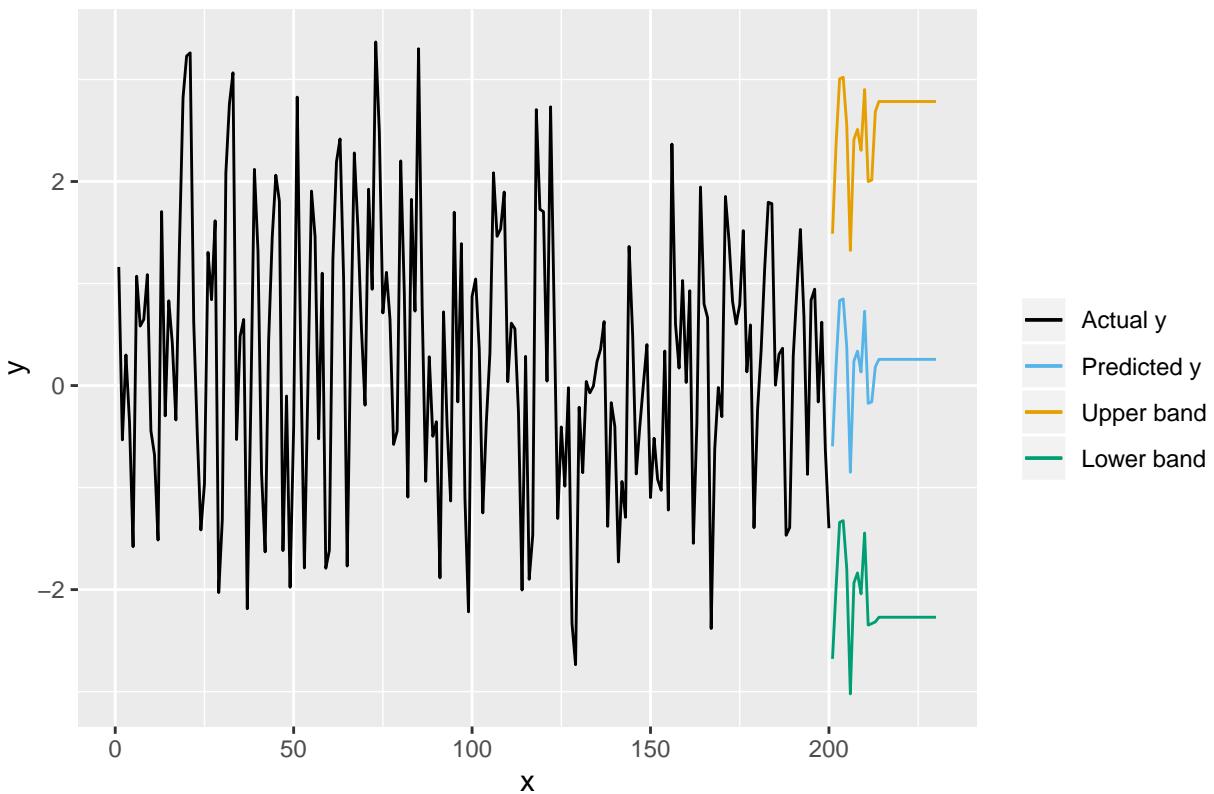
df3 <- data.frame(y = predicted_x_t_kernlab, x=201:230)

df2 <- data.frame(predicted_x_t = predicted_x_t$pred,
                   predicted_x_t_upper = predicted_x_t_upper_band,
                   predicted_x_t_lower = predicted_x_t_lower_band,
                   x = 201:230)

ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
  geom_line(data=df2, aes(x=x, y=predicted_x_t, color="Predicted y")) +
  geom_line(data=df2, aes(x=x, y=predicted_x_t_upper, color="Upper band")) +
  geom_line(data=df2, aes(x=x, y=predicted_x_t_lower, color="Lower band")) +
  scale_colour_manual("", breaks = c("Actual y", "Predicted y", "Upper band", "Lower band"),
                      values = c("#000000", "#009E73", "#56B4E9", "#E69F00")) +
  ggtitle("Original vs. Predicted y with confidence bands")

```

Original vs. Predicted y with confidence bands

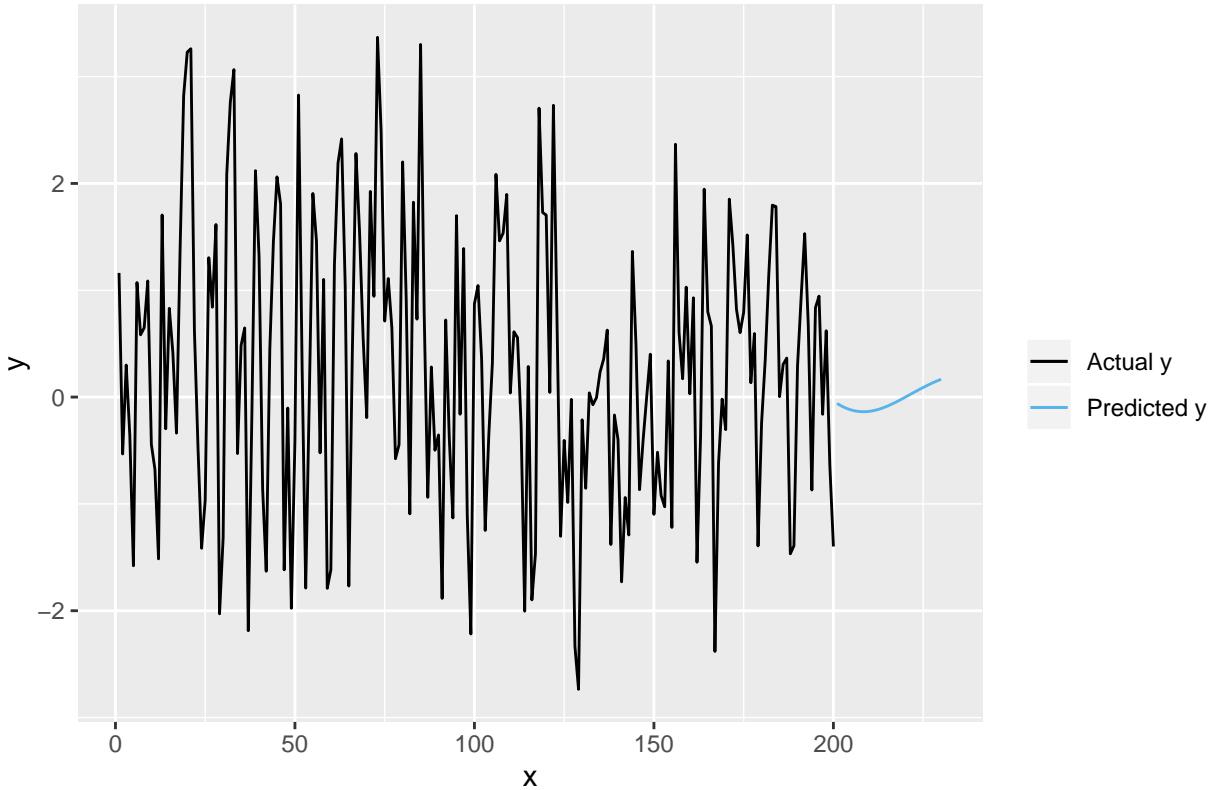


```

ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
  geom_line(data=df3, aes(x=x, y=y, color="Predicted y")) +
  scale_colour_manual("", breaks = c("Actual y", "Predicted y"),
                      values = c("#000000", "#56B4E9")) +
  ggtitle("Original vs. Predicted y using gausspr")

```

## Original vs. Predicted y using gausspr



## Prediction Band

e) Generate 50 observations from ARMA(1, 1) process with  $\phi = 0.7$ ,  $\theta = 0.50$ . 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?

```
x_t <- arima.sim(model = list(ar = c(0.7), ma=c(0.5)), n=50)
fit_x_t <- arima(x_t[1:40], order = c(1,0,1), include.mean = 0)

predicted_x_t <- predict(fit_x_t, n.ahead=10)
predicted_x_t_upper_band <- predicted_x_t$pred + 1.96 * predicted_x_t$se
predicted_x_t_lower_band <- predicted_x_t$pred - 1.96 * predicted_x_t$se

df <- data.frame(y = x_t[1:40], x=1:40)
df2 <- data.frame(y = predicted_x_t$pred,
                    upper_band=predicted_x_t_upper_band,
                    lower_band=predicted_x_t_lower_band,
                    x = 41:50)

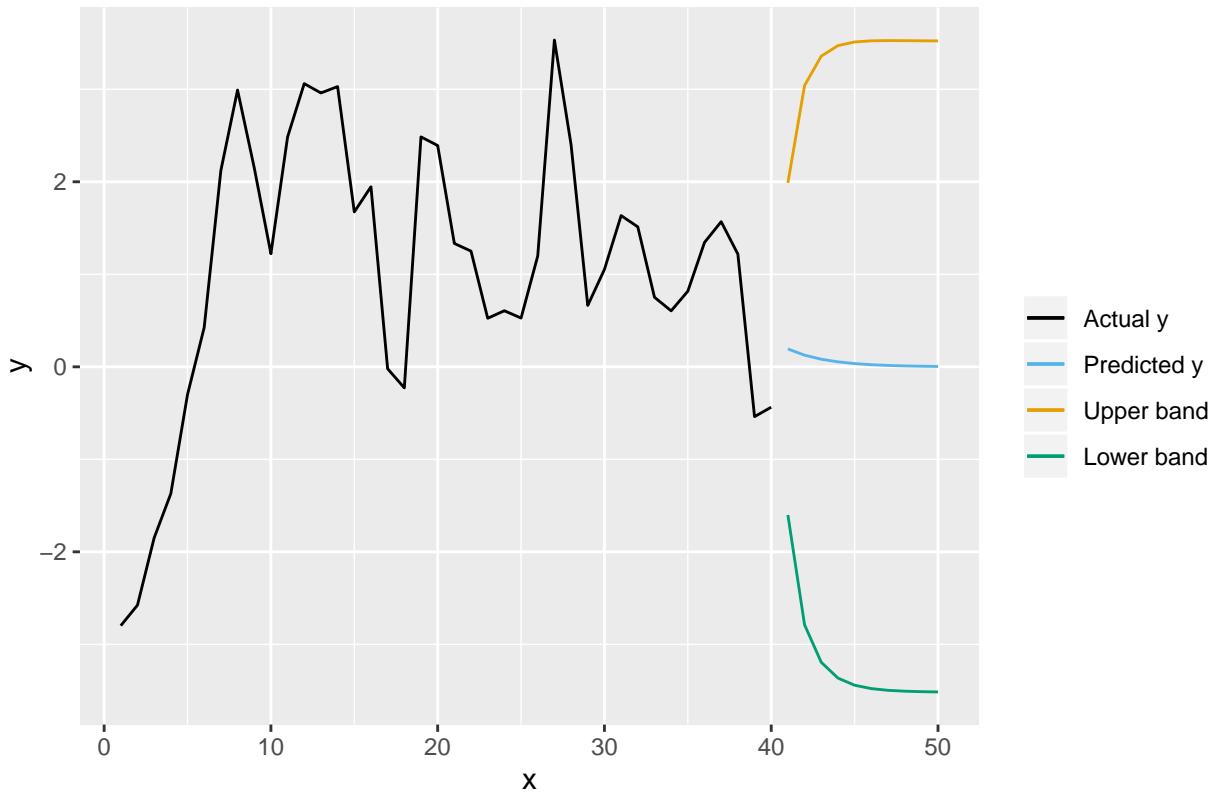
ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
  geom_line(data=df2, aes(x=x, y=y, color="Predicted y")) +
  geom_line(data=df2, aes(x=x, y=upper_band, color="Upper band")) +
  geom_line(data=df2, aes(x=x, y=lower_band, color="Lower band")) +
  scale_colour_manual("", breaks = c("Actual y", "Predicted y", "Upper band", "Lower band"),
```

```

values = c("#000000", "#009E73", "#56B4E9", "#E69F00")) +
ggttitle("Original vs. Predicted y with confidence bands")

```

Original vs. Predicted y with confidence bands



## Finding a Suitable ARIMA Model and EACF

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.

```

set.seed(12345)

# visualization
autoplots(ts(oil, start = 2000, frequency = 52)) +
  ylab("Price of Oil") + xlab("Year") +
  ggttitle("Price of Oil vs. Years")

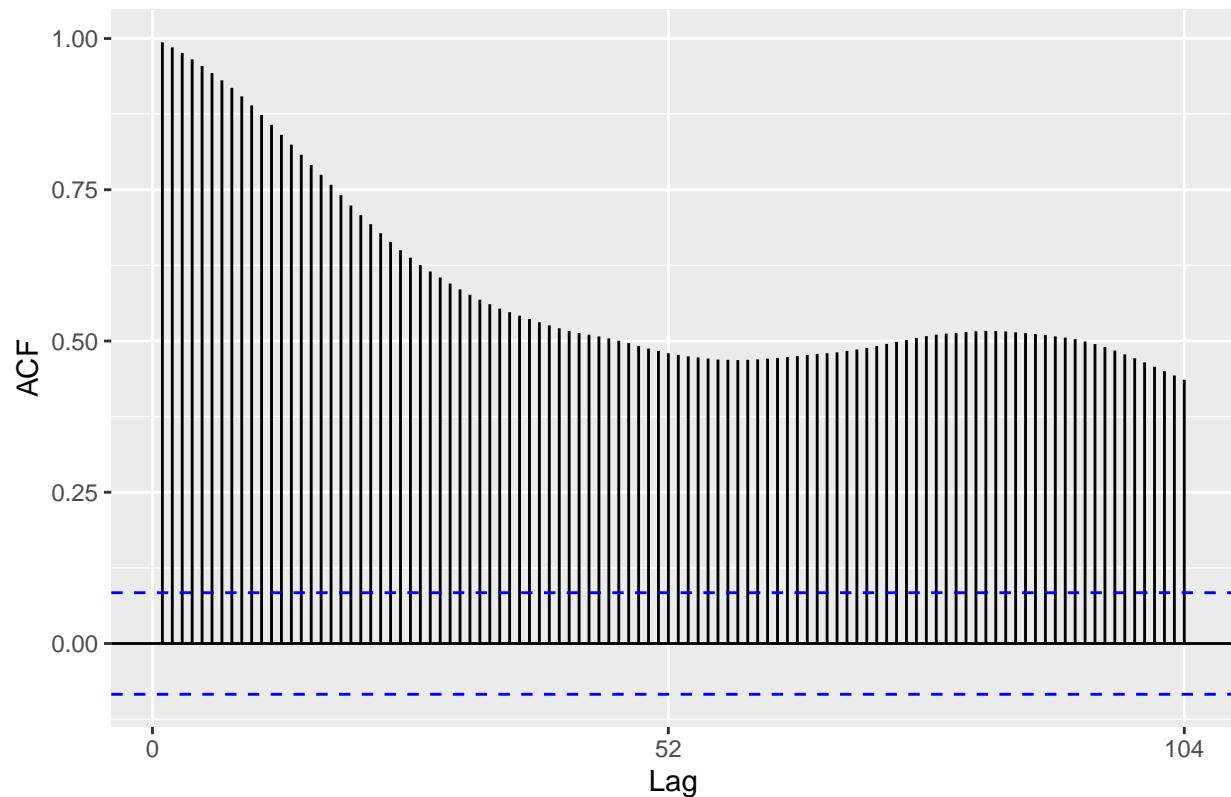
```

## Price of Oil vs. Years



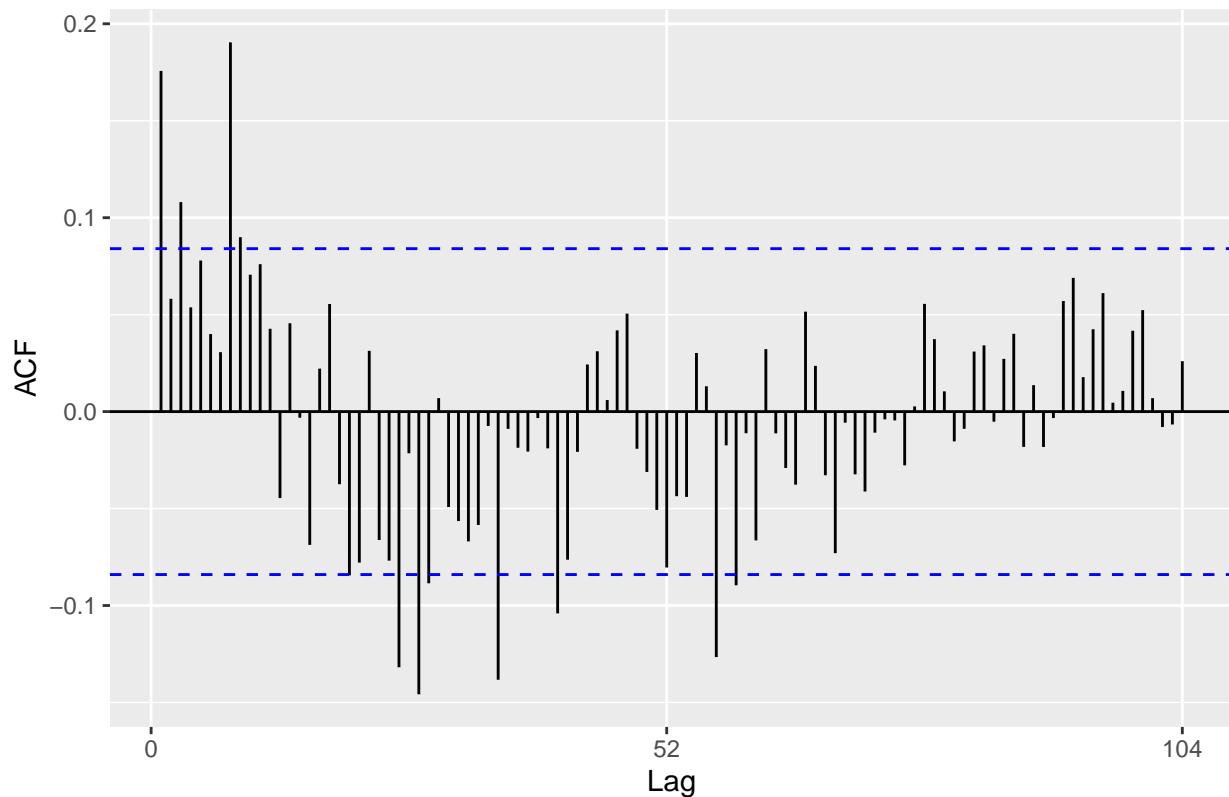
```
ggAcf(oil) + ggtitle("ACF for Oil")
```

ACF for Oil



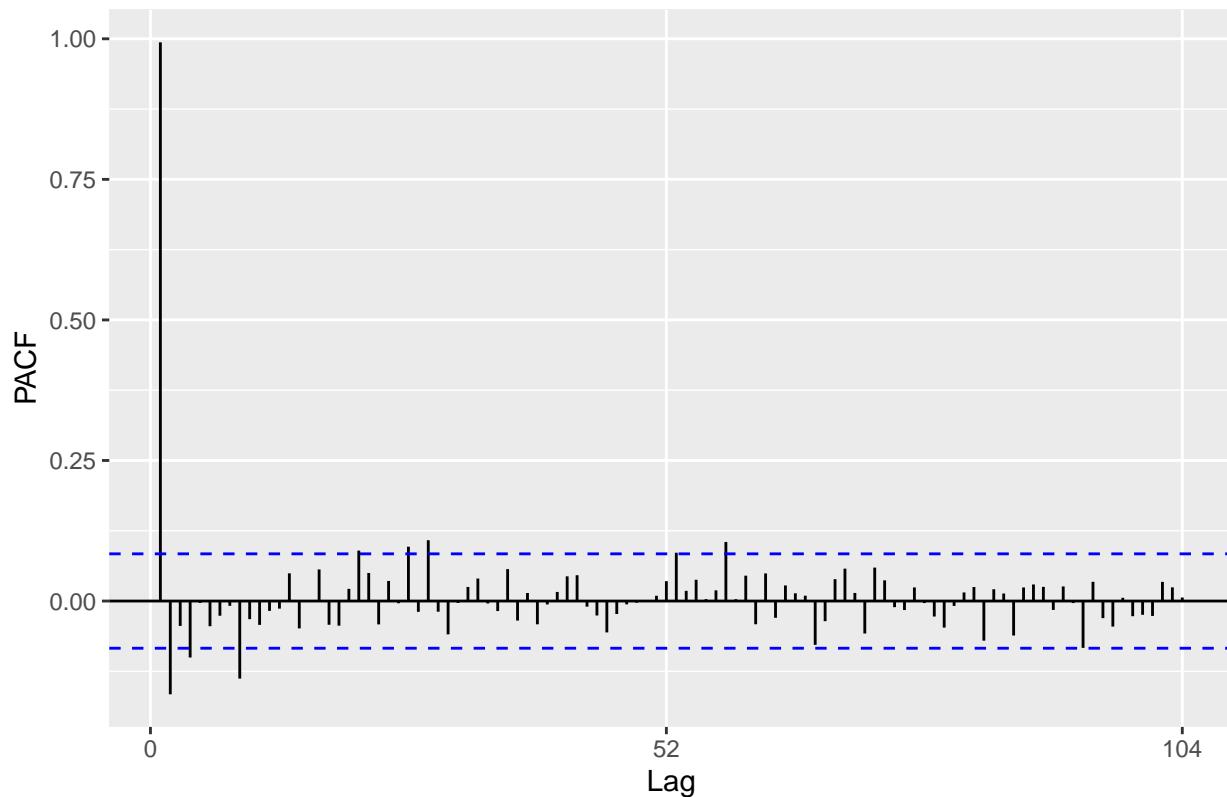
```
ggAcf(diff(oil)) + ggtitle("ACF for Oil with one diff")
```

ACF for Oil with one diff



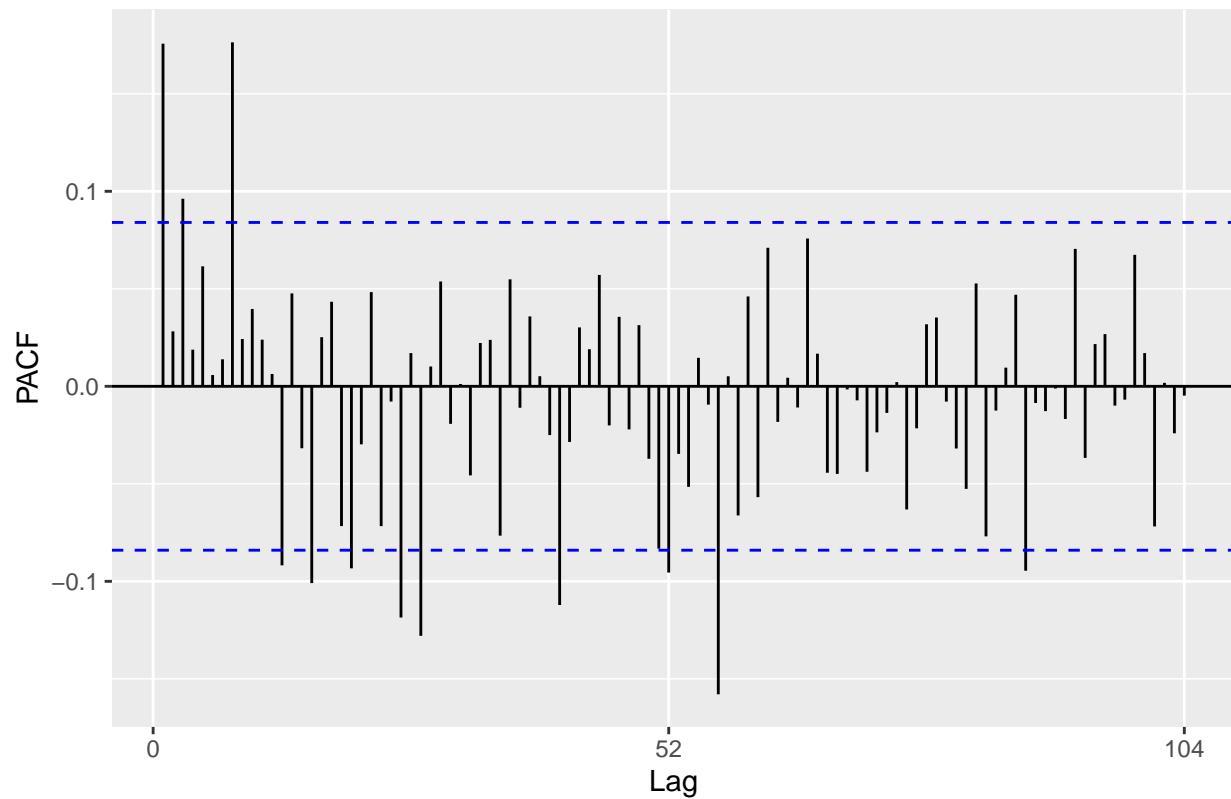
```
ggPacf(oil) + ggtitle("PACF for Oil")
```

PACF for Oil



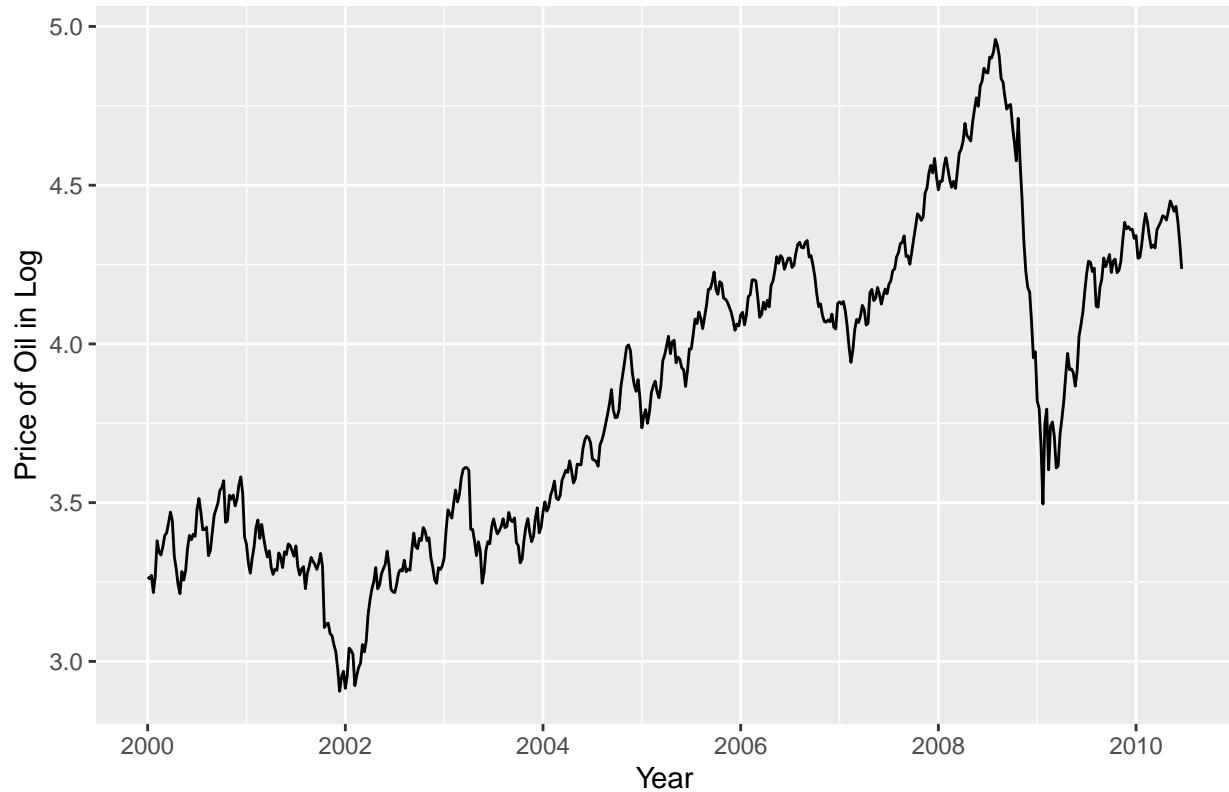
```
ggPacf(diff(oil)) + ggtitle("PACF for Oil with one diff")
```

## PACF for Oil with one diff



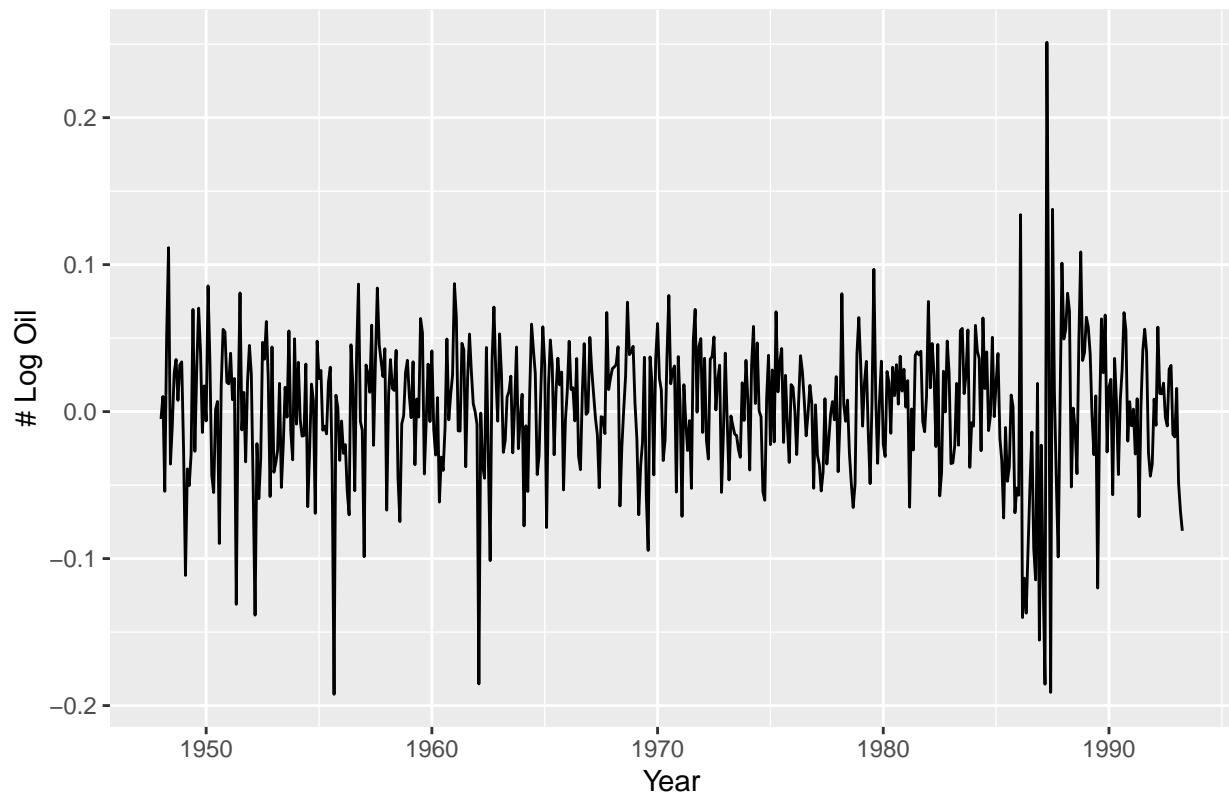
```
# with log
autoplot(ts(log(oil), start = 2000, frequency = 52)) +
  ylab("Price of Oil in Log") + xlab("Year") +
  ggtitle("Price of Log Oil vs. Years")
```

Price of Log Oil vs. Years



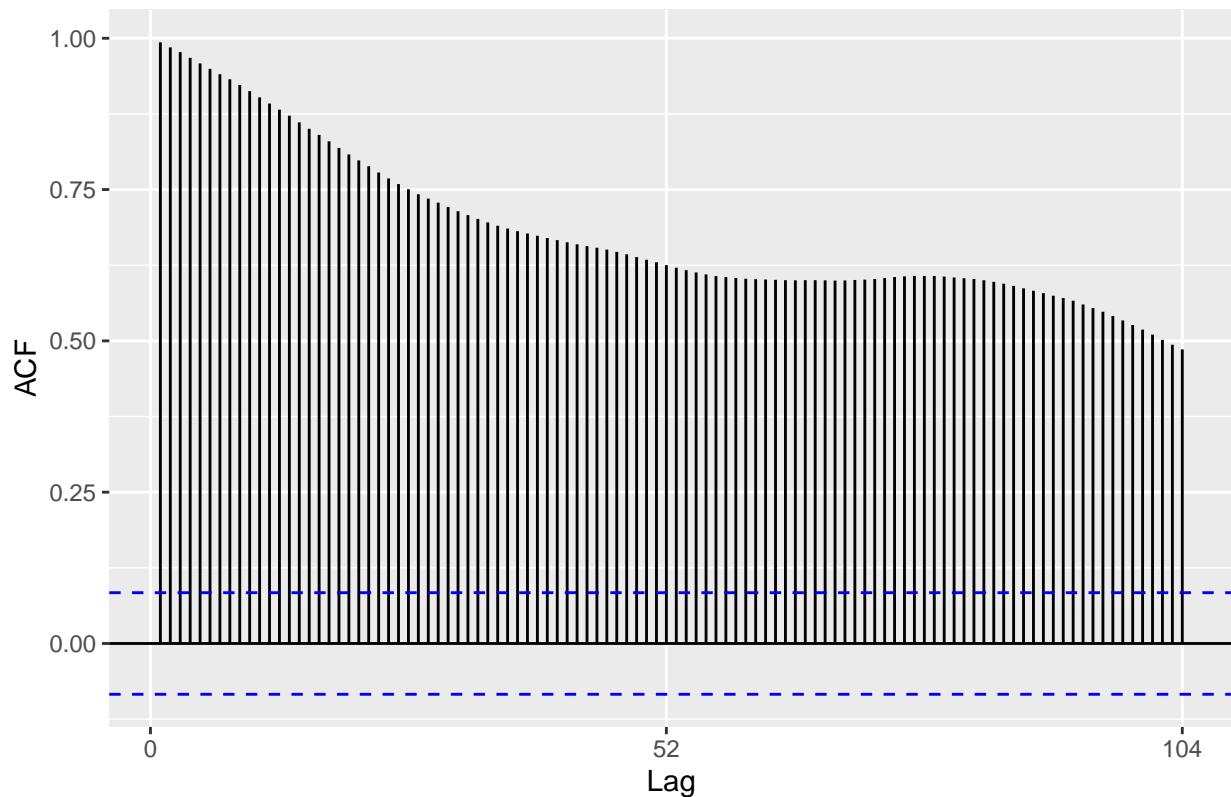
```
autoplot(ts(diff(log(oil), lag=1), start = 1948, frequency = 12)) +
  ylab("# Log Oil") + xlab("Year") +
  ggtitle("Price of log oil with one lags vs. Years")
```

Price of log oil with one lags vs. Years



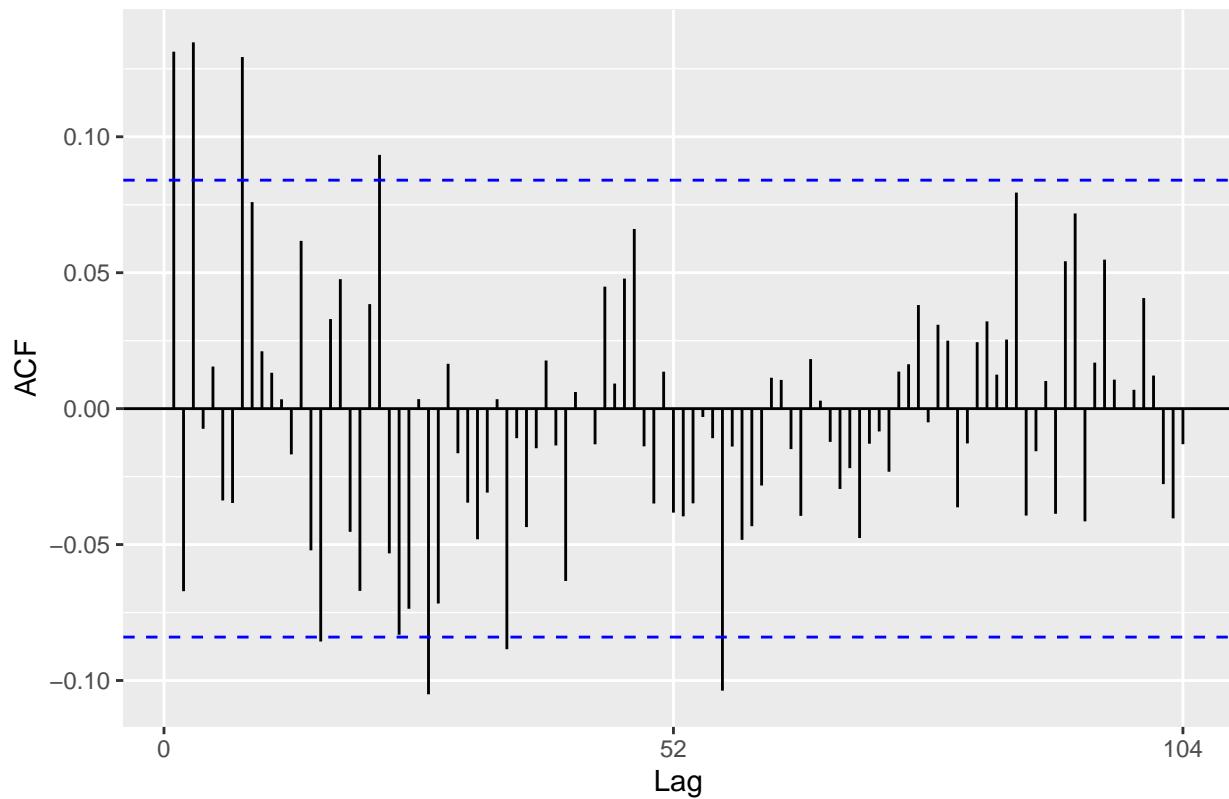
```
ggAcf(log(oil)) + ggttitle("ACF for log Oil")
```

ACF for log Oil



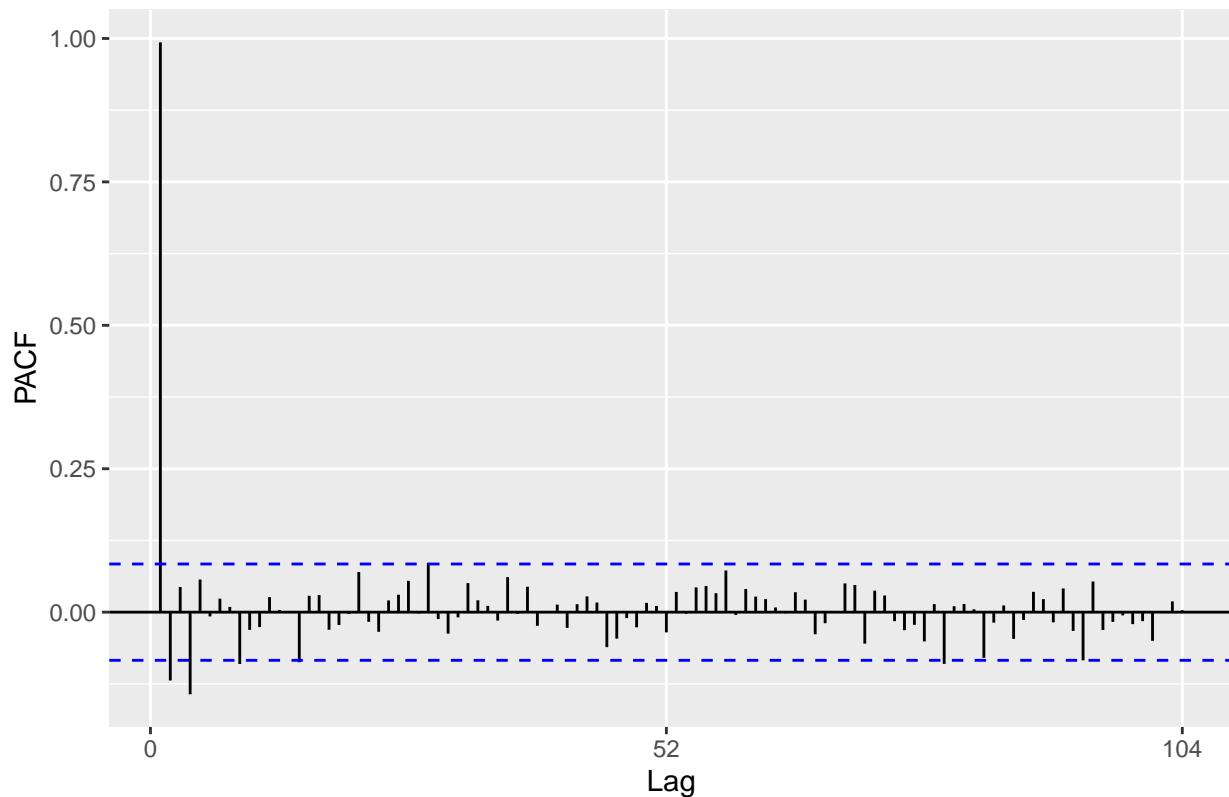
```
ggAcf(diff(log(oil))) + ggttitle("ACF for log Oil with one diff")
```

ACF for log Oil with one diff



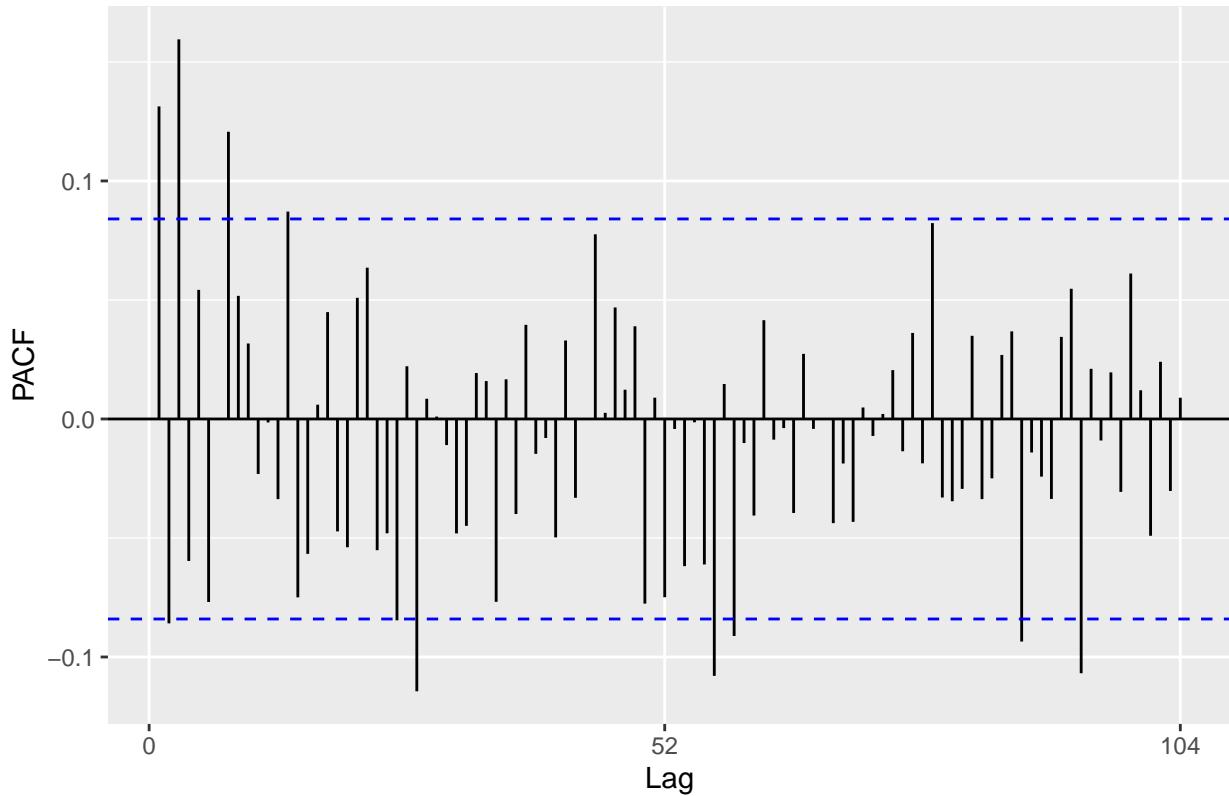
```
ggPacf(log(oil)) + ggtitle("PACF for log Oil")
```

PACF for log Oil



```
ggPacf(diff(log(oil))) + ggttitle("PACF for log Oil with one diff")
```

## PACF for log Oil with one diff



```
# EACF
eacf(diff(log(oil)))
```

```
## 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 x o o o o o o o
## 2 x x x o o o o o x o o o o o o o
## 3 x x x o o o o o x o o o o o o o
## 4 x o x 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
## 6 o x x o x x o x o o o o o o x
## 7 o x x x x x x x o x o o o o o o
```

Analysis: ARIMA(0,1,1) or ARIMA(1,1,1) or ARIMA(0,1,3) according to EACF

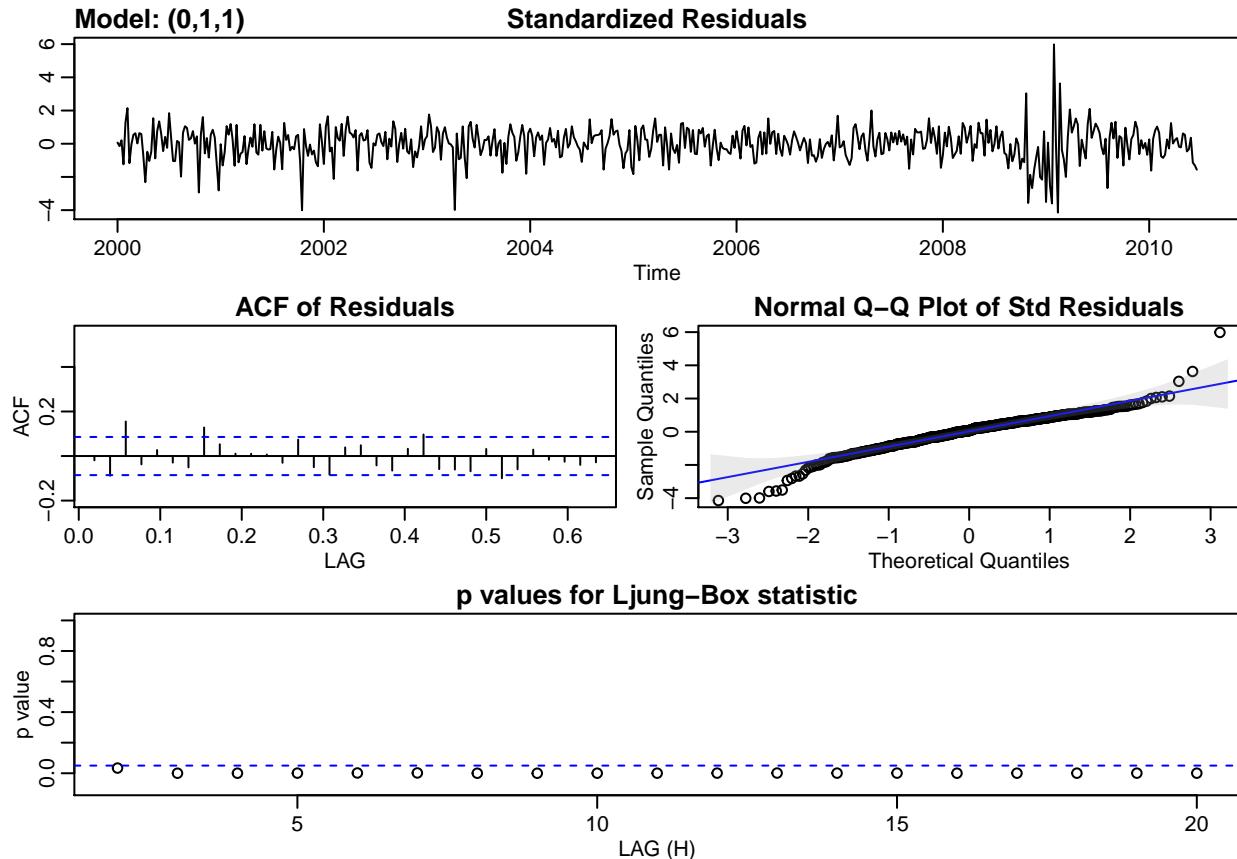
```
#Suggested Models
modelA <- sarima(log(oil), 0,1,1)
```

```
## initial  value -3.058495
## iter    2 value -3.068906
## iter    3 value -3.069474
## iter    4 value -3.069476
## iter    4 value -3.069476
## iter    4 value -3.069476
```

```

## final value -3.069476
## converged
## initial value -3.069450
## iter 2 value -3.069450
## iter 2 value -3.069450
## iter 2 value -3.069450
## final value -3.069450
## converged

```



```
modelB <- sarima(log(oil), 1,1,1)
```

```

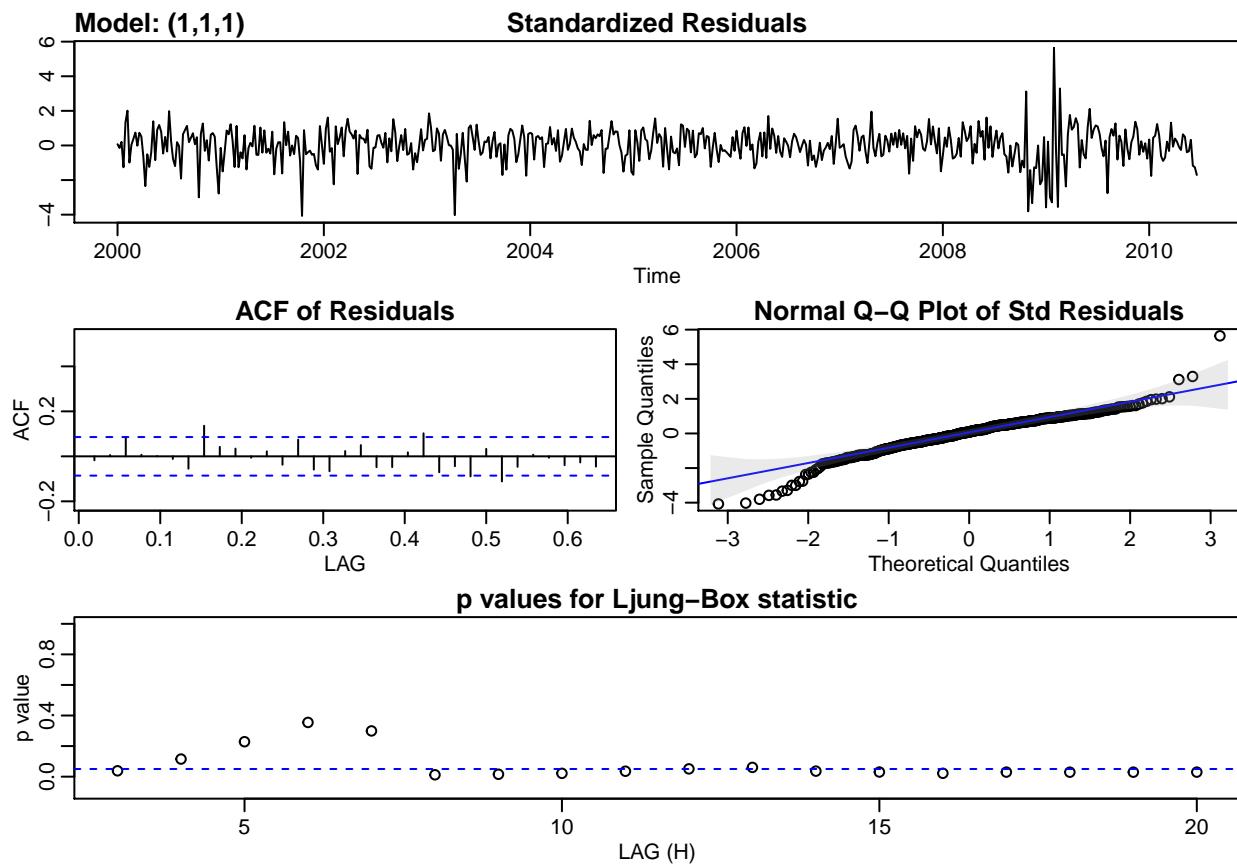
## initial value -3.057594
## iter 2 value -3.061420
## iter 3 value -3.067360
## iter 4 value -3.067479
## iter 5 value -3.071834
## iter 6 value -3.074359
## iter 7 value -3.074843
## iter 8 value -3.076656
## iter 9 value -3.080467
## iter 10 value -3.081546
## iter 11 value -3.081603
## iter 12 value -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

```



```
modelC <- sarima(log(oil), 0,1,3)
```

```

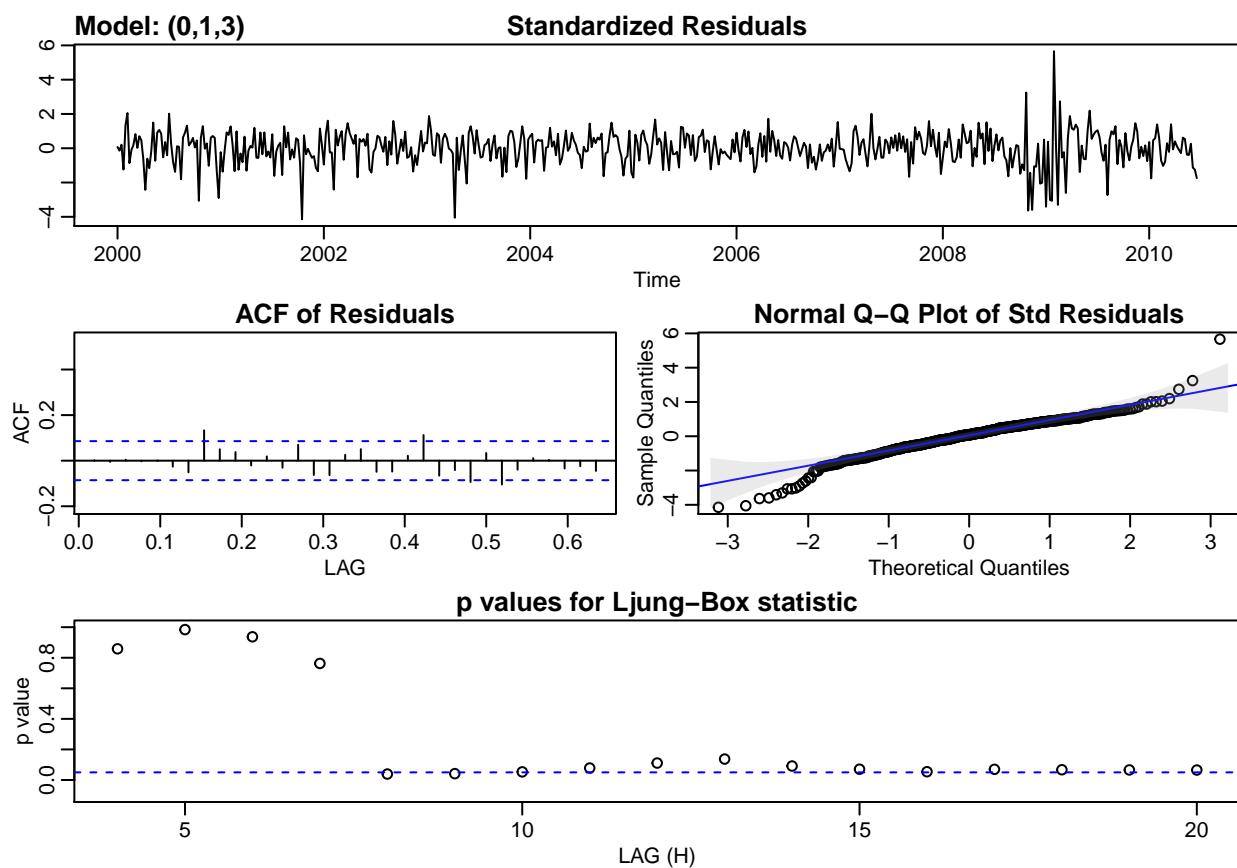
## initial value -3.058495
## iter 2 value -3.086110
## iter 3 value -3.086980
## iter 4 value -3.087501
## iter 5 value -3.087521
## iter 6 value -3.087521
## iter 7 value -3.087522
## iter 8 value -3.087522
## iter 9 value -3.087522

```

```

## 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

```



```

#ADF test
adf.test(modelA$fit$residuals)

```

```

## Warning in adf.test(modelA$fit$residuals): p-value smaller than printed p-
## value

```

```

##
## Augmented Dickey-Fuller Test
##
## data: modelA$fit$residuals
## Dickey-Fuller = -6.5298, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary

```

```

adf.test(modelB$fit$residuals)

## Warning in adf.test(modelB$fit$residuals): p-value smaller than printed p-
## value

##
## Augmented Dickey-Fuller Test
##
## data: modelB$fit$residuals
## Dickey-Fuller = -6.461, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary

```

```
adf.test(modelC$fit$residuals)
```

```

## Warning in adf.test(modelC$fit$residuals): p-value smaller than printed p-
## value

##
## Augmented Dickey-Fuller Test
##
## data: modelC$fit$residuals
## Dickey-Fuller = -6.7187, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary

```

*#Redundancy check*

```
summary(modelA$fit)
```

```

##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##             ma1   constant
##             0.1701    0.0018
## s.e.    0.0499    0.0023
##
## sigma^2 estimated as 0.002157:  log likelihood = 897.88,  aic = -1789.76
##
## Training set error measures:

## Warning in trainingaccuracy(f, test, d, D): test elements must be within
## sample

##               ME RMSE MAE MPE MAPE
## Training set NaN  NaN NaN NaN  NaN

```

```
summary(modelB$fit)
```

```

##  

## Call:  

## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,  

##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,  

##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))  

##  

## Coefficients:  

##      ar1     ma1   constant  

##      -0.5264  0.7146    0.0018  

## s.e.  0.0871  0.0683    0.0022  

##  

## sigma^2 estimated as 0.002102:  log likelihood = 904.89,  aic = -1801.79  

##  

## Training set error measures:  

##  

## Warning in trainingaccuracy(f, test, d, D): test elements must be within  

## sample  

##  

##               ME RMSE MAE MPE MAPE  

## Training set NaN  NaN  NaN  NaN  NaN  

summary(modelC$fit)  

##  

## Call:  

## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,  

##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,  

##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))  

##  

## Coefficients:  

##      ma1     ma2     ma3   constant  

##      0.1688  -0.0900  0.1447    0.0017  

## s.e.  0.0424   0.0425  0.0430    0.0024  

##  

## sigma^2 estimated as 0.00208:  log likelihood = 907.67,  aic = -1805.34  

##  

## Training set error measures:  

##  

## Warning in trainingaccuracy(f, test, d, D): test elements must be within  

## sample  

##  

##               ME RMSE MAE MPE MAPE  

## Training set NaN  NaN  NaN  NaN  NaN  

#BIC  

BIC(modelA$fit)  

## [1] -1776.859  

BIC(modelB$fit)  

## [1] -1784.592

```

```
BIC(modelC$fit)
```

```
## [1] -1783.844
```

```
#AIC
```

```
AIC(modelA$fit)
```

```
## [1] -1789.756
```

```
AIC(modelB$fit)
```

```
## [1] -1801.787
```

```
AIC(modelC$fit)
```

```
## [1] -1805.339
```

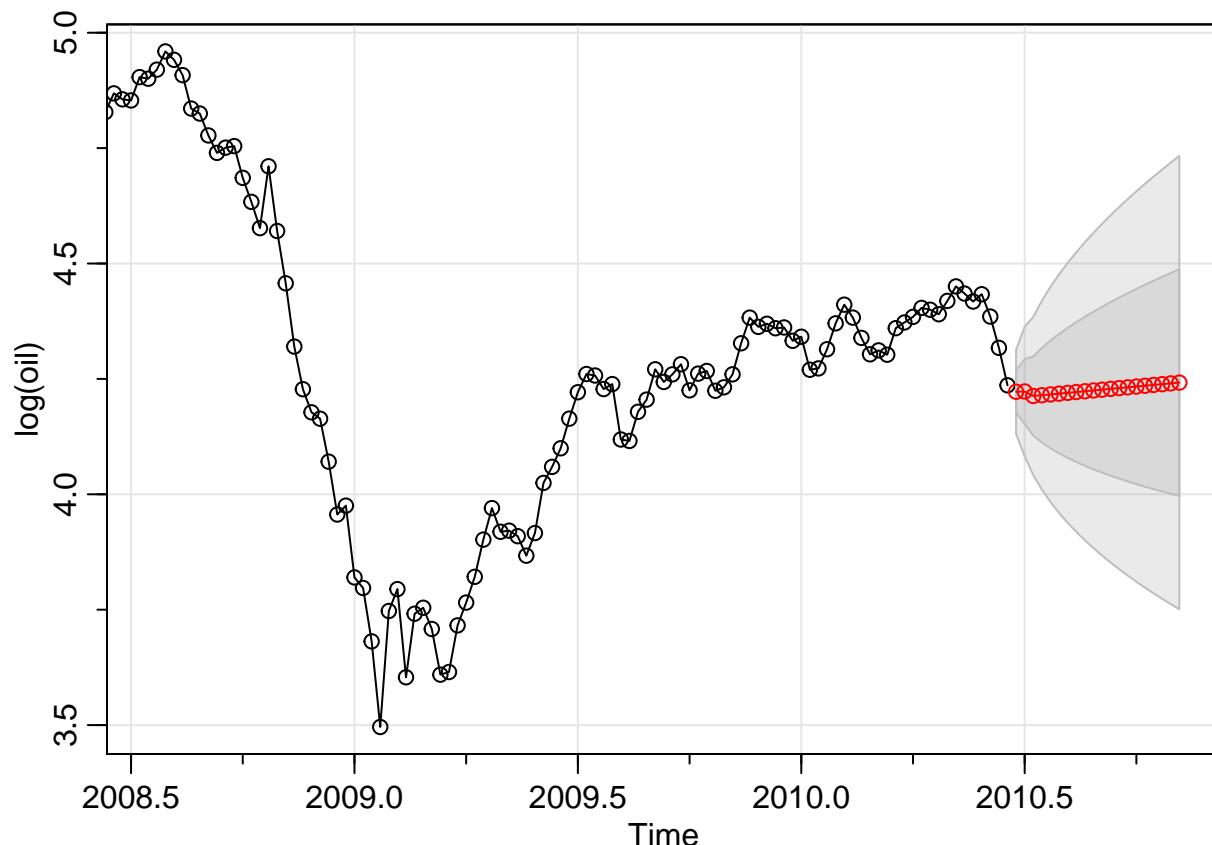
```
#Model C is the best
```

According to AIC and BIC the ModelC (ARIMA 0,1,3) is the best

Model equation is  $\Delta x_t = w_t + 0.1688w_{t-1} - 0.0900w_{t-2} + 0.1447w_{t-3}$

```
#Forecasting
```

```
arima.for(log(oil), 0,1,3, n.ahead = 20)
```



```

## $pred
## Time Series:
## Start = c(2010, 26)
## End = c(2010, 45)
## Frequency = 52
## [1] 4.222141 4.222731 4.212938 4.214647 4.216356 4.218066 4.219775
## [8] 4.221485 4.223194 4.224904 4.226613 4.228323 4.230032 4.231741
## [15] 4.233451 4.235160 4.236870 4.238579 4.240289 4.241998
##
## $se
## Time Series:
## Start = c(2010, 26)
## End = c(2010, 45)
## Frequency = 52
## [1] 0.04561249 0.07016150 0.08569792 0.10226755 0.11650396 0.12918085
## [7] 0.14072033 0.15138273 0.16134203 0.17072132 0.17961149 0.18808192
## [13] 0.19618697 0.20397021 0.21146718 0.21870731 0.22571532 0.23251220
## [19] 0.23911597 0.24554218

```

## State Space Models

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.

$$Z_t = A_{t-1}Z_{t-1} + e_t$$

$$x_t = C_t z_t + \nu_t$$

$$\nu_t \sim N(0, R_t)$$

$$e_t \sim N(0, Q_t)$$

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

$$z_k = z_{k-1} + N(0, Q_t)$$

$$x_k = z_k + N(0, R_t)$$

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

```

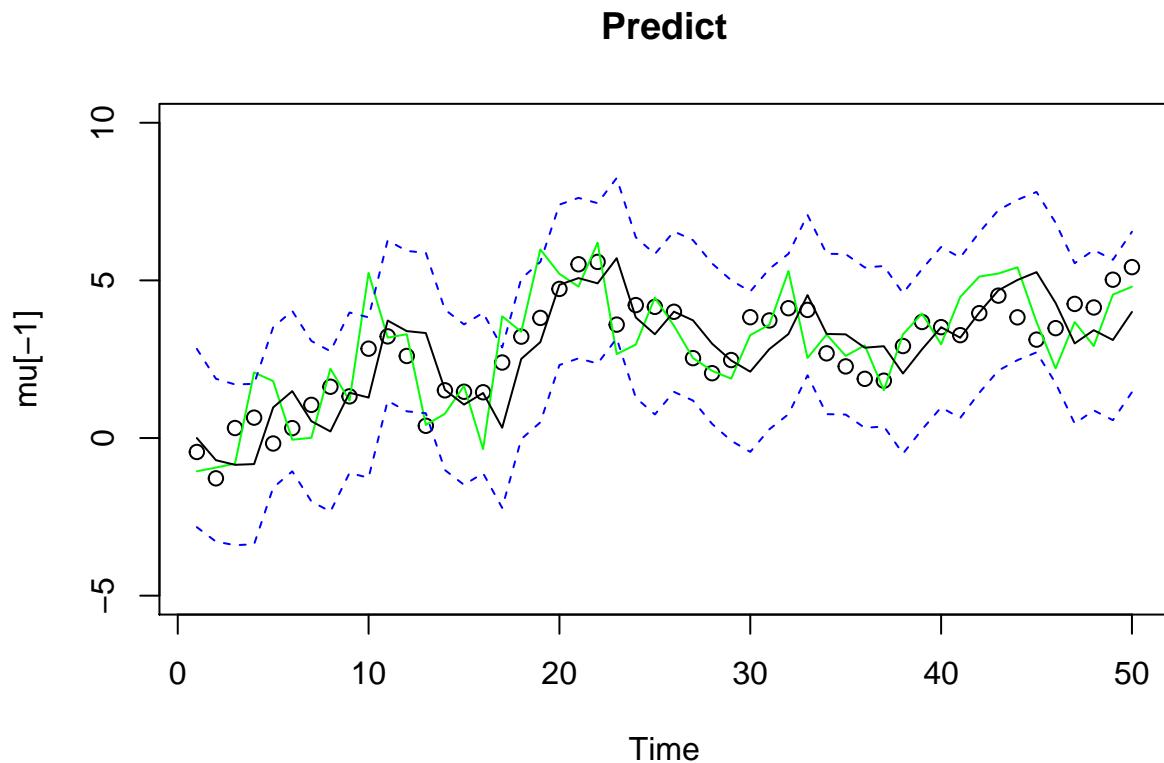
# generate dataset
set.seed(1)
num = 50
w = rnorm(num+1, 0, 1)
v = rnorm(num , 0, 1)
mu = cumsum(w) # state: mu[0], mu[1],..., mu[50]
y = mu[-1] + v # obs: y[1],..., y[50]
# filter and smooth (Ksmooth0 does both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)
# start figurepar(mfrow=c(3,1))
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)

```

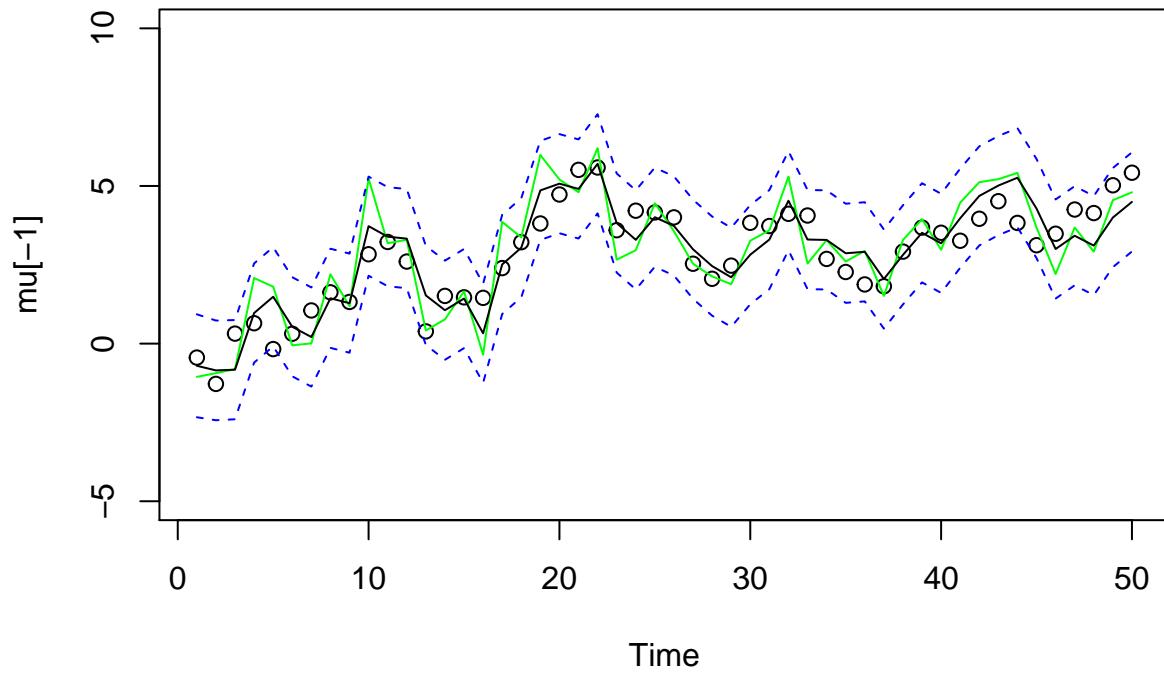


```

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)

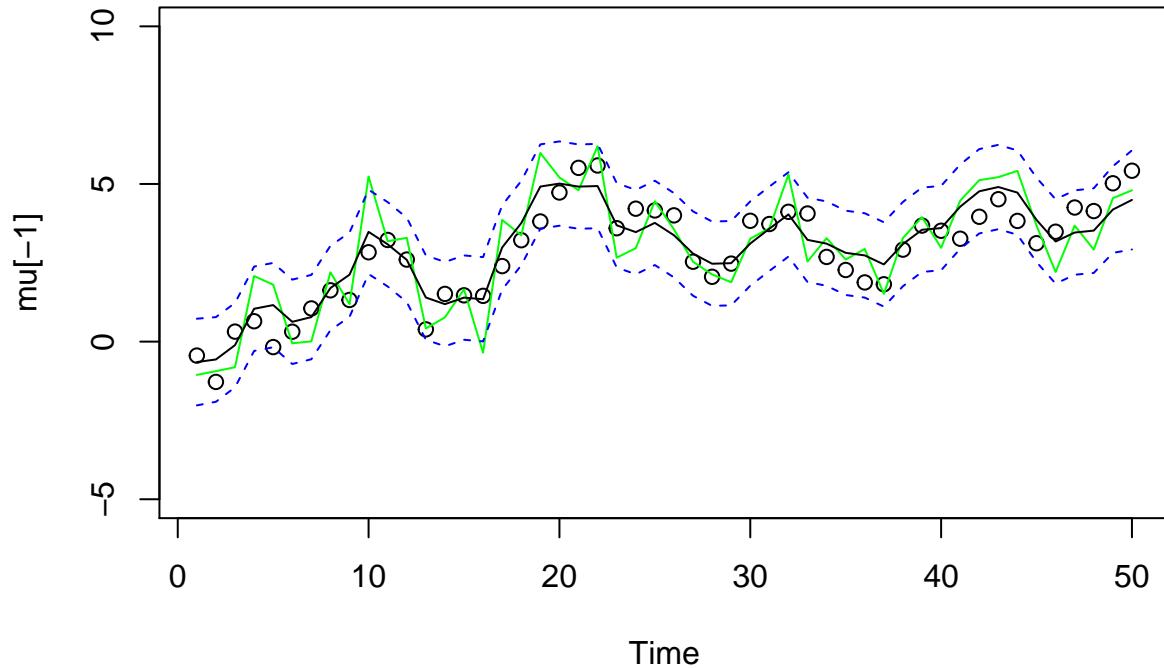
```

## Filter



```
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)
```

## Smooth



```
mu[1]

## [1] -0.6264538

ks$xOn

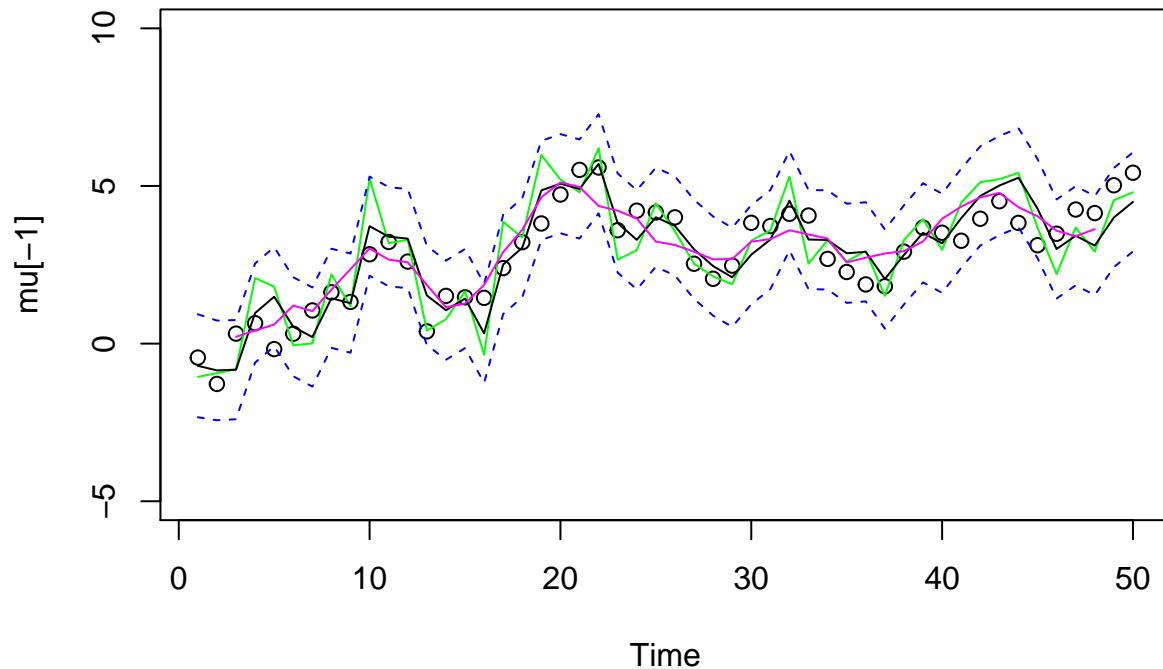
##          [,1]
## [1,] -0.3241541

sqrt(ks$P0n) # initial value info

##          [,1]
## [1,] 0.7861514

# filtering with moving average 5
plot(Time , mu[-1], ylim=c(-5,10), main="Moving average smoothing with order 5")
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)
lines(ma(y, order=5, ), col=6)
```

## Moving average smoothing with order 5

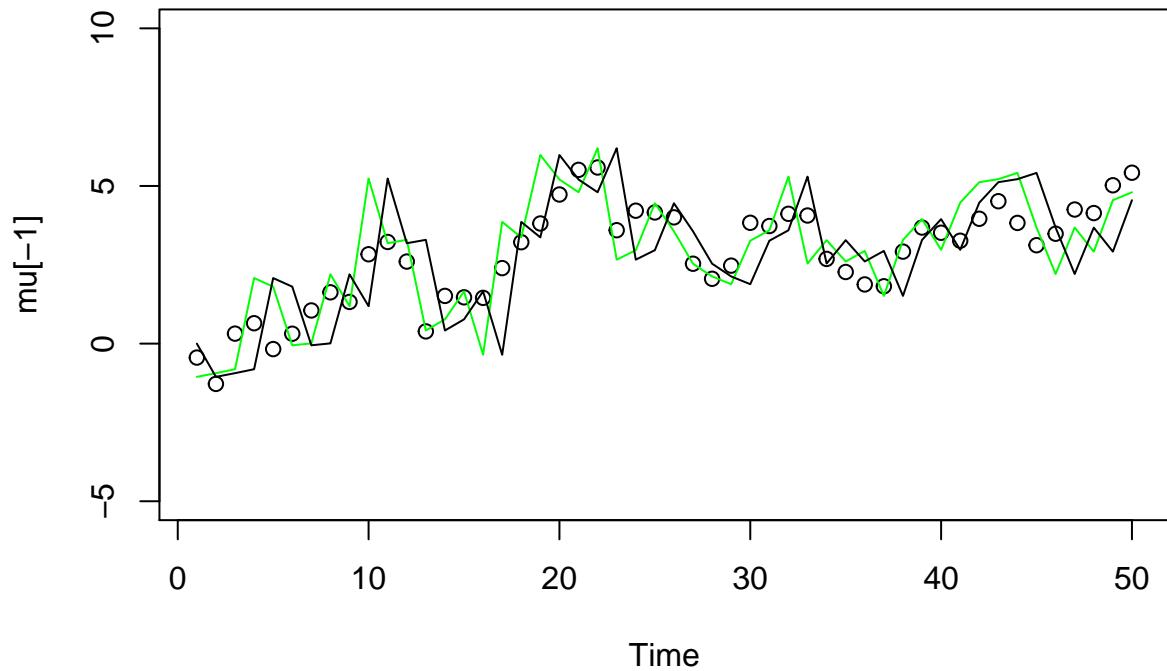


Analysis: We find that the moving average smoothing function with order 5 is the worst fit since its losing all the variability that is captured by our kalman filter.

c) 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?

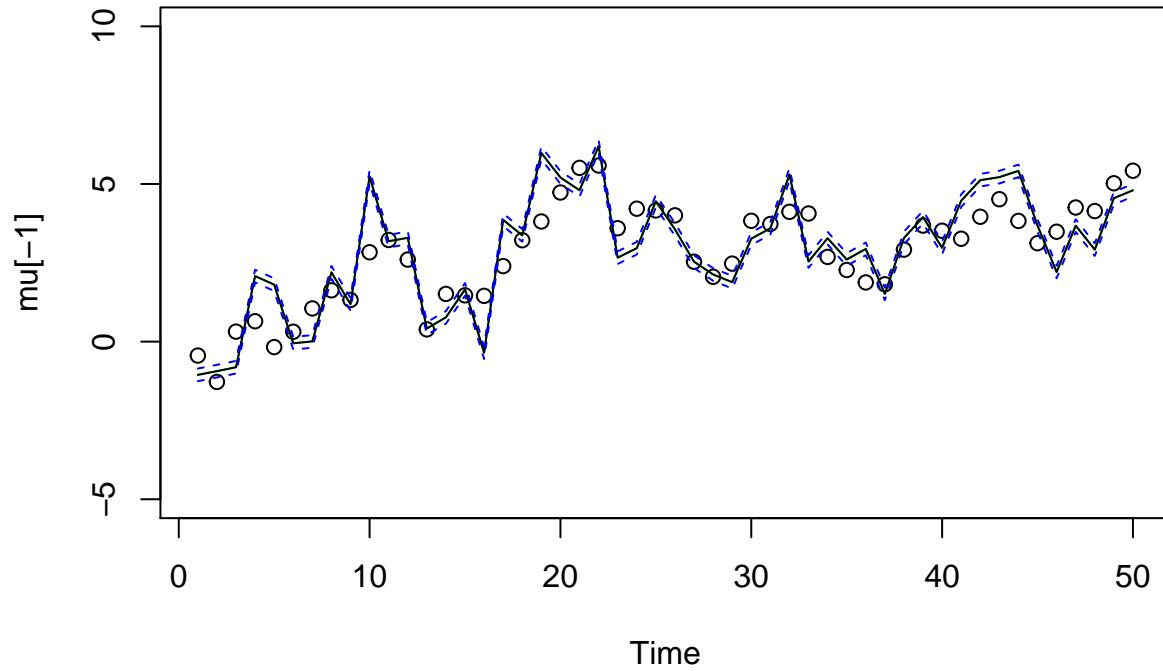
```
# filter and smooth (Ksmooth0 does both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=10, cR=0.1)
# start figurepar(mfrow=c(3,1))
Time = 1:num
plot(Time , mu[-1], main="Predict", ylim=c(-5,10))
lines(Time ,y,col="green")
lines(ks$xp)
lines(ks$xp+2*sqrt(ks$Pp), lty=2, col=4)
lines(ks$xp -2*sqrt(ks$Pp), lty=2, col=4)
```

## Predict



```
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)
```

## Filter

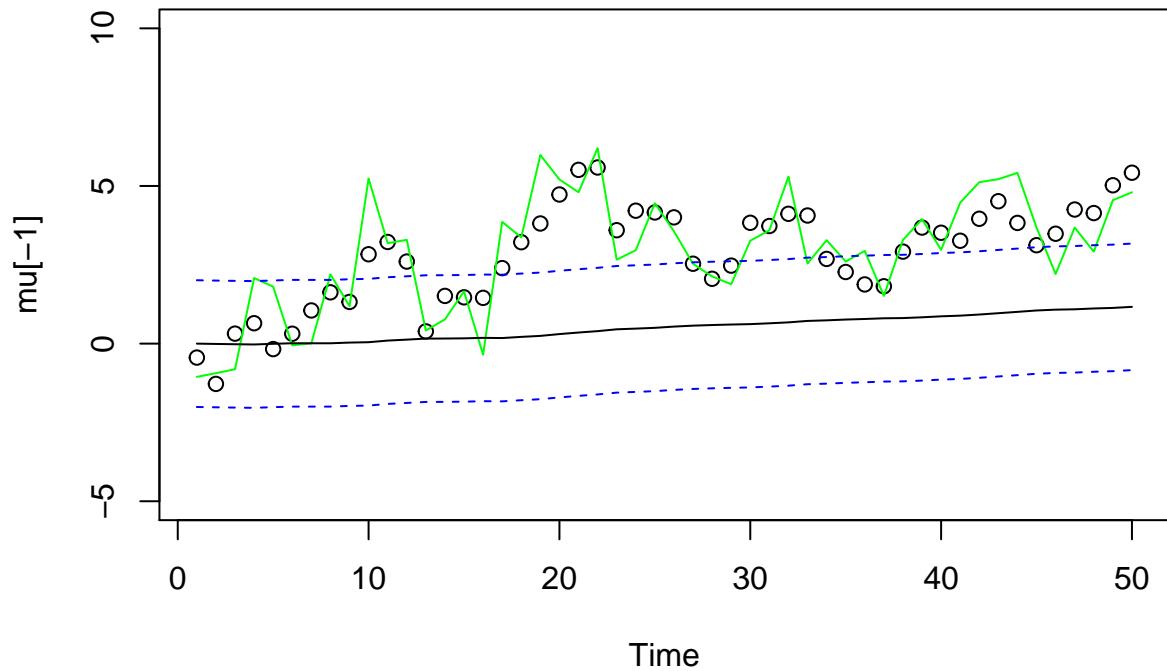


Analysis: We find that the filtering output resembles the true value much more than the previously run values.

- d) 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?

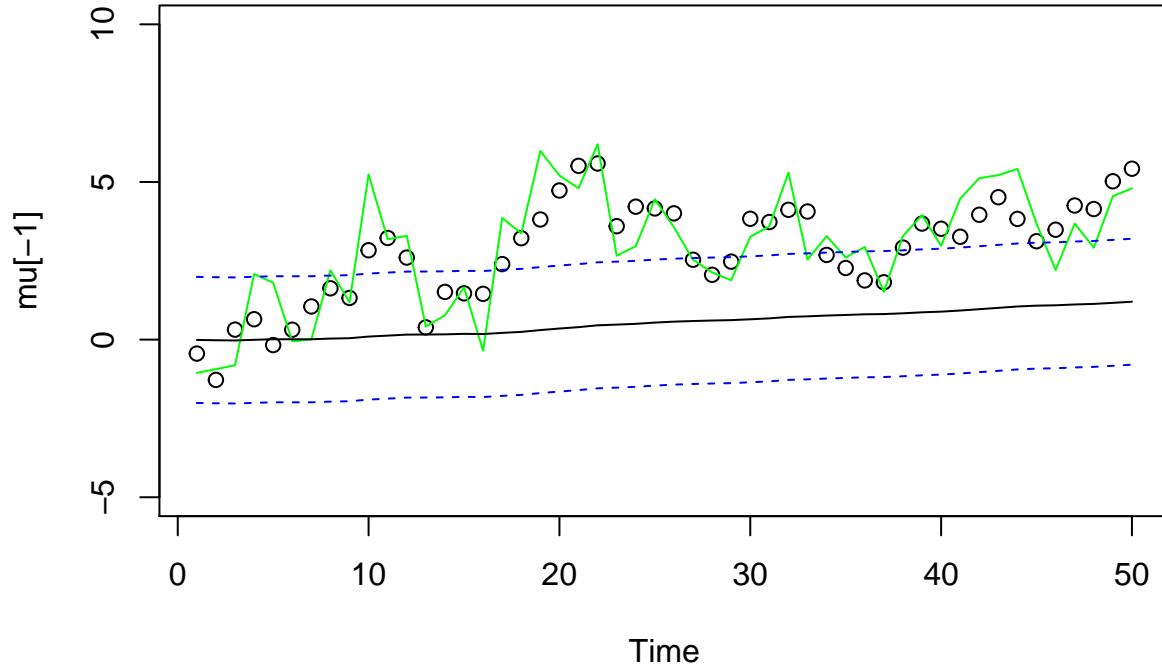
```
# filter and smooth (Ksmooth0 does both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=0.1, cR=10)
# start figure
plot(Time , mu[-1], main="Predict", ylim=c(-5,10))
lines(Time ,y,col="green")
lines(ks$xp)
lines(ks$xp+2*sqrt(ks$Pp), lty=2, col=4)
lines(ks$xp -2*sqrt(ks$Pp), lty=2, col=4)
```

## Predict



```
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)
```

## Filter



Analysis: We find that the filtering output is far off the true value and hardly moving in terms of the predicted values, this is because kalman gain assumes that the uncertainty in measurement is high and it largely sticks with the mean of the series.

### Kalman Filter Code

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

```
#Times: Number of time steps
#A: How the mean of z_t will be affected by z_(t-1) Used in transition model
#C: Scale the mean (z_t) in the emission model
#Q: Covariate matrix in the emission model
#R: Covariate matrix in the transition model
#y: All observations
#mu0: first mean
#sigma0: first varience
my_kalman <- function(Times, y, A, mu0, Sigma0, C, Q, R) {
  mu <- rep(1,Times)
  sigma <- rep(1,Times)
  my_unweighted <- rep(1,Times)
  sigma_unweighted <- rep(1,Times)
  kalman_gain <- rep(1,Times)

  # Our best guess is that my_1 is mu0 else it could be the first observation
  mu[1] <- mu0
```

```

# We don't know what sigma_1 is, so we choose 1 else it would be provided
sigma[1] <- Sigma0

for (t in 2:Times) {
  # Calculate the unweighted prediction of the mean
  my_unweighted[t] <- A[t] %*% mu[t - 1]

  # Calculate the unweighted prediction of the covariate matrix
  sigma_unweighted[t] <- A[t] %*% sigma[t - 1] %*% t(A[t]) + Q[t]

  # Kalman gain Used to weight between our unweighted prediction and the obs
  kalman_gain[t] <- sigma_unweighted[t] %*% t(C[t]) %*%
    solve(C[t] %*% sigma_unweighted[t] %*% t(C[t] + R[t]))

  # Calculate the weighted mean, thus our prediction of the hidden state
  mu[t] <- my_unweighted[t] + kalman_gain[t] %*% (y[t] - C[t] %*% my_unweighted[t])

  # Calculate the weighted covariance matrix, thus our prediction of the predition error
  sigma[t] <- (1 - kalman_gain[t] %*% C[t]) %*% sigma_unweighted[t]
}

return (list(mu = mu, sigma = sigma))
}

```

### Alternative version

```

### MODIFIED, DOES WORK

kalman_filter = function(num, data, A, C, Q, R, m0, P0) {

  if (length(A) == 1)
    A = as.list(rep(A, num+1))

  if (length(C) == 1)
    C = as.list(rep(C, num+1))

  if (length(Q) == 1)
    Q = as.list(rep(Q, num+1))

  if (length(R) == 1)
    R = as.list(rep(R, num+1))

  # Init
  m = list()
  P = list()
  K = list()

  # Setup
  # Note that the formula with the inverse has been changed, as otherwise the
  # dimensions have to be determined first.
  m[[1]] = m0

```

```

P[[1]] = P0

for (t in 2:(num)) {

  # Prediction Step
  m[[t-1]] = A[[t-1]] %*% m[[t-1]]
  P[[t-1]] = A[[t-1]] %*% P[[t-1]] %*% t(A[[t-1]]) + Q[[t]]

  # Observation Update Step
  K[[t]] = P[[t-1]] %*% t(C[[t]]) %*% solve(C[[t]] %*% P[[t-1]]
                                              %*% t(C[[t]]) + R[[t]])
  m[[t]] = m[[t-1]] + K[[t]] %*% (data[[t]] - C[[t]] %*% m[[t-1]])
  P[[t]] = P[[t-1]] - K[[t]] %*% C[[t]] %*% P[[t-1]]
  #P[[t]] = (diag(ncol(K[[t]])) - K[[t]] %*% C[[t]]) %*% P[[t-1]]

}

return(list(m = unlist(m), P=unlist(P), K=unlist(K)))
}

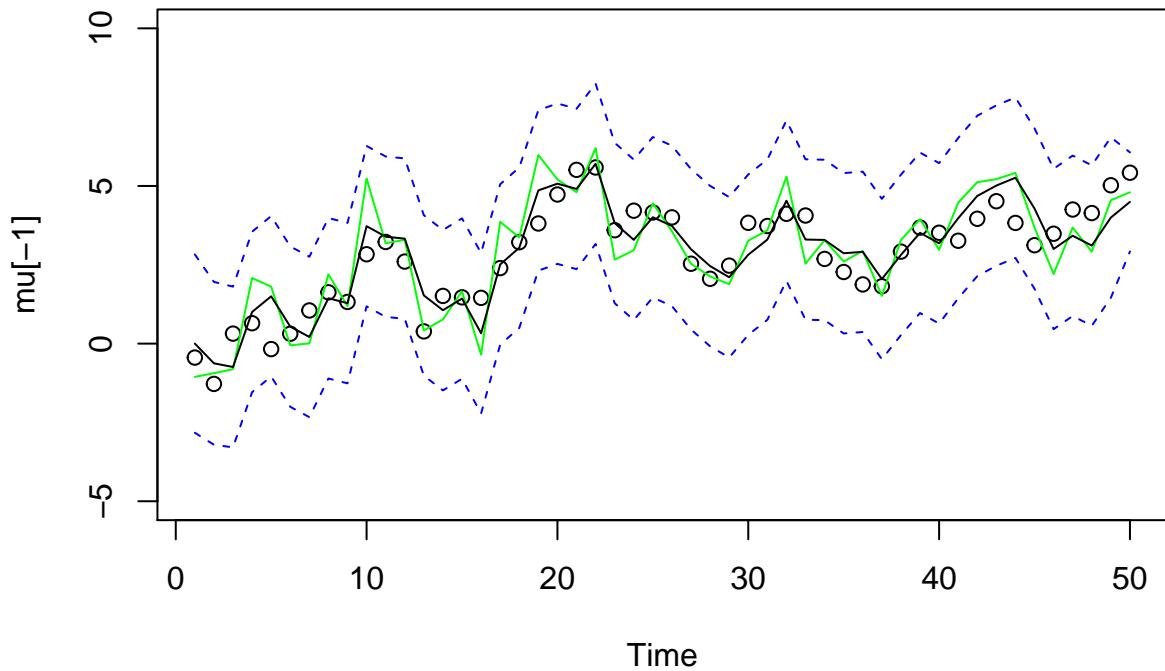
# generate dataset
set.seed(1)
num = 50
w = rnorm(num+1, 0, 1)
v = rnorm(num, 0, 1)
mu = cumsum(w) # state: mu[0], mu[1], ..., mu[50]
y = mu[-1] + v # obs: y[1], ..., y[50]

# filter and smooth (Ksmooth0 does both)
ks = kalman_filter(num, y, A=1, m0=0, P0=1, C=1, Q=1, R=1)
# start figurepar(mfrow=c(3,1))
Time = 1:num

plot(Time, mu[-1], main="Predict (custom Kalman Filter)", ylim=c(-5, 10))
lines(Time, y, col="green")
lines(ks$m)
lines(ks$m+2*sqrt(ks$P), lty=2, col=4)
lines(ks$m-2*sqrt(ks$P), lty=2, col=4)

```

## Predict (custom Kalman Filter)

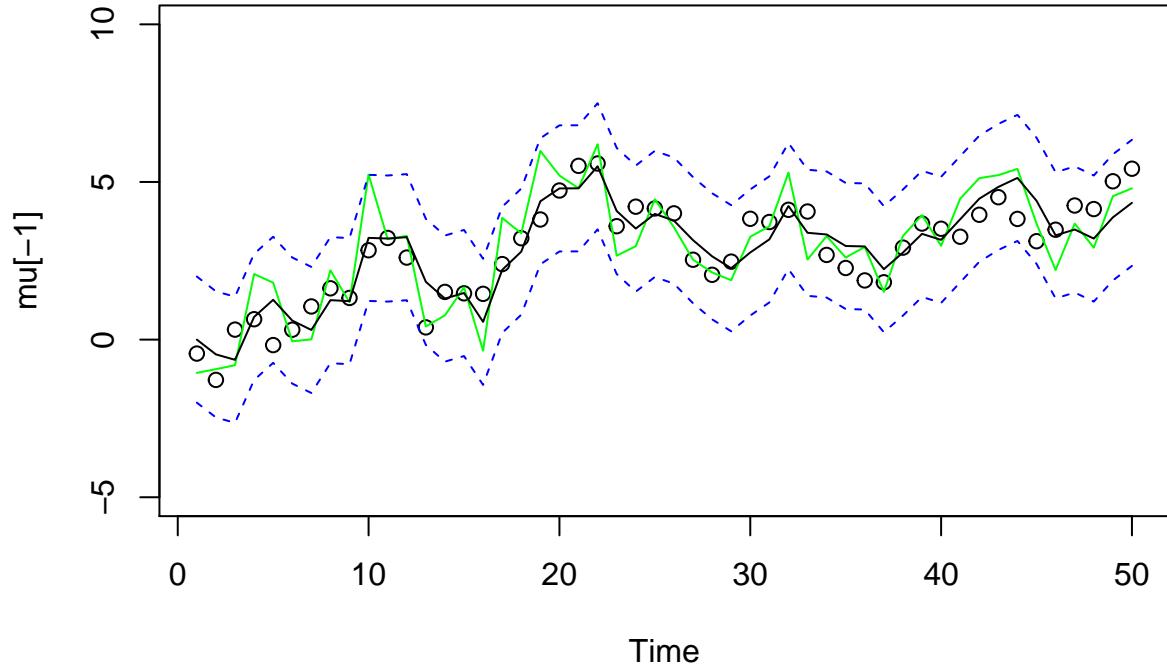


```
# generate dataset
set.seed(1)
num = 50
w = rnorm(num+1,0,1)
v = rnorm(num ,0,1)
mu = cumsum(w) # state: mu[0], mu[1],..., mu[50]
y = mu[-1] + v # obs: y[1],..., y[50]

my_ks = my_kalman(Times=num , y=y, A=rep(1,num), mu0=0, Sigma0=1,
                    C=rep(1,num), Q=rep(1,num), R=rep(1,num))

plot(Time , mu[-1], main="Predict", ylim=c(-5,10))
lines(Time ,y,col="green")
lines(my_ks$mu)
lines(my_ks$mu+2*sqrt(my_ks$sigma), lty=2, col=4)
lines(my_ks$mu -2*sqrt(my_ks$sigma), lty=2, col=4)
```

## Predict



f) How do you interpret the Kalman gain?

Analysis: Kalman gain is given by  $K = \frac{P_k H_k^T}{P_k H_k^T + R_k}$  where you will realize that the relative magnitudes of matrices  $R_k$  and  $P_k$  control a relation between the filter's use of predicted state estimate  $z_t$  and measurement  $x_t$ .

When  $R_k$  tends to zero then  $x_t = z_t + K(y_t - H_k)$  suggests that when the magnitude of  $R$  is small, meaning that the measurements are accurate, the state estimate depends mostly on the measurements.

When the state is known accurately, then numerator is small compared to  $R$ , and the filter mostly ignores the measurements relying instead on the prediction derived from the previous state.

## Lecture Slides

### Lecture 1

# Time Series Analysis

## Lecture 1: Introduction

**Tohid Ardestiri**

Linköping University  
Division of Statistics and Machine Learning

September 2, 2019

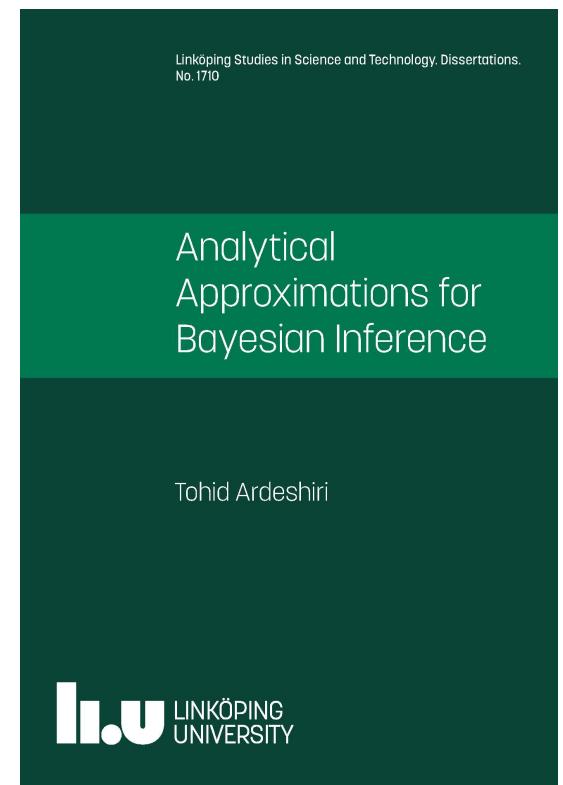


# Course teacher

Tohid Ardestiri  
PhD in 2015 in Bayesian inference



Senior Data Scientist at  
Qamcom Research & Technology AB



# Bayesian Inference

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

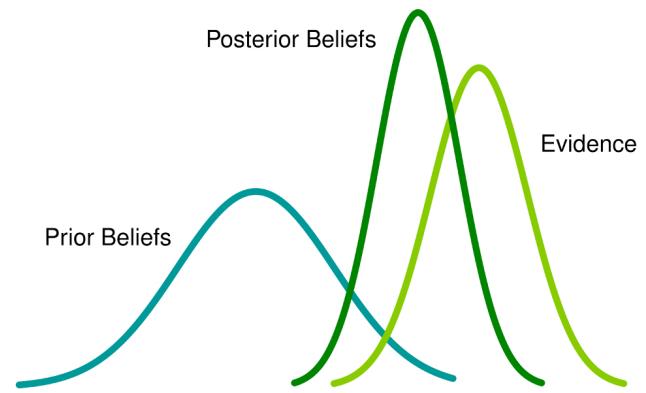
## Example: Parameter learning

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

## Probability Calculus

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

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



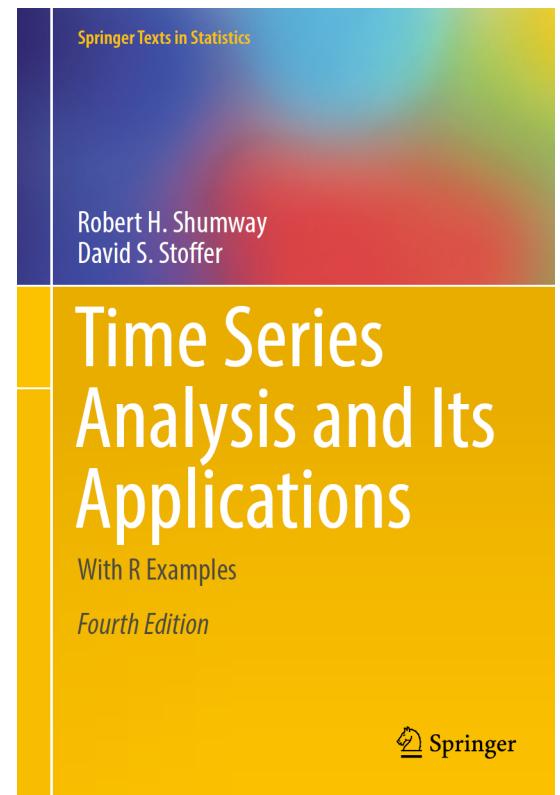
# Course literature and software

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 .

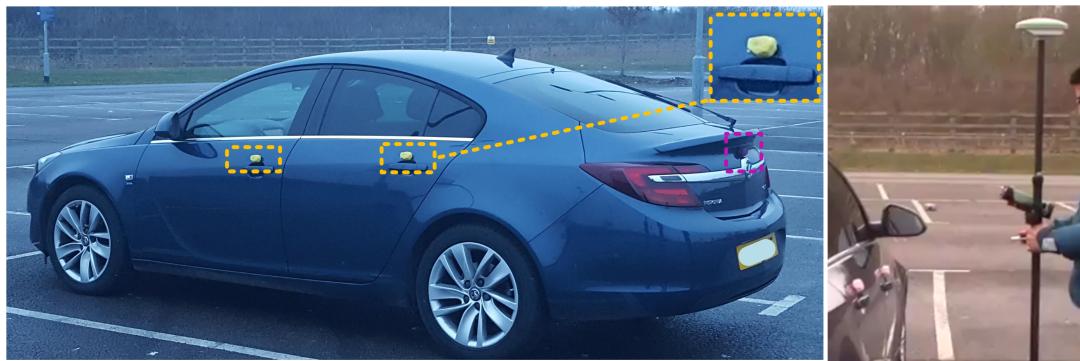
Sequential data: A sentence or a word

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

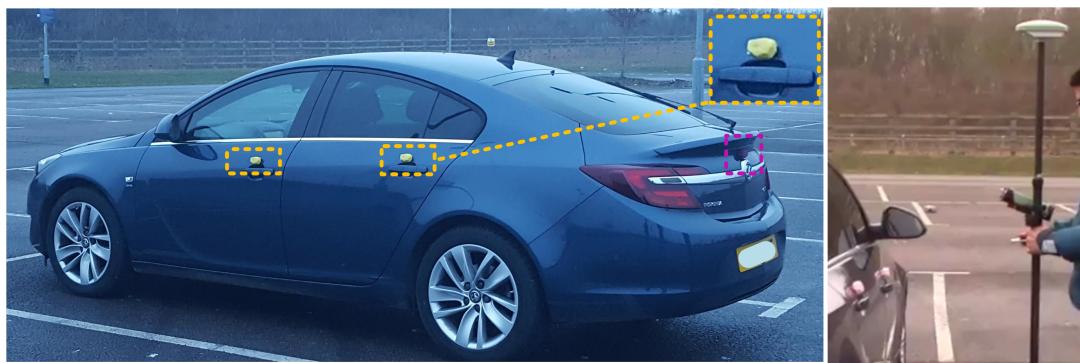
Received signal strength indicator (RSSI) is a common observation (data).



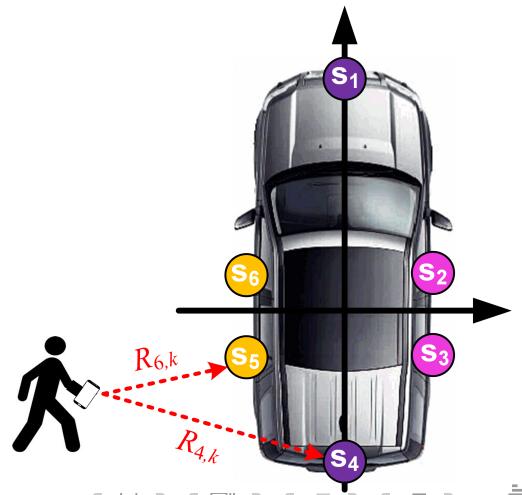
Where is the driver?

## A look at real data

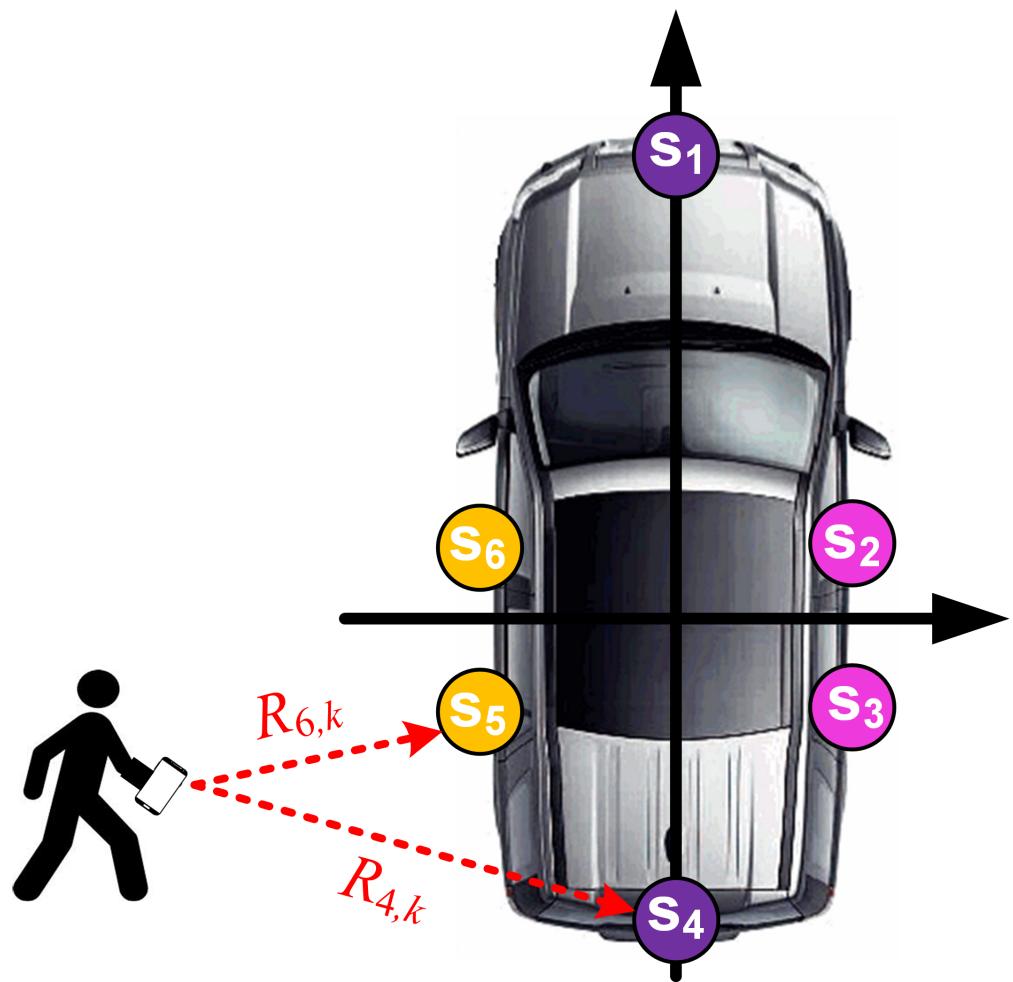
Received signal strength indicator (RSSI) is a common observation (data).



Where is the driver?



# Where is the driver?



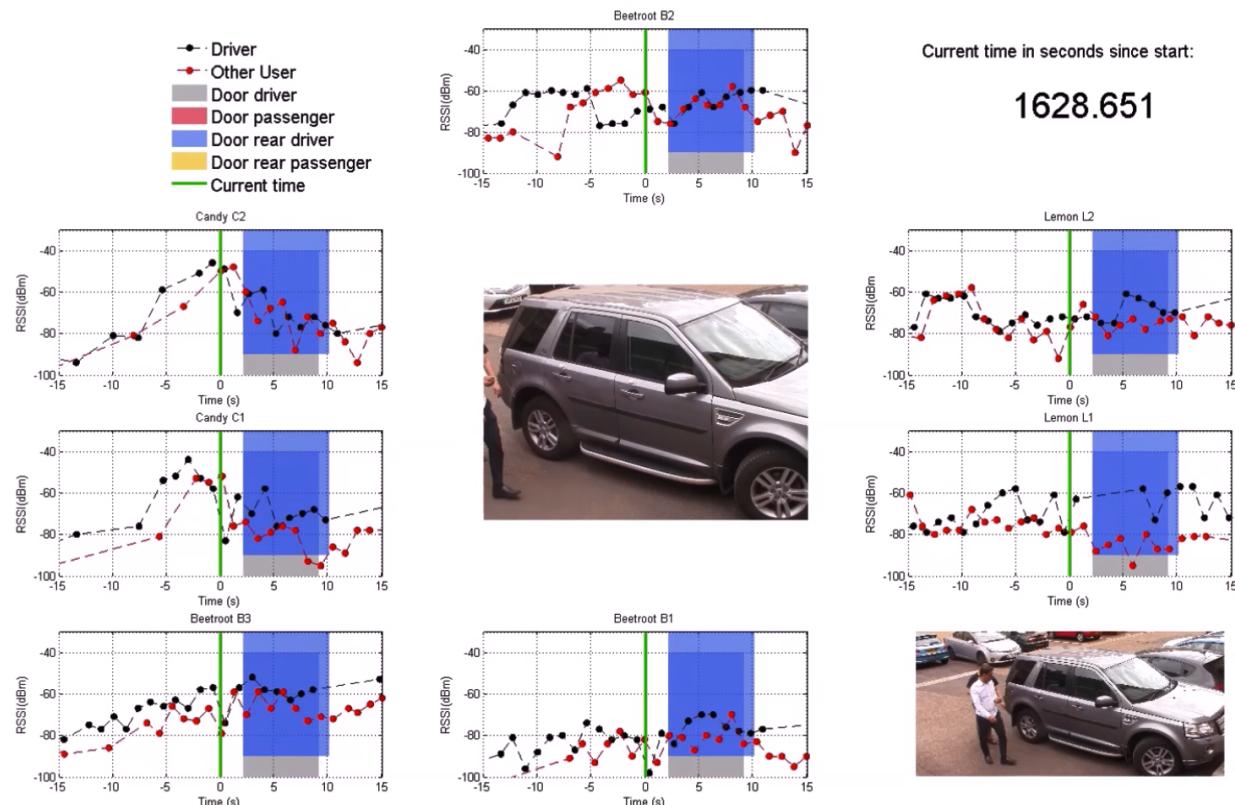
# Where is the driver?

Video of data collection



# Where is the driver?

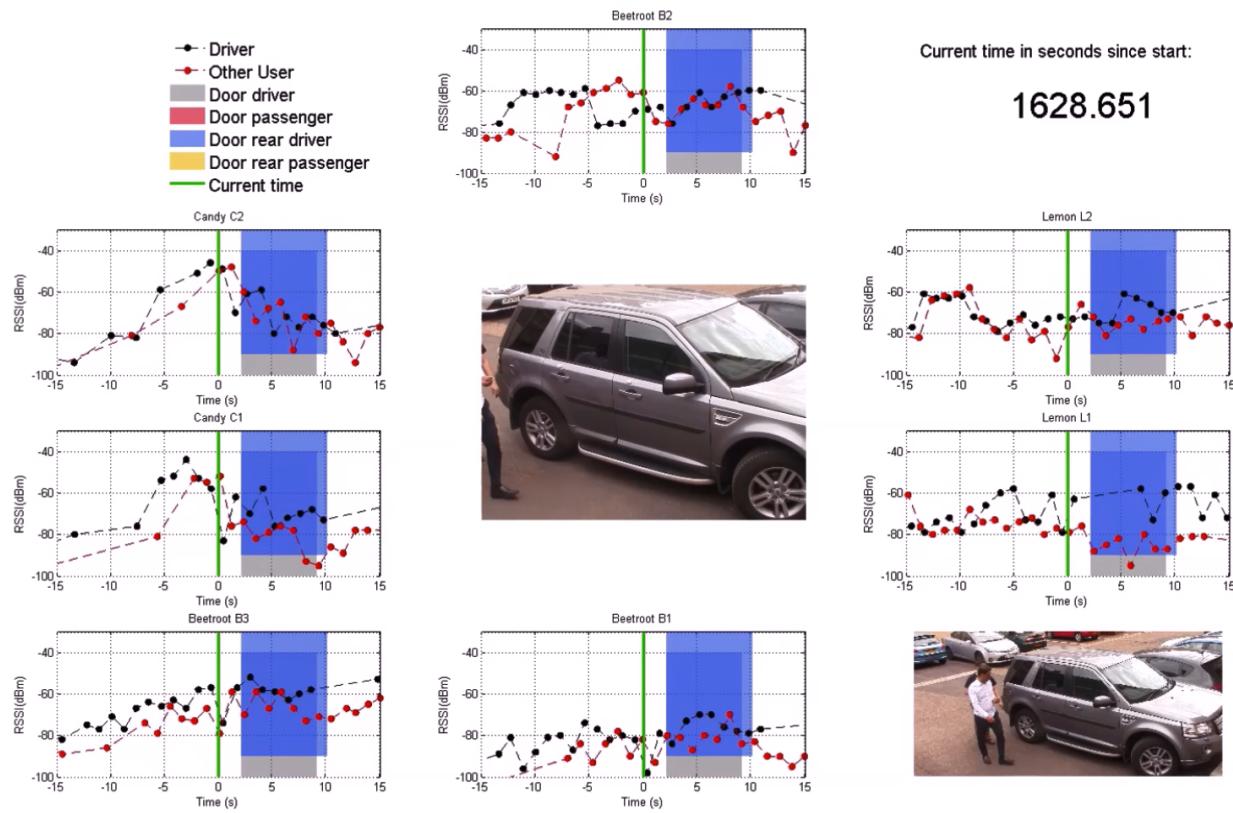
## Animation of the of signals



# Time Series Analysis

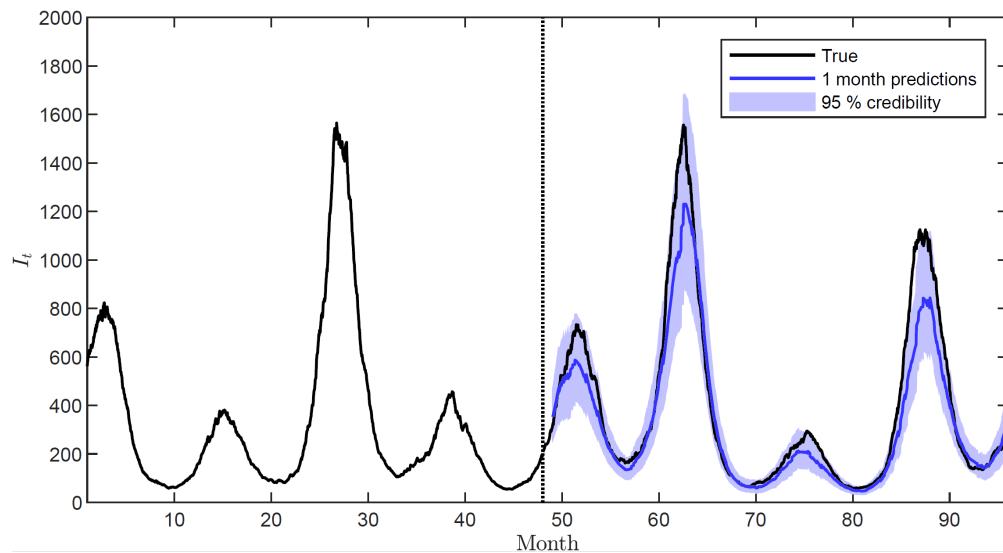
## What is a Time Series?

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



# Time Series Analysis

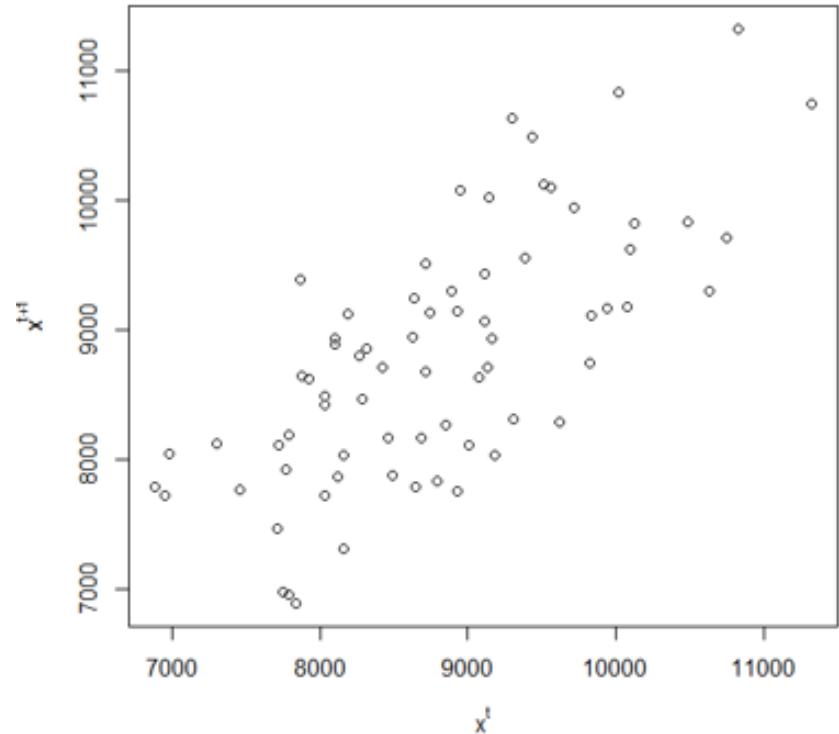
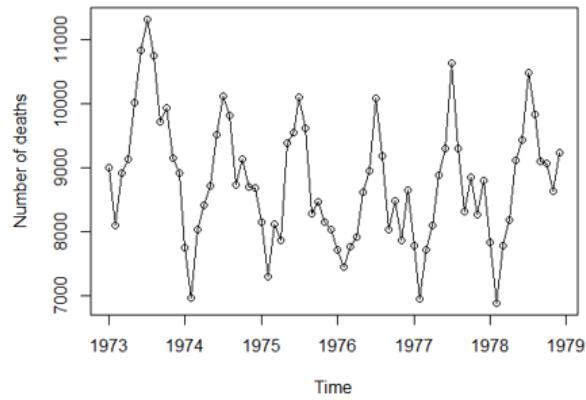
- Understand the properties of the underlying process
- Be able to predict (forecast) possible future values
- Reason about the **uncertainties** in the predictions  
**requires statistical methods!**



# Time Series Analysis

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

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

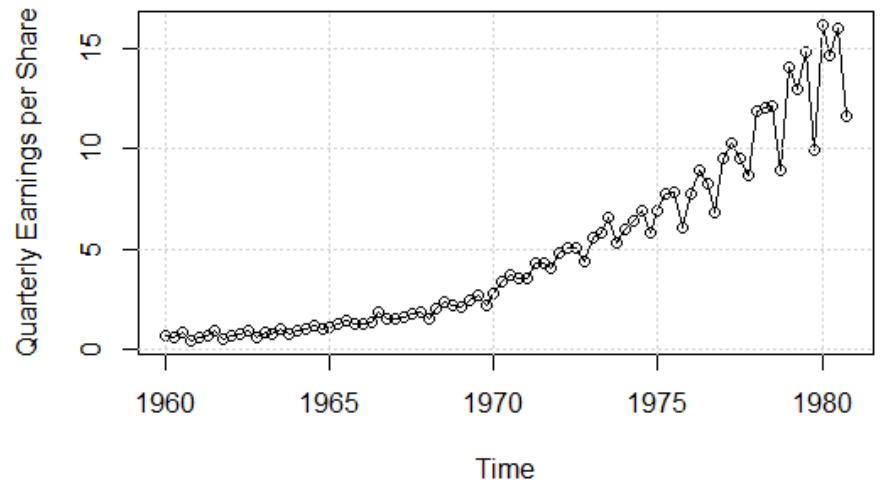


Ex) See connection  
between  $x_t$  and  $x_{t+1}$

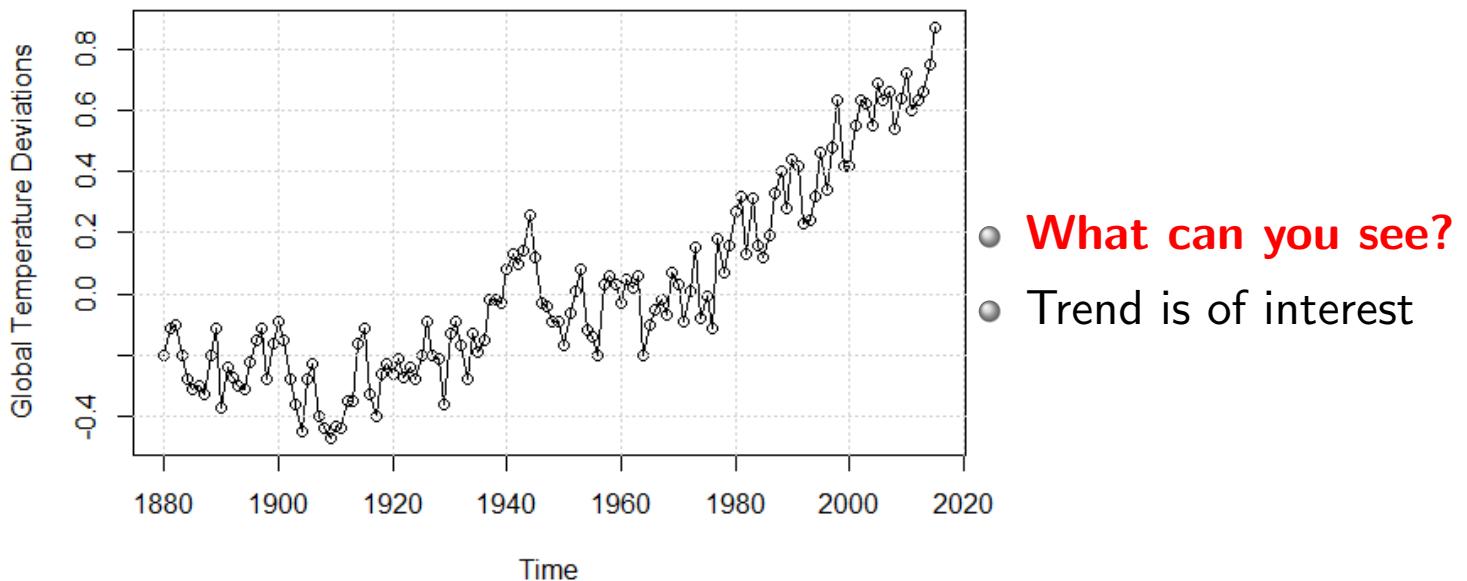
## Ex 1: Johnson & Johnson quarterly earnings

- **What can you see?**

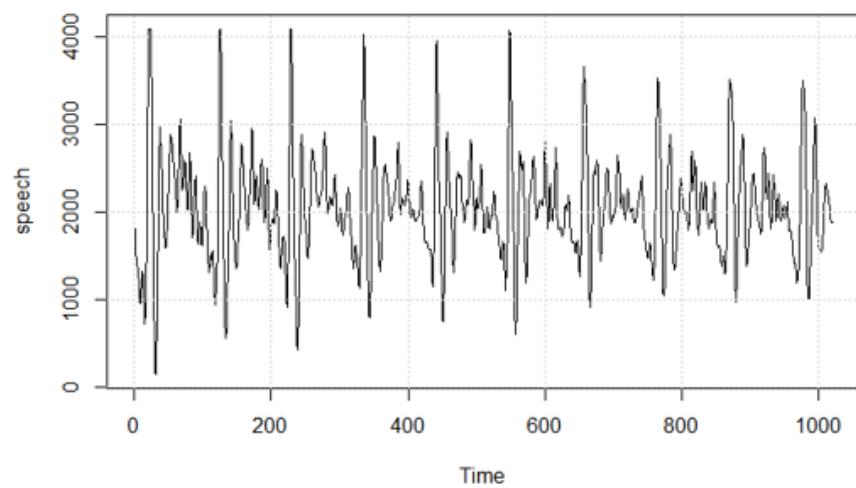
- ▶ Trend?
  - ★ Constant
  - ★ Linear
  - ★ Other
- ▶ Variation?
- ▶ Seasonality?
- ▶ Outliers?



## Ex 2: Global warming



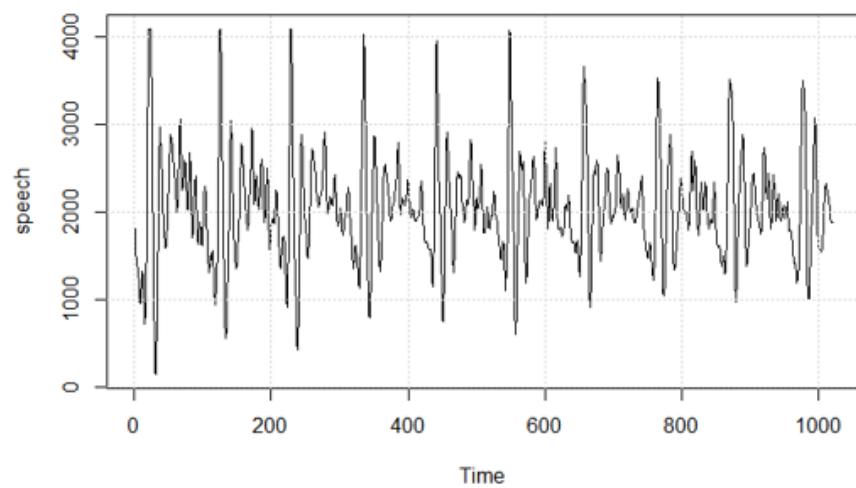
## Ex 3: Speech data



• **What can you see?**

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

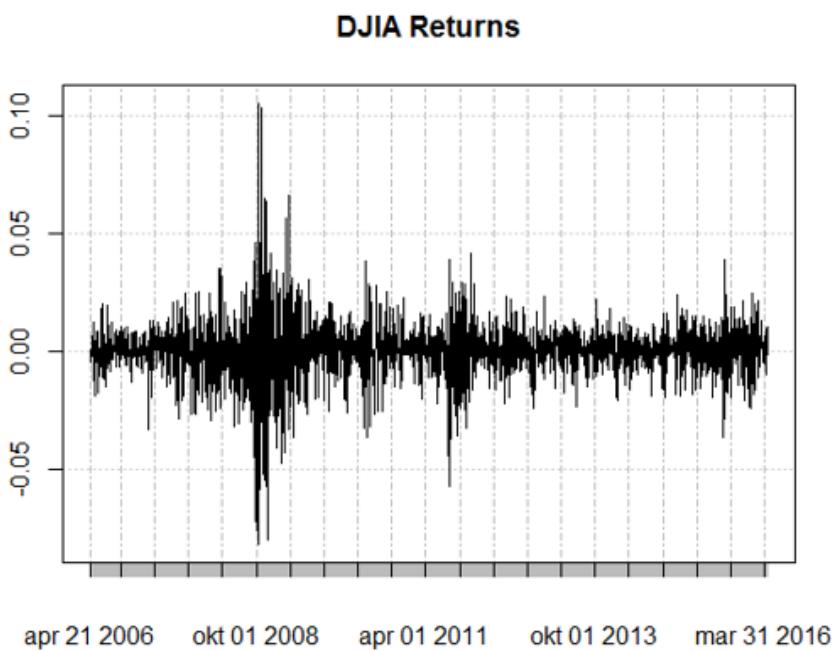
## Ex 3: Speech data



• **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

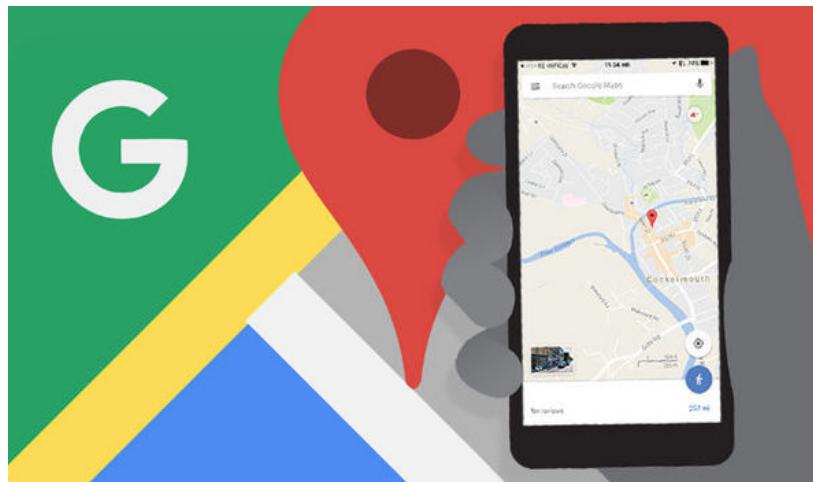
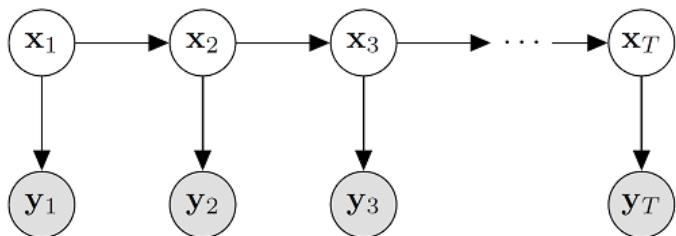


- What can you see here?

Pattern of periodicity is of interest → Stochastic volatility

## Ex 5: Dynamical systems

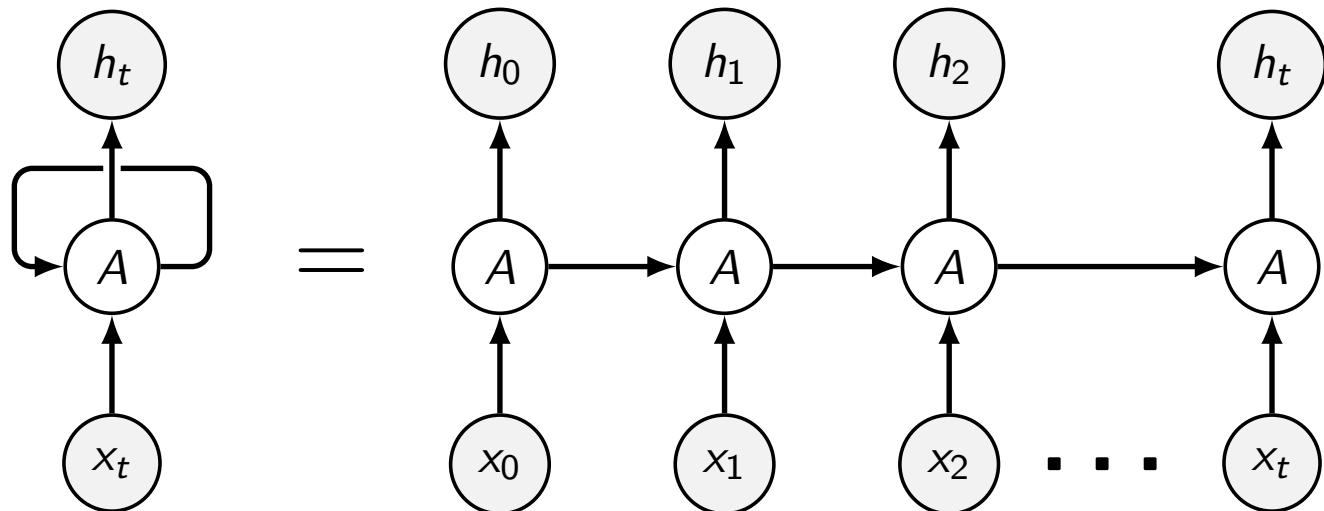
Linear and Gaussian state-space models for tracking objects



## Ex 6: Recurrent neural networks

Natural Language Processing

This        is        a        sequential        data        type .

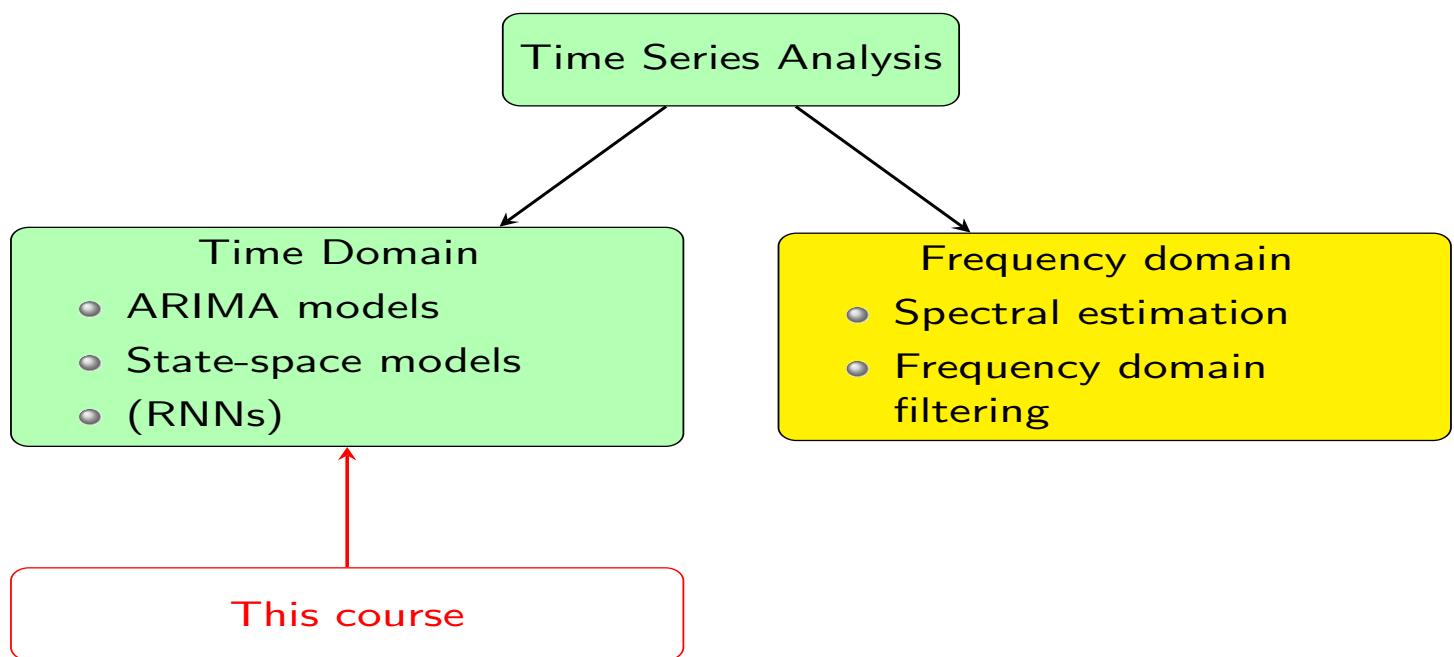


# Time Series Analysis

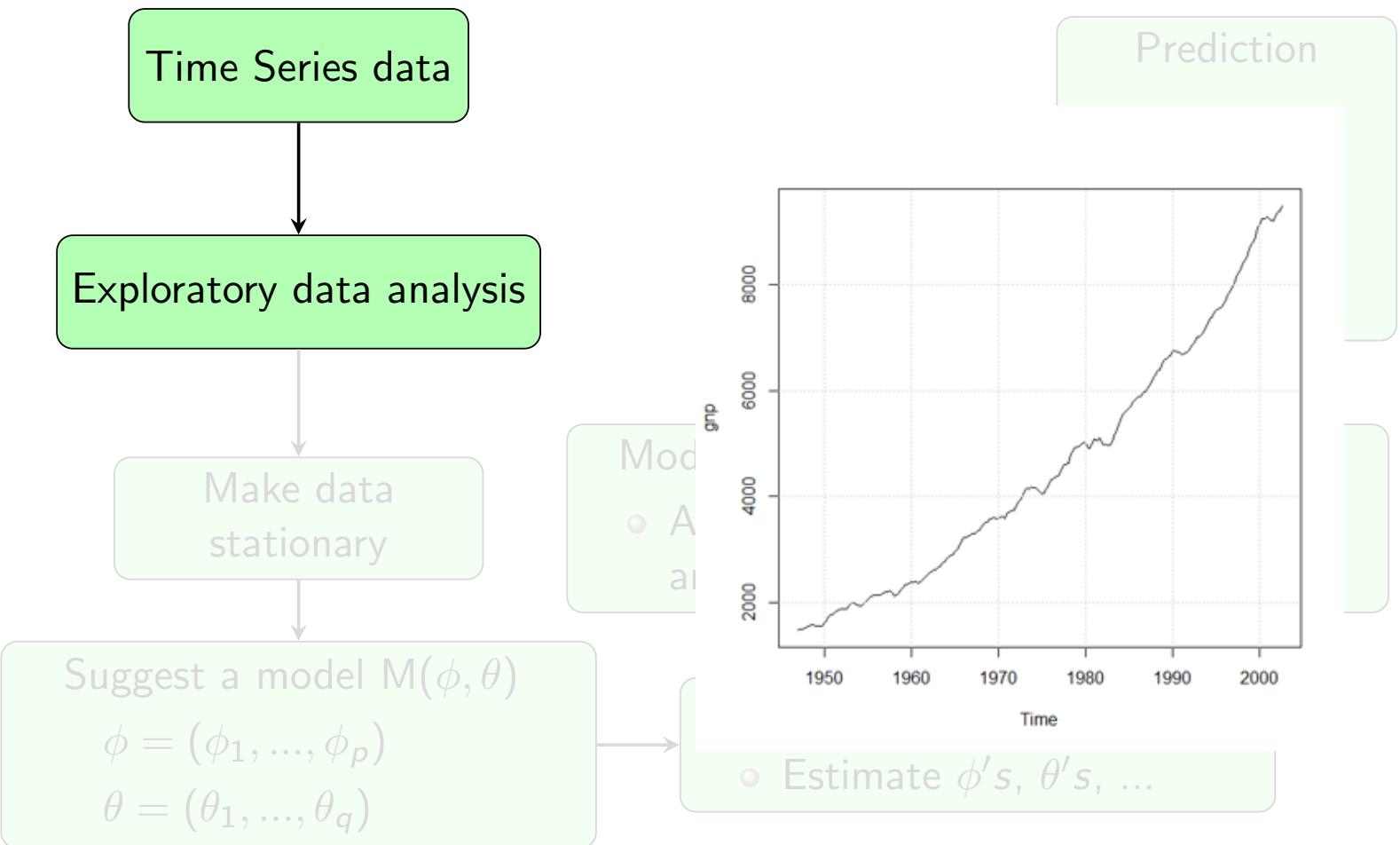
## Application areas

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

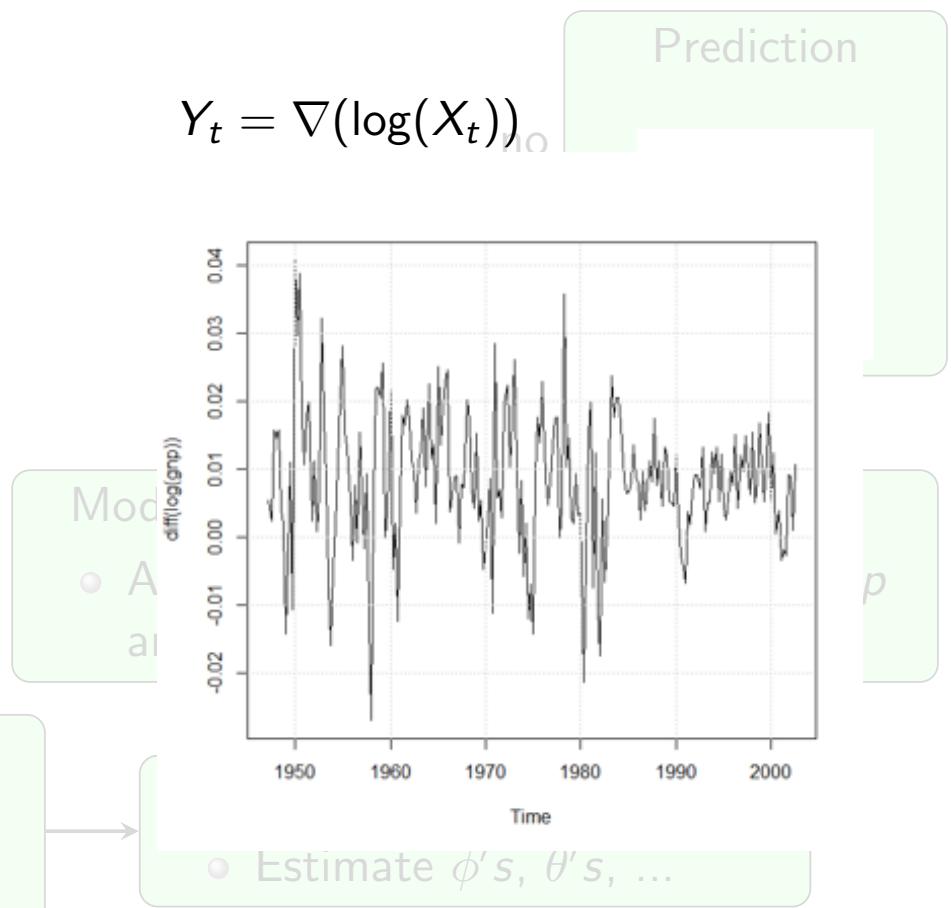
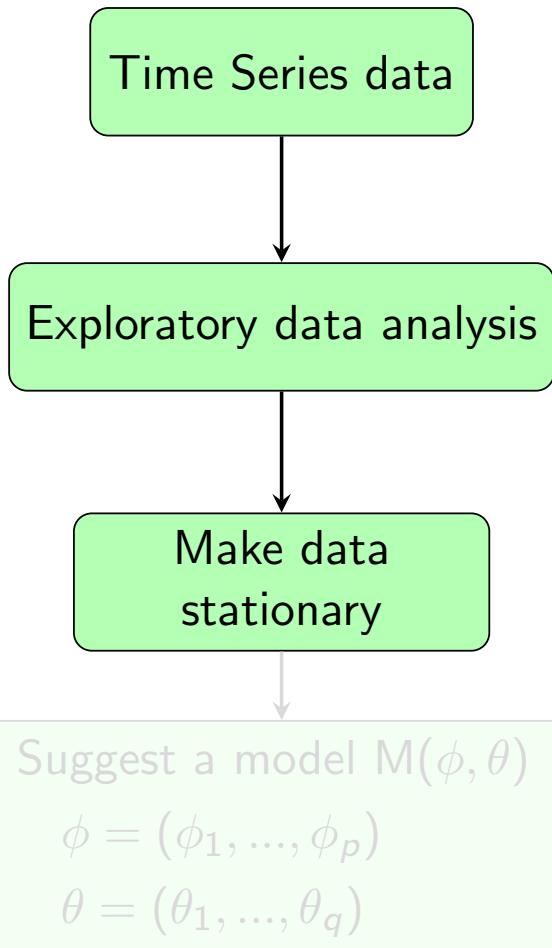
# The Big Picture



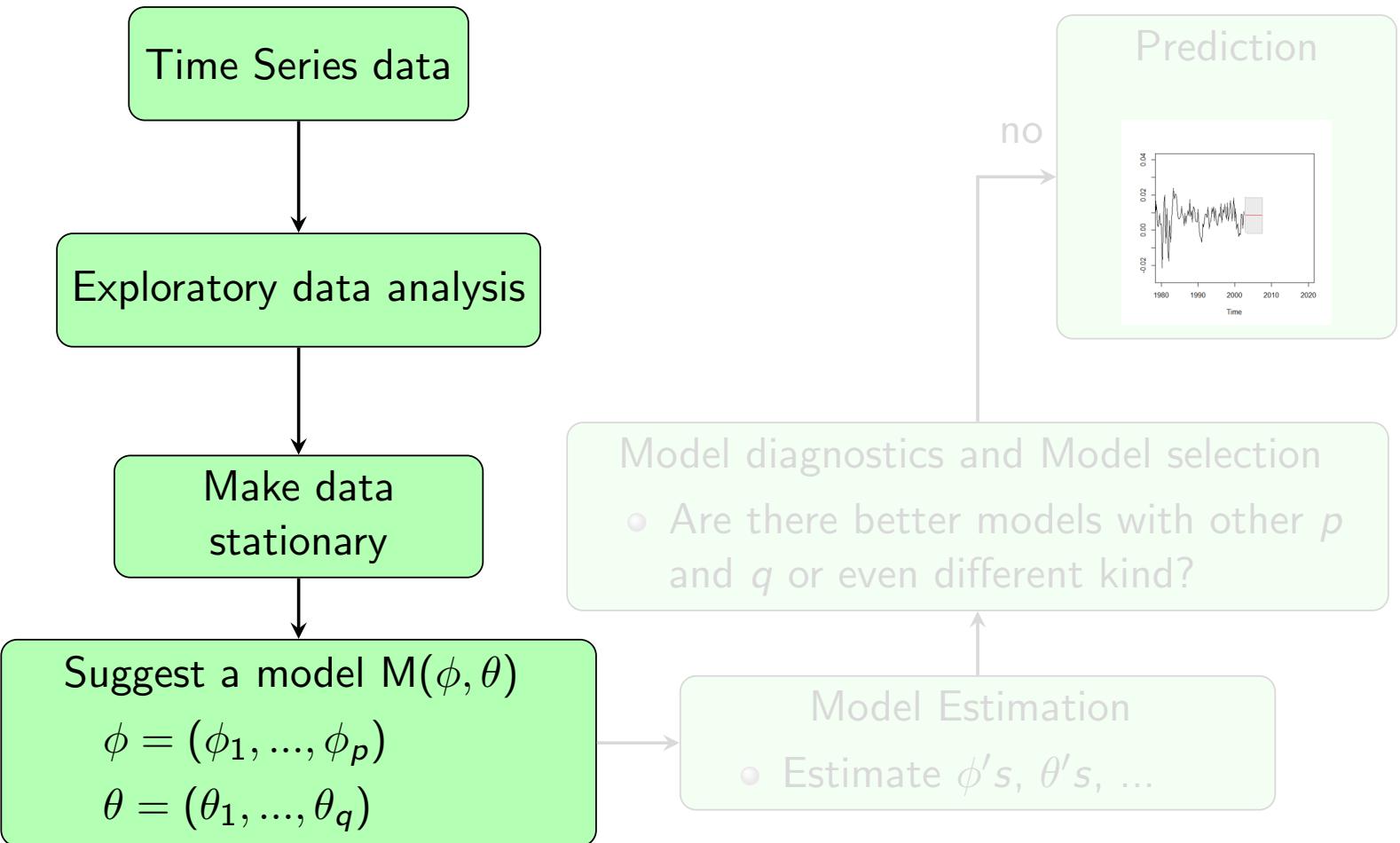
# Time domain: The Big Picture



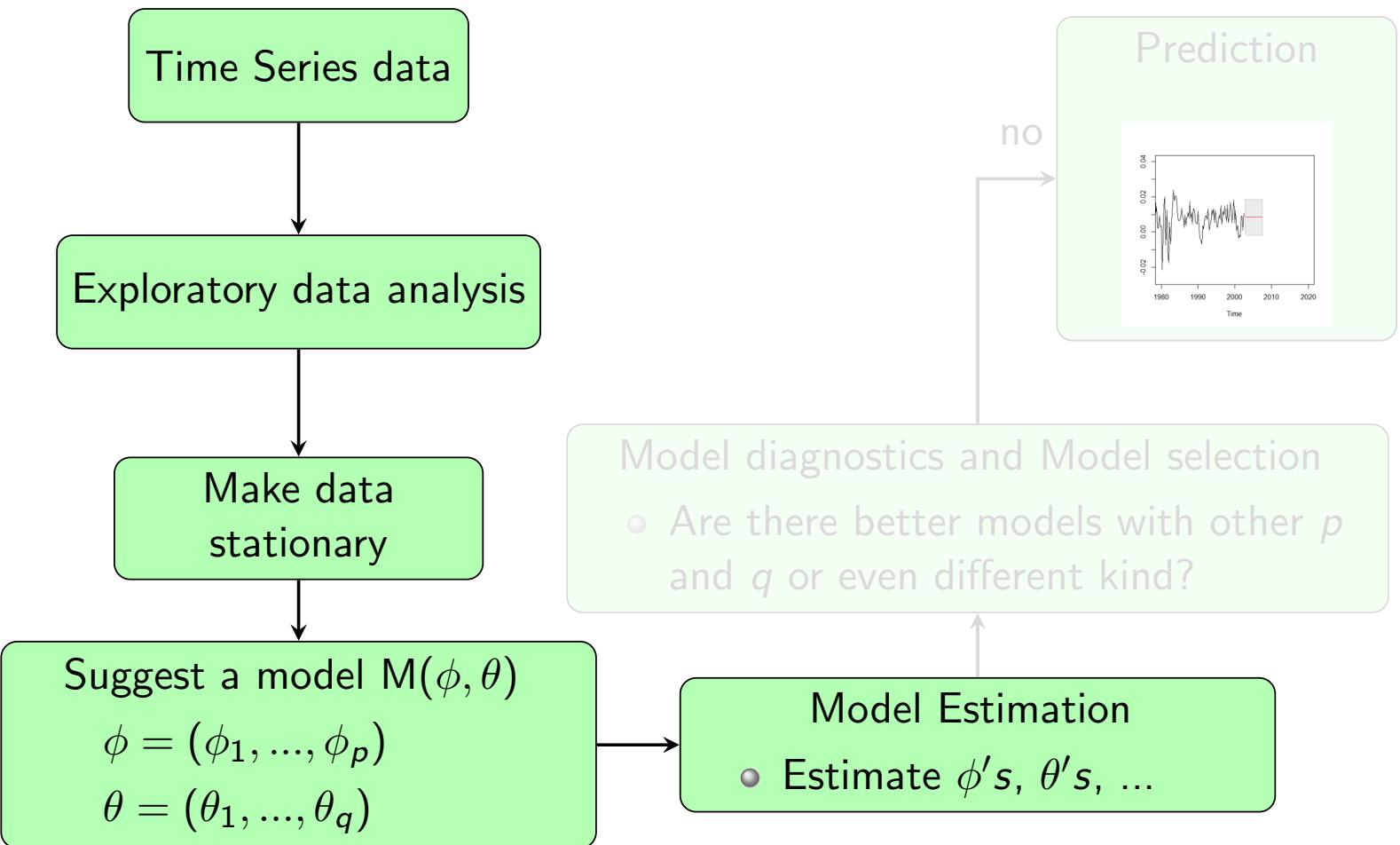
# Time domain: The Big Picture



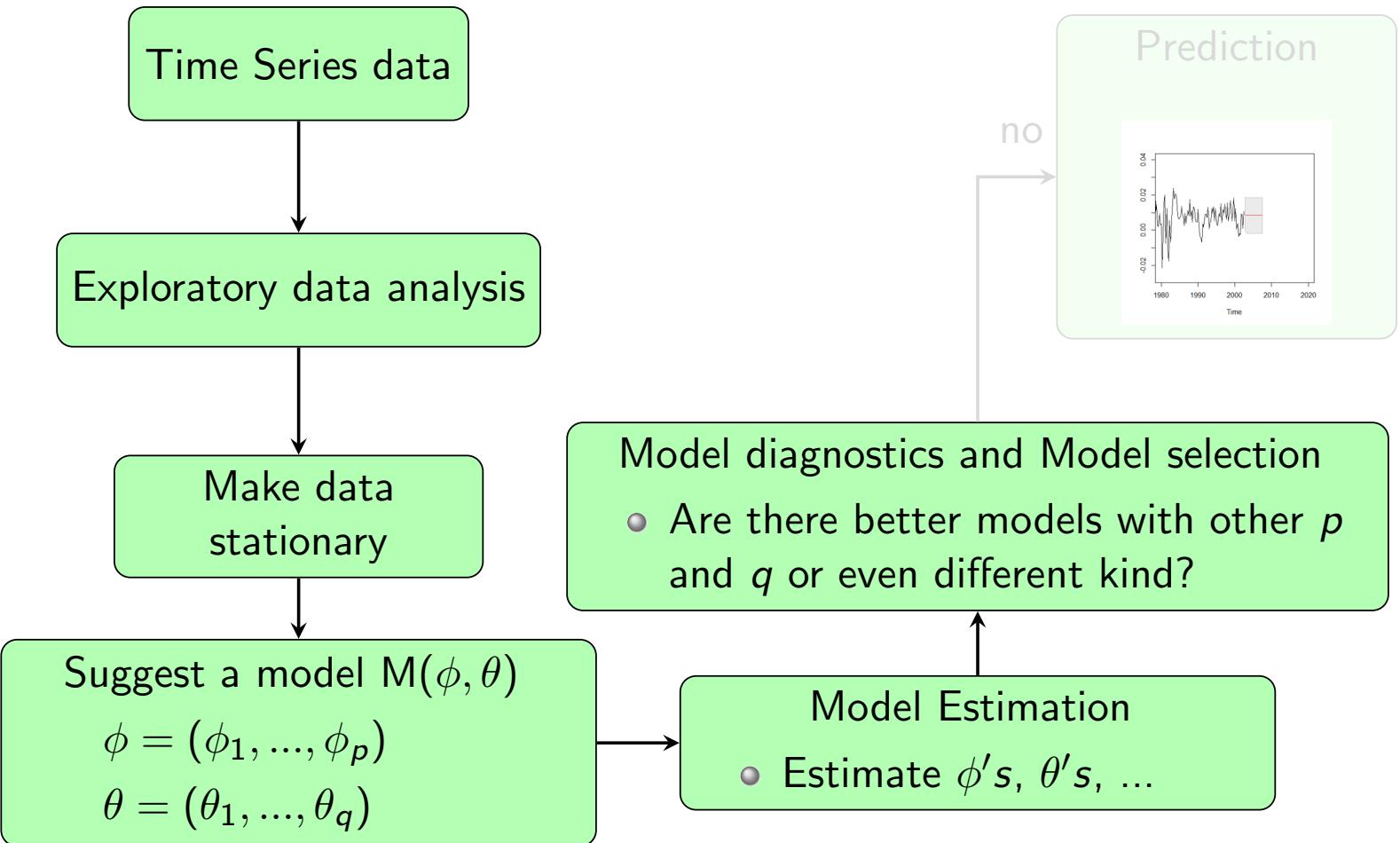
# Time domain: The Big Picture



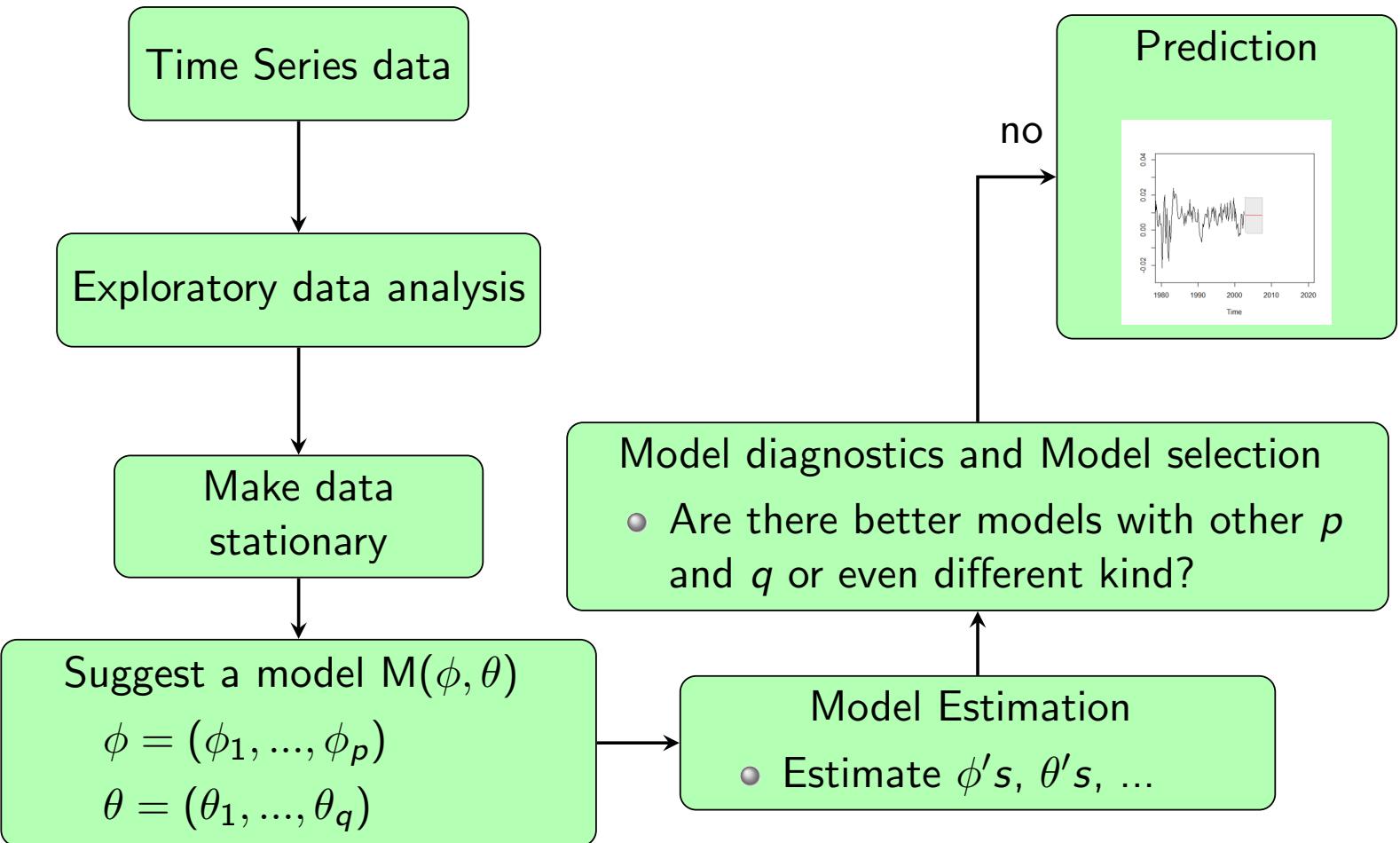
# Time domain: The Big Picture



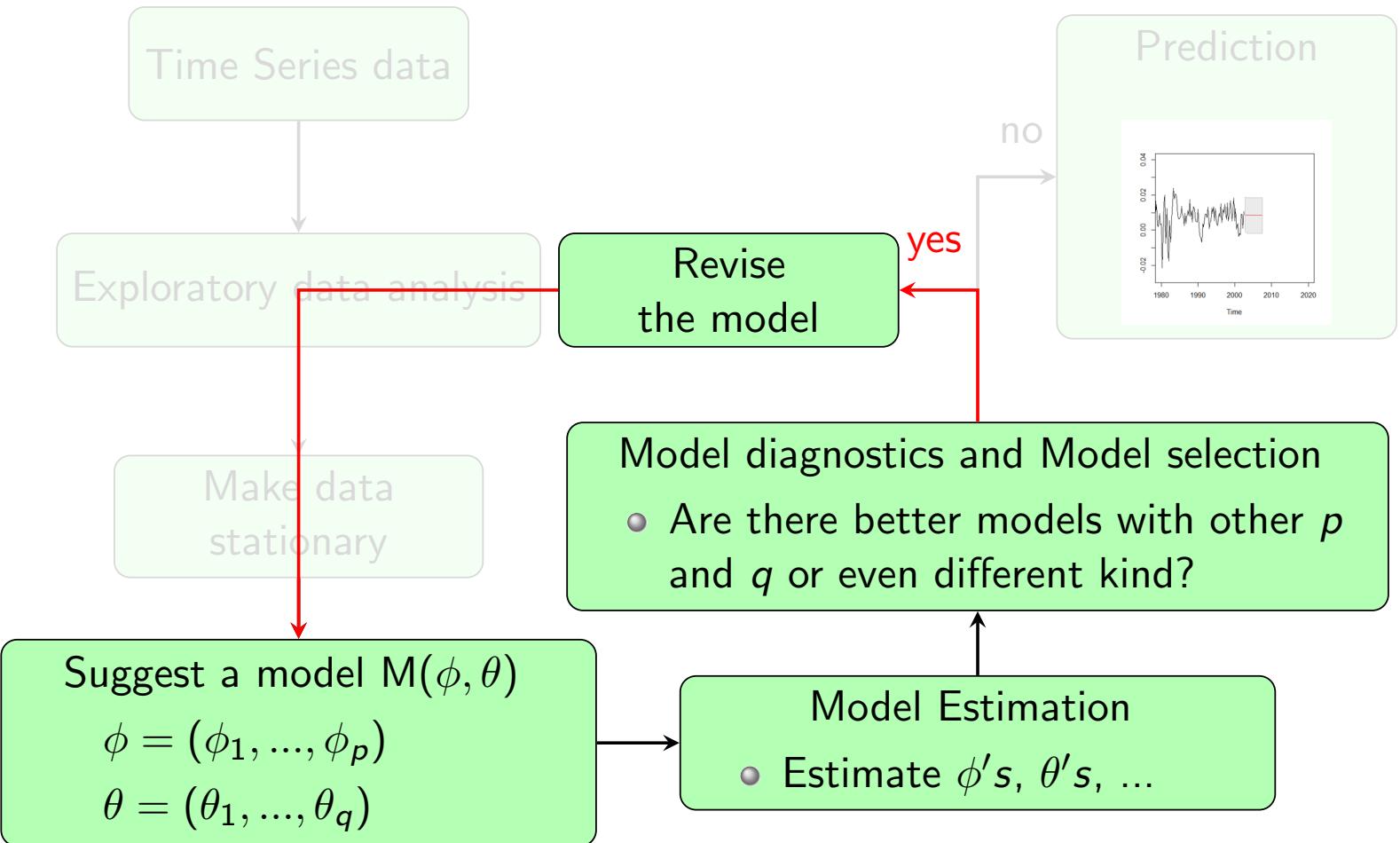
# Time domain: The Big Picture



# 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/1tzG35WSDWRhHWFA0cNOLLWUzoYqdn0GhZUwvXz3HhII/edit?usp=sharing>
  - ▶ **Difficult to find a group? Put your name in some cell.. I will merge you to someone**



# Course organization

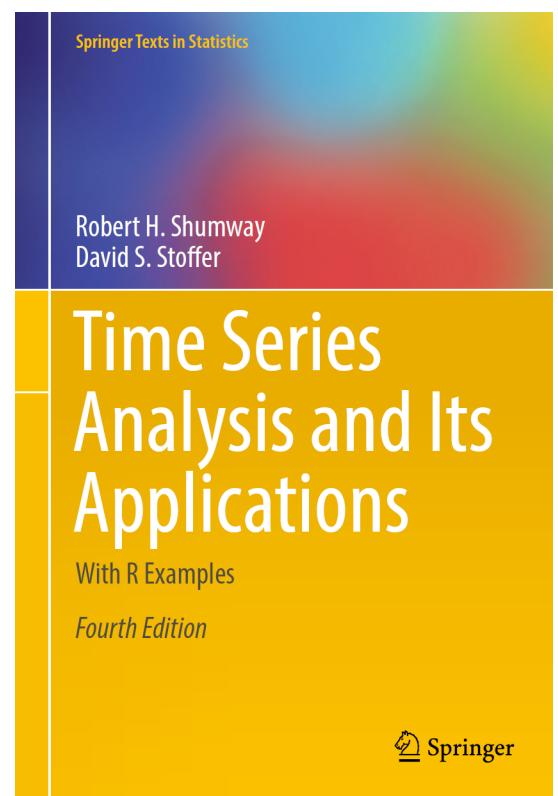
## Course literature:

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)

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

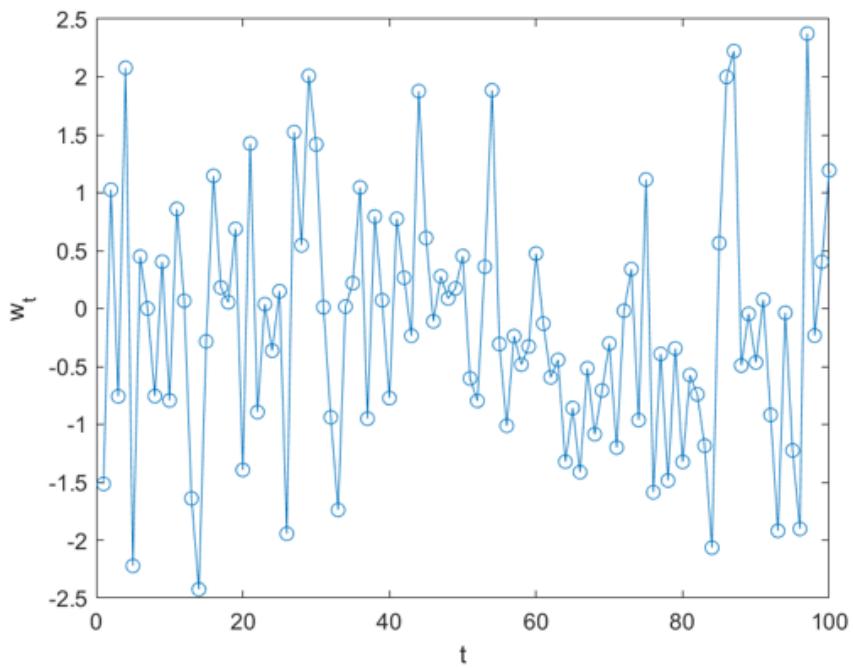
- Reminder:

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

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

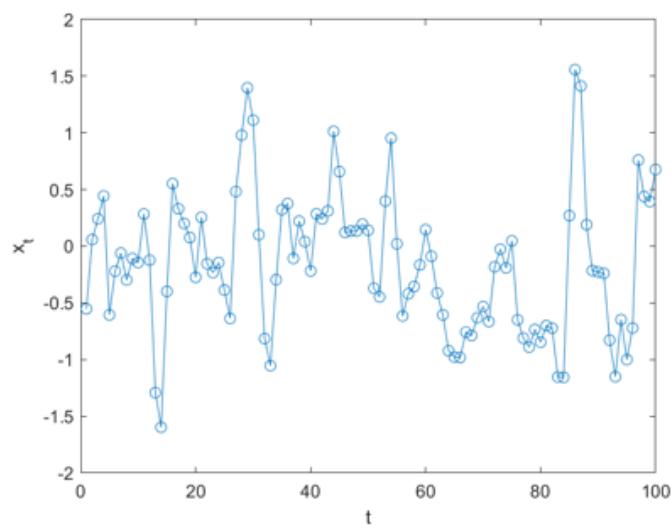
# White noise

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



## Moving average

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



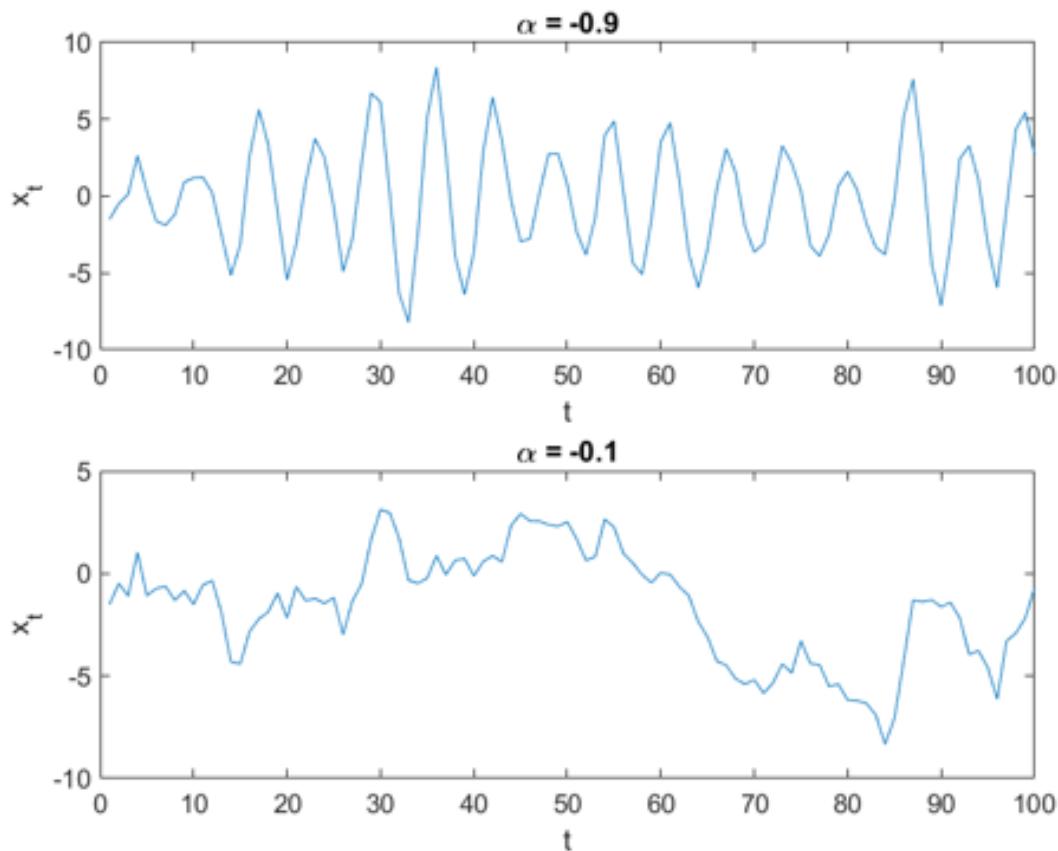
**Very Interesting Fact:** most stationary processes can be represented as a sum of lagged white noise:

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

# 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

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

- $\delta$  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

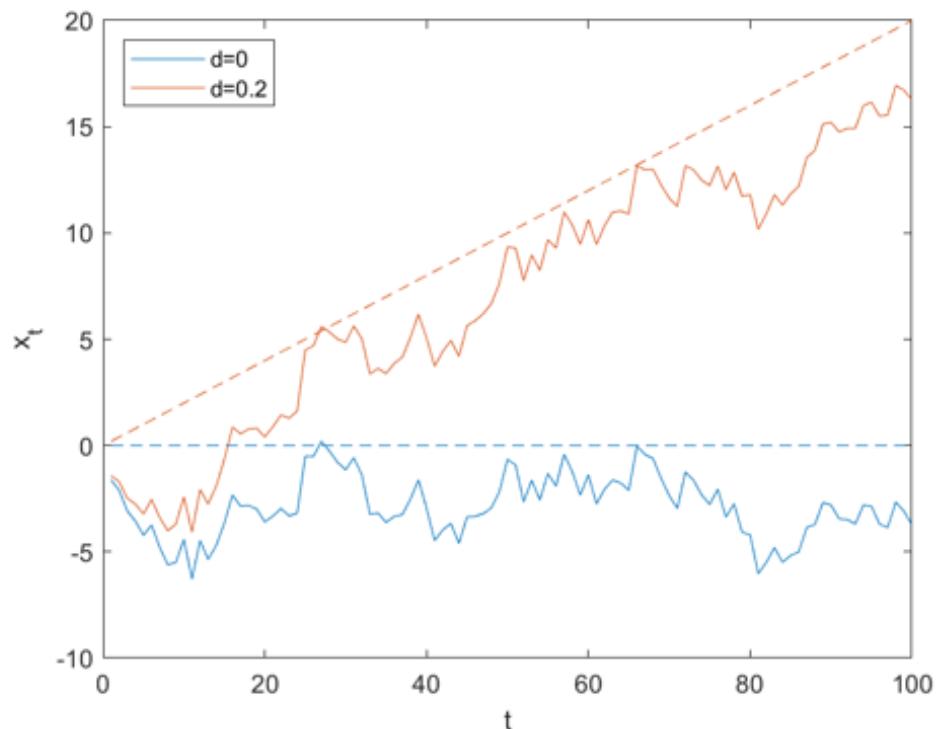
$$\delta=0 \text{ and } \delta=0.2$$

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

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

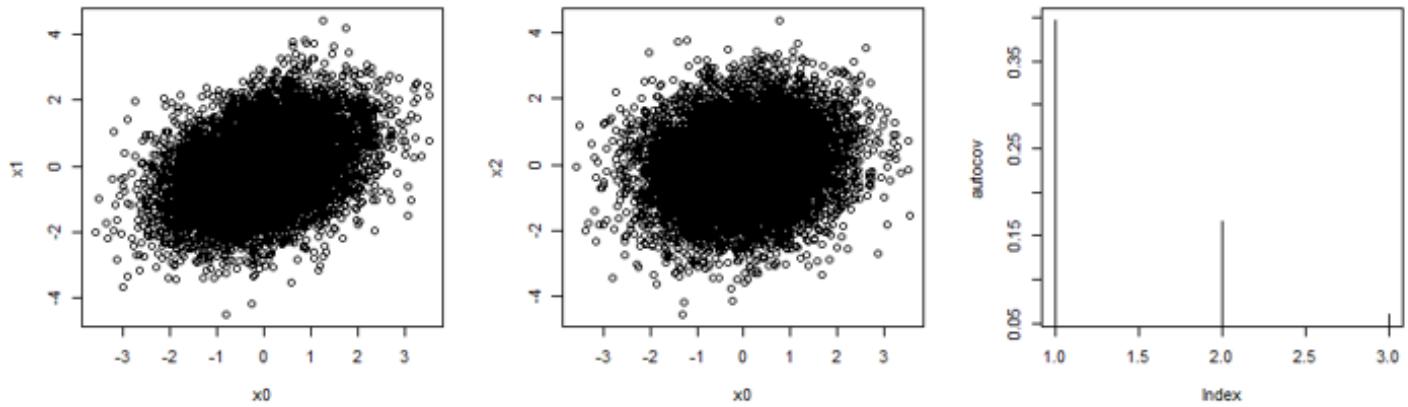
- Autocorrelation function (ACF)

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

# 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

## Lecture 2

# Time Series Analysis

Lecture 2: Exploratory analysis and Time Series Regression

**Tohid Ardestiri**

Linköping University  
Division of Statistics and Machine Learning

September 4, 2019



# 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

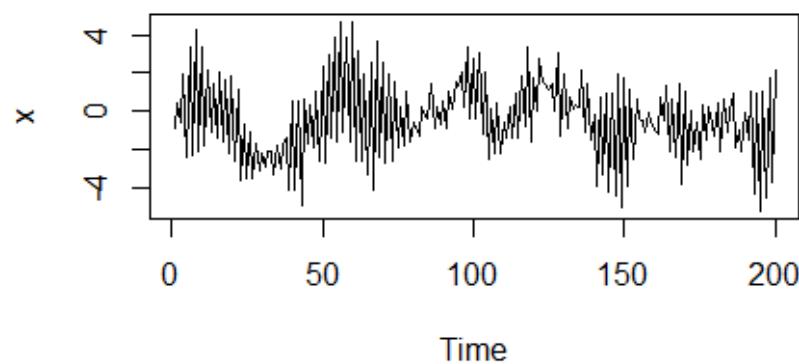
- White noise ✓
- Random walk  $x_t = \delta t + \sum_{j=1}^t w_j$
- Moving average  $x_t = 0.2w_{t-1} + 0.5w_t + 0.2w_{t+1}$

# Autocovariance

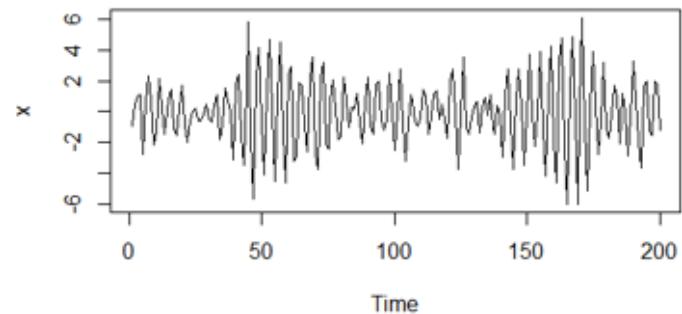
- Intuition:

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

$$\alpha = 0.9$$



$$\alpha = -0.9$$



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

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

# Autocovariance (read at home)

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

Mean function:

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

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

Variance  $\text{var}(x_t)$ :

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

When  $x_0 = 0$  and  $\sigma = 1$ ,  $\text{var}(x_t) = 1$

Autocovariance when  $Ex_0 = 0$  and  $w_t$  is uncorrelated with  $x_0$  for all  $t$ :

$$\begin{aligned}\text{cov}(x_t, x_{t-1}) &= E\{(x_t - Ex_t)(x_{t-1} - Ex_{t-1})\} = E(x_t x_{t-1}) = \\ &E\{(\alpha x_{t-1} + w_t)x_{t-1}\} = \alpha \text{var}(x_{t-1}) \text{ iff } E(w_t x_0) = 0\end{aligned}$$

$$\text{cov}(x_t, x_{t-h}) = \alpha^h \text{ iff } E(w_t x_0) = 0$$

# 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)

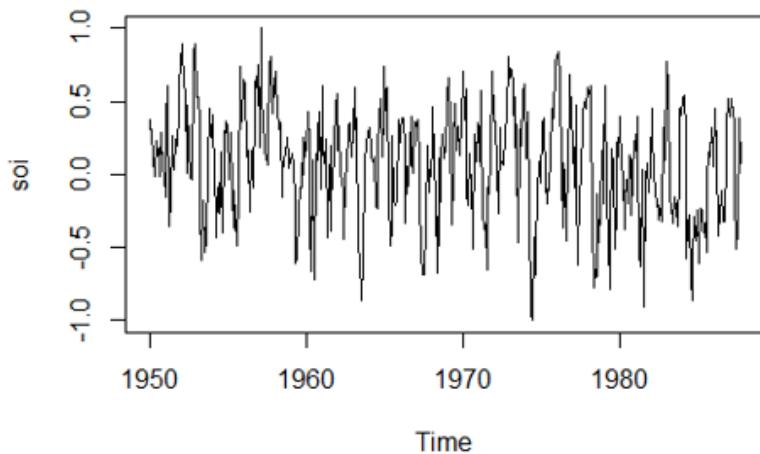
$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}$$

# Sample ACF

In R: `acf()`

**Example:** southern oscillation index (SOI)

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



```
> print(rho)
```

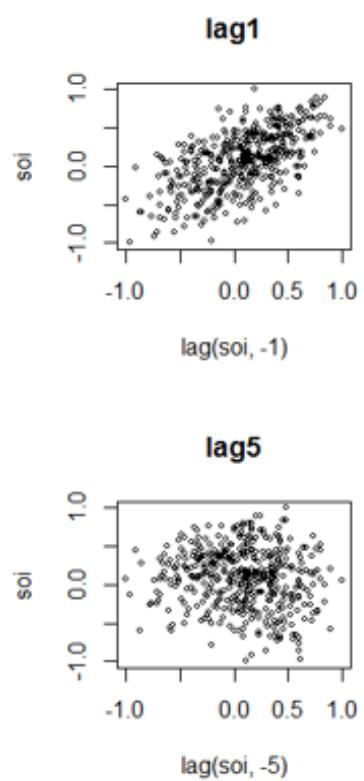
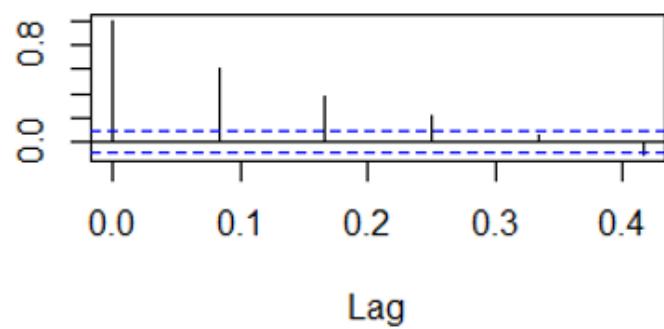
```
Autocorrelations of series 'soi', by lag
```

```
0.0000 0.0833 0.1667 0.2500 0.3333 0.4167  
1.000 0.604 0.374 0.214 0.050 -0.107
```

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

# Sample ACF

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



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

- n=10

Autocorrelations of series 'y1', by lag

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

- n=1000

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

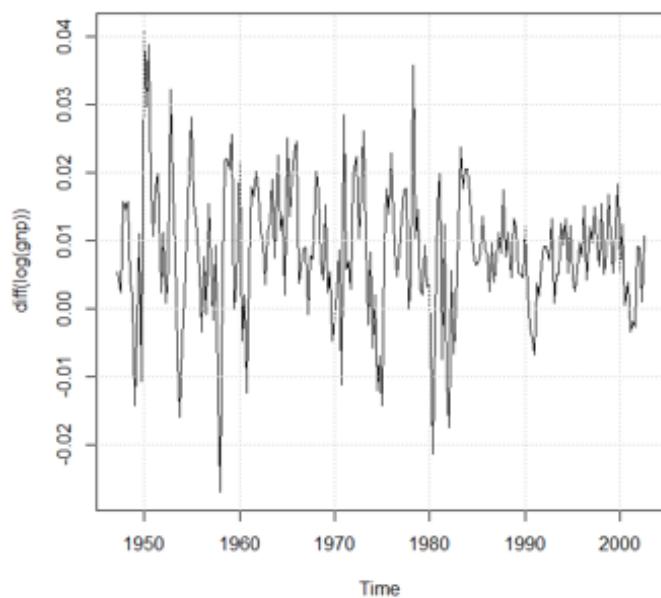
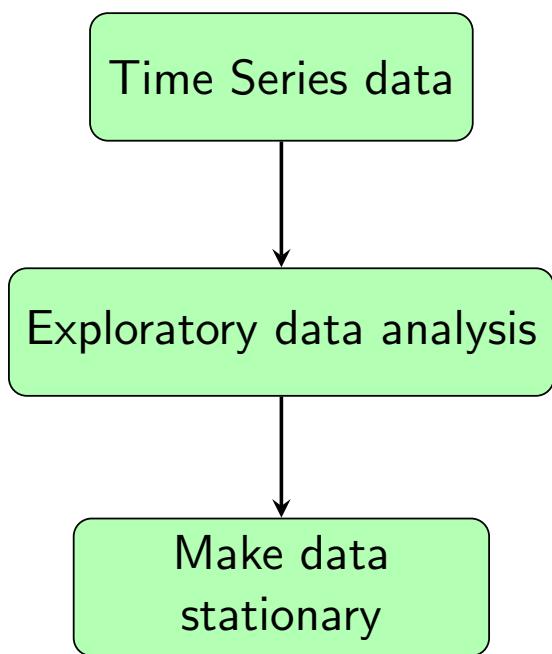
$$\mu = E(x_t) \approx \bar{x} = \frac{1}{n} \sum_{t=1}^n x_t$$

- 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  $\beta$

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

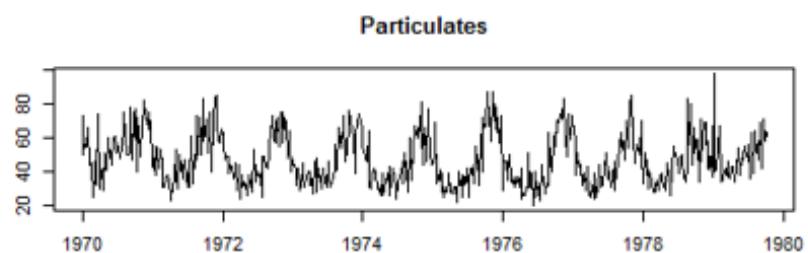
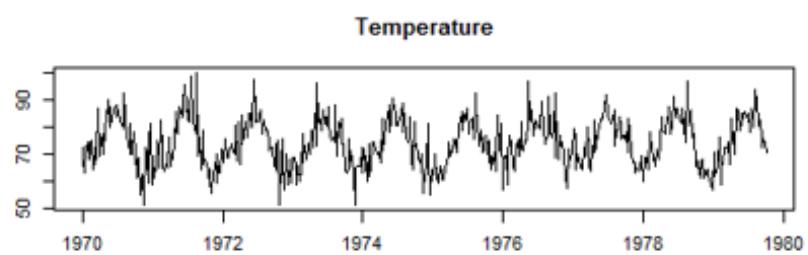
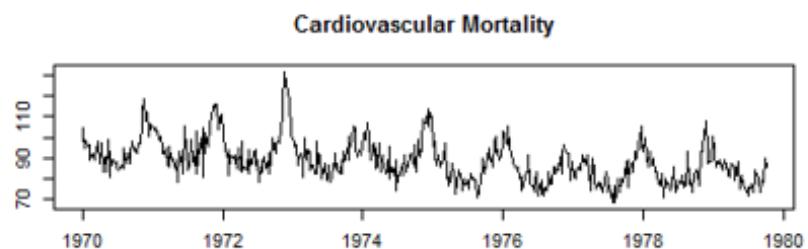
Least squares estimate is computed as

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

## Trend removal

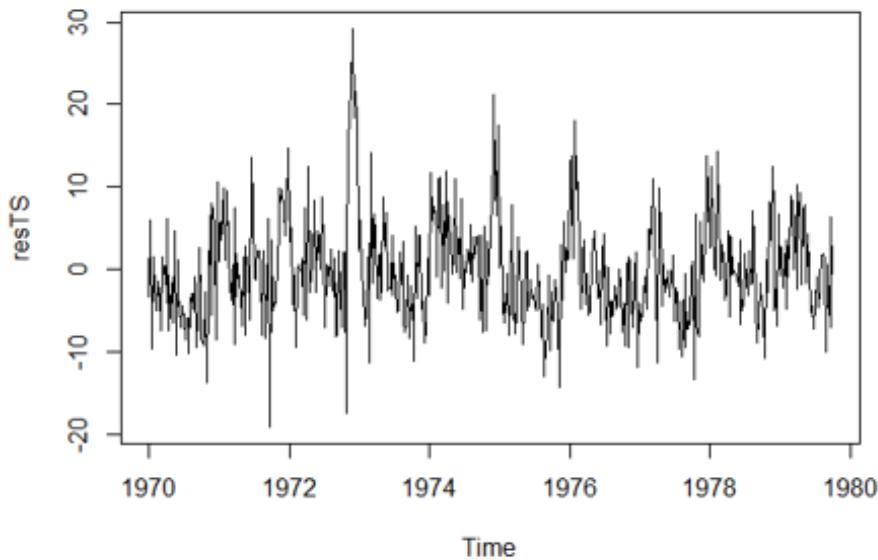
Example: Mortality

- $x_t$ : 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



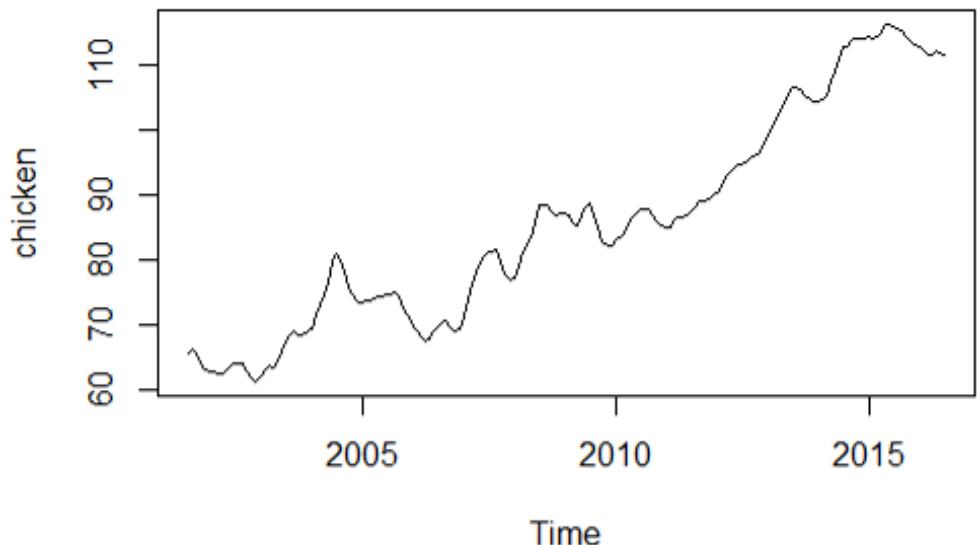
# Differencing

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

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

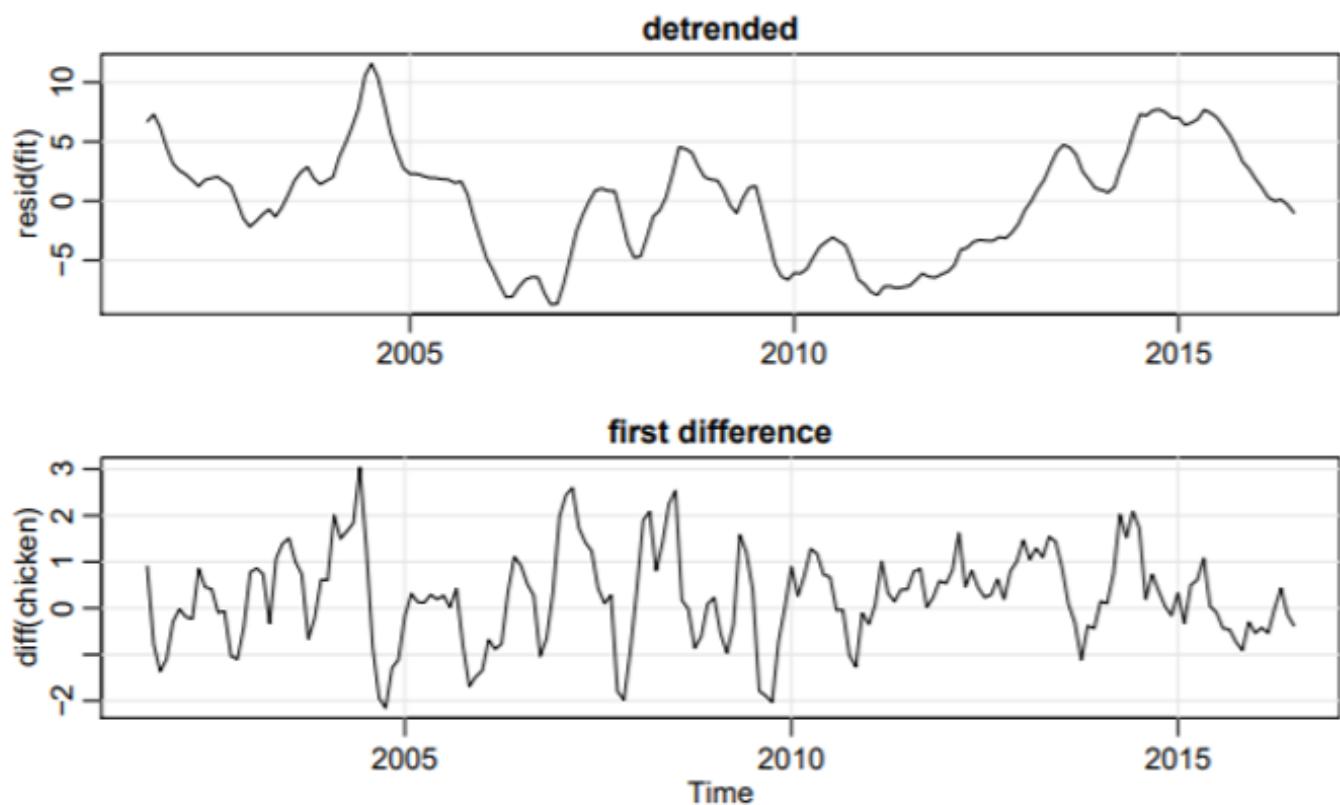
- **Property 1:** If  $\mu_t = \alpha_0 + \alpha_1 t$  then  $z_t$  is stationary
- **Property 2:** If  $\mu_t$  is random walk with a drift then  $z_t$  is stationary

**Example:**  
Chicken prices

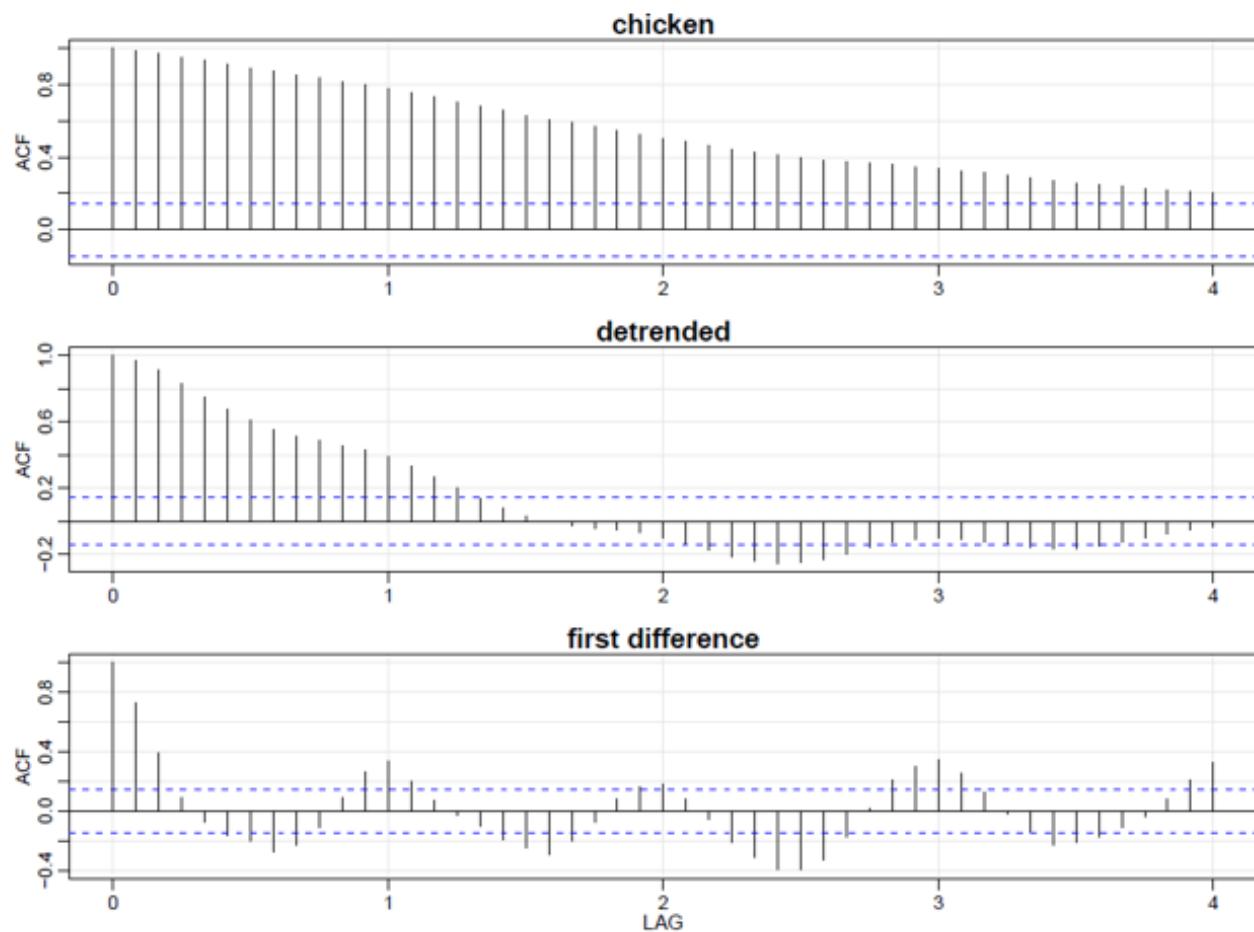


# Differencing

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  $p$

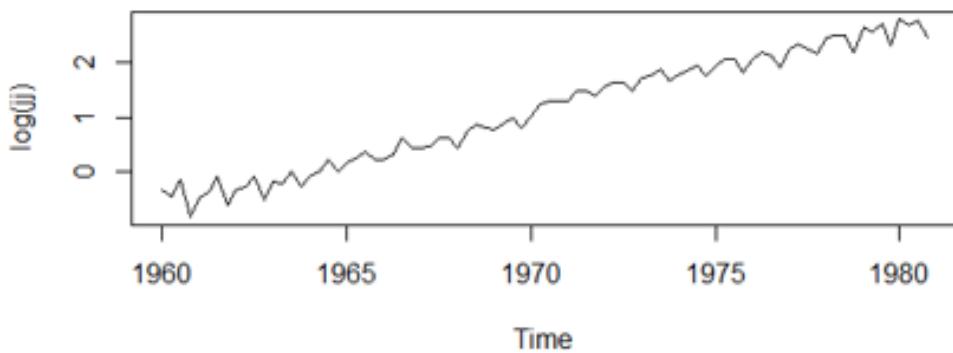
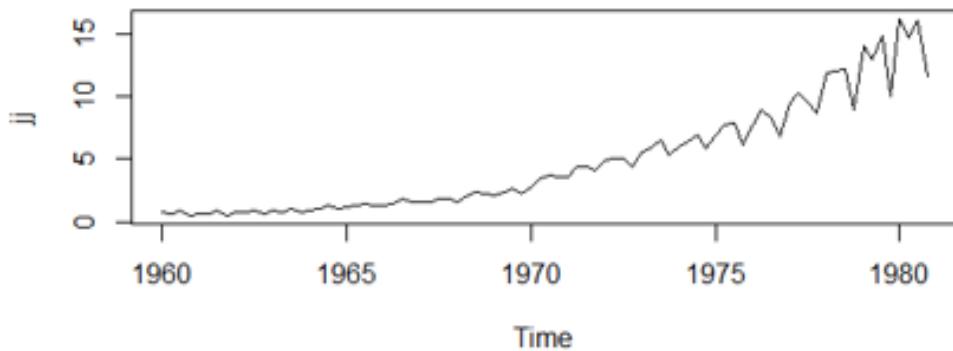
# Transformations

- Often used to stabilize variance
  - ▶ If  $\text{ex.var}(x_t) \neq \text{var}(x_s)$  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

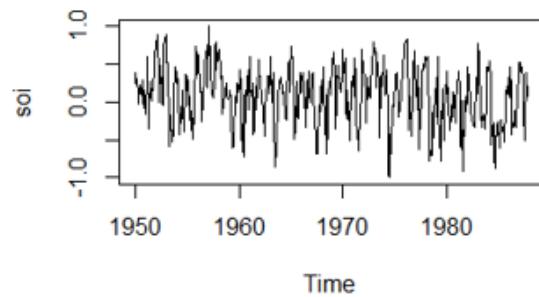
- Johnson & Johnson quarterly earnings



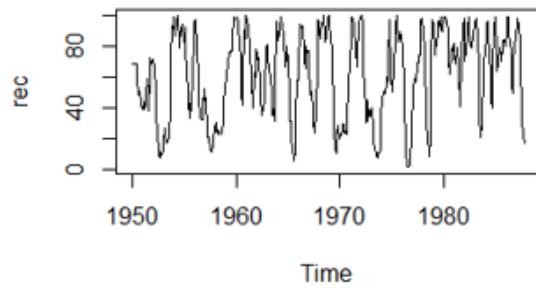
# Scatterplots

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

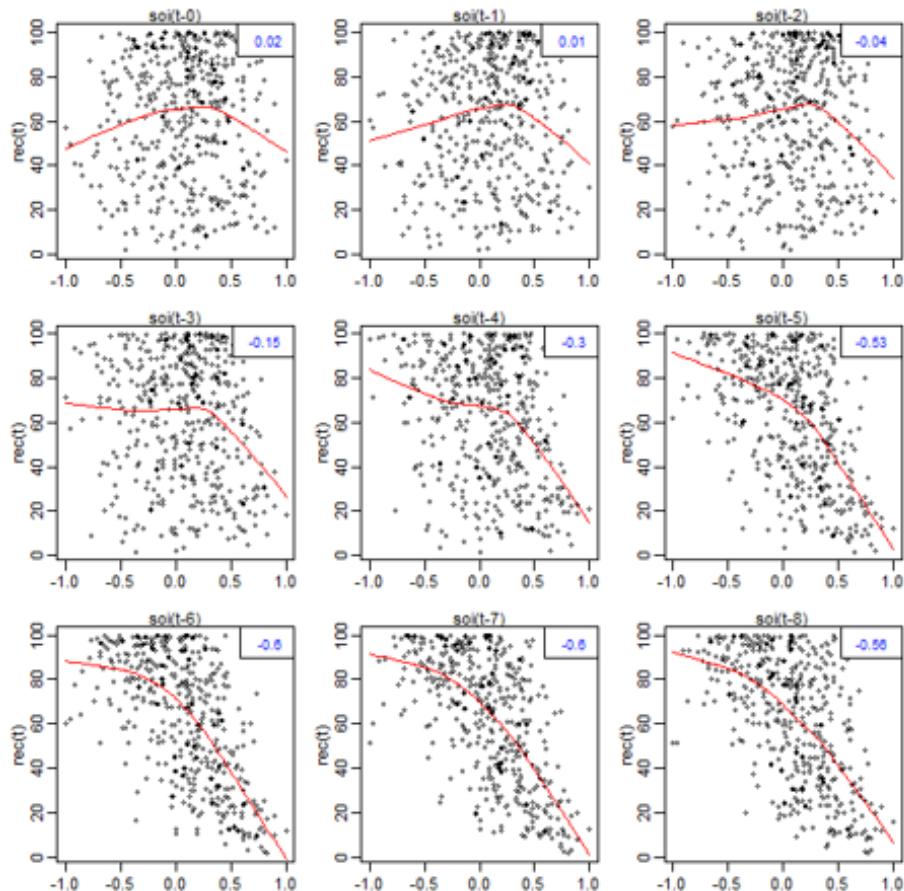
$$x_t = f(z_{t_1}, z_{t_2}, \dots, z_{t_q}) + w_t$$



- Example: SOI and Recruitment



# Scatterplots



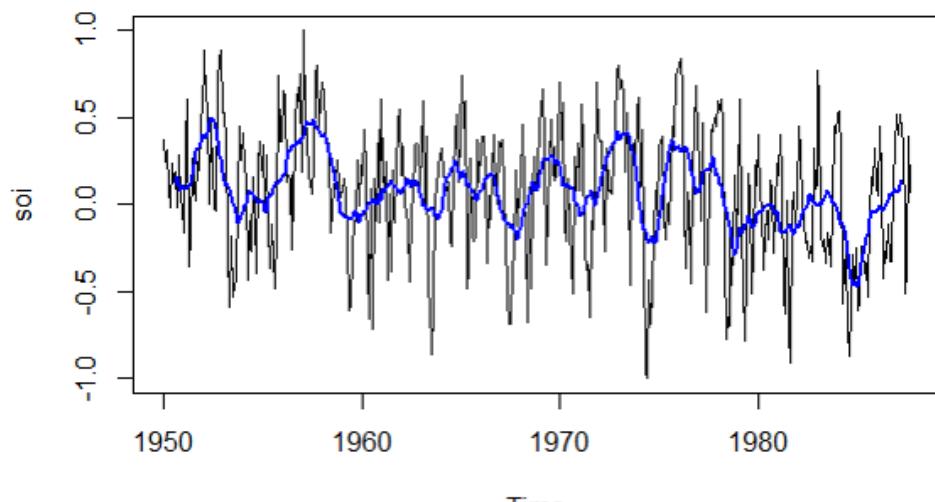
- Which relationships are nonlinear?
- Conclusion: include dummy variables  
 $I(\text{soi}(t - j) > 0)$  in the linear model

# 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?



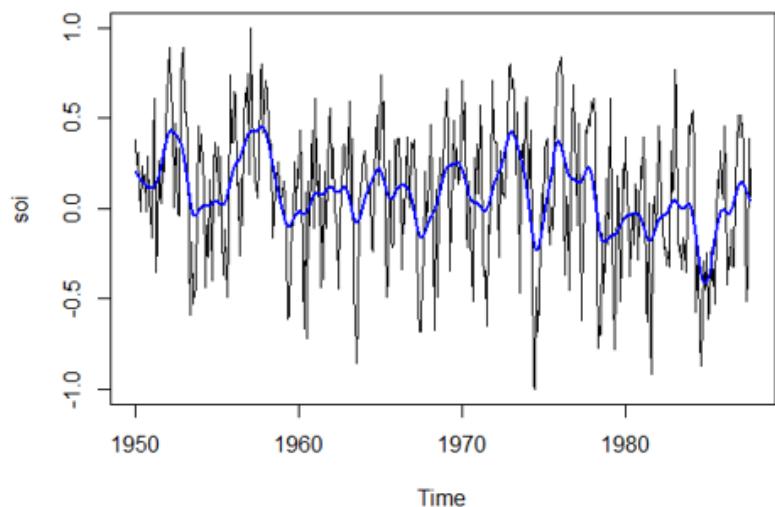
# Smoothing

More flexible models?

Example: kernel smoothers

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

Welcome to ML courses!!



# Home reading

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

## Lecture 3

# Time Series Analysis

## Lecture 3: Introduction to ARIMA

**Tohid Ardestiri**

Linköping University  
Division of Statistics and Machine Learning

September 6, 2019



# Recap

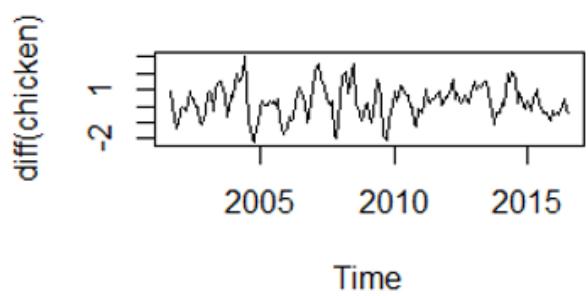
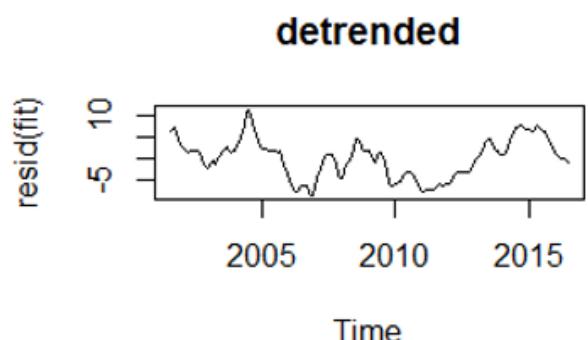
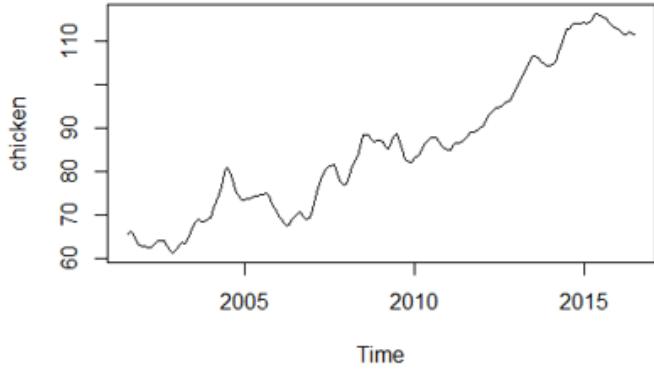
How to make data stationary?

- 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

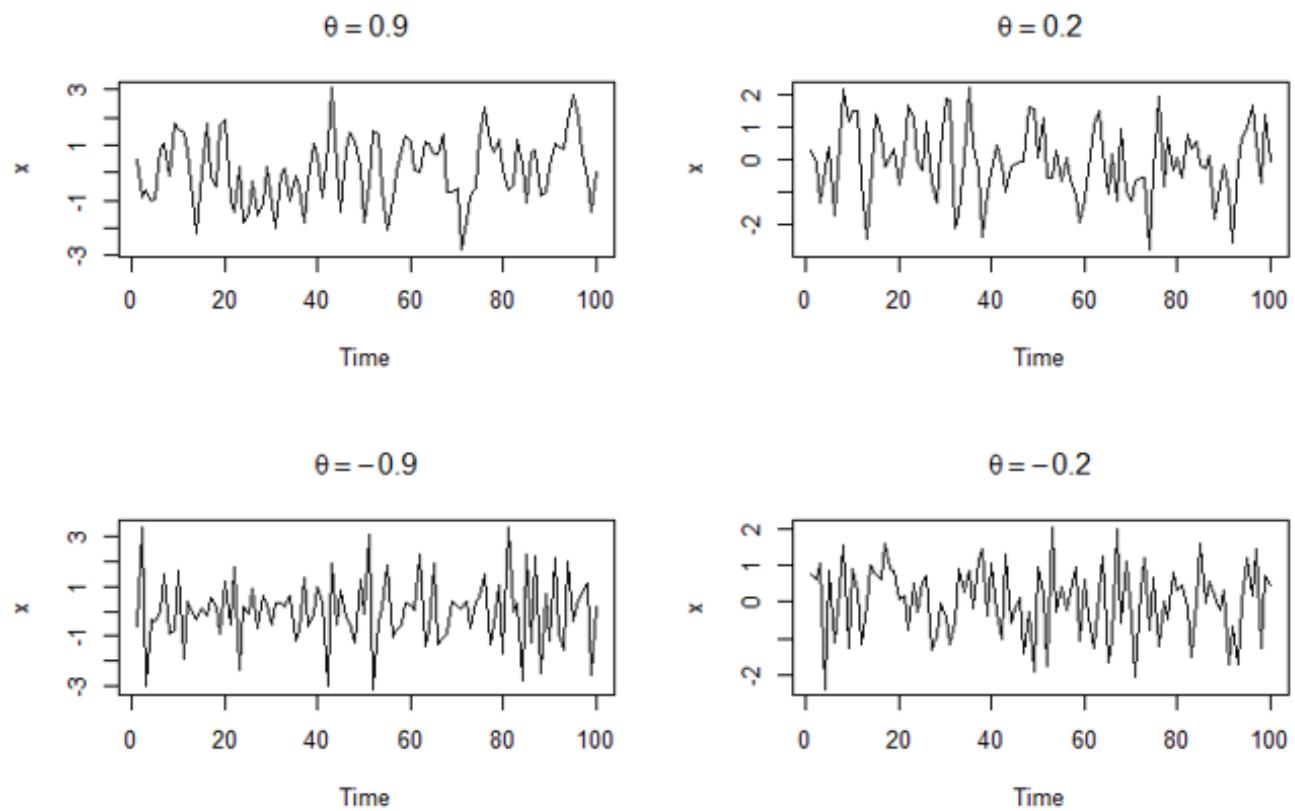
$$x_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j w_{t-j}$$

**Property:** It can be shown that

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

## Example: MA(1)

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



## Example: MA(1)

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

- Autocovariance and ACF

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

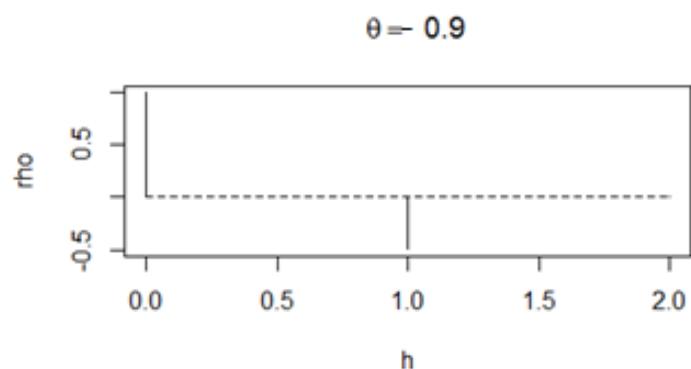
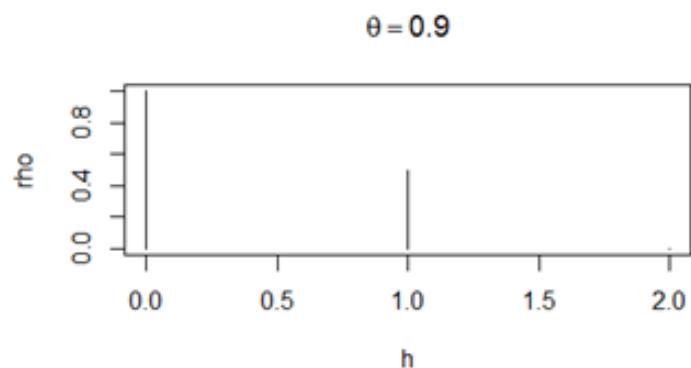
$$\rho(h) = \begin{cases} \frac{\theta}{1+\theta^2} & h = 1 \\ 0 & h > 1 \end{cases}$$

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
  - ▶  $w_t \sim wn(0, \sigma_w^2)$
  - ▶  $\phi_1, \dots, \phi_p$  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**

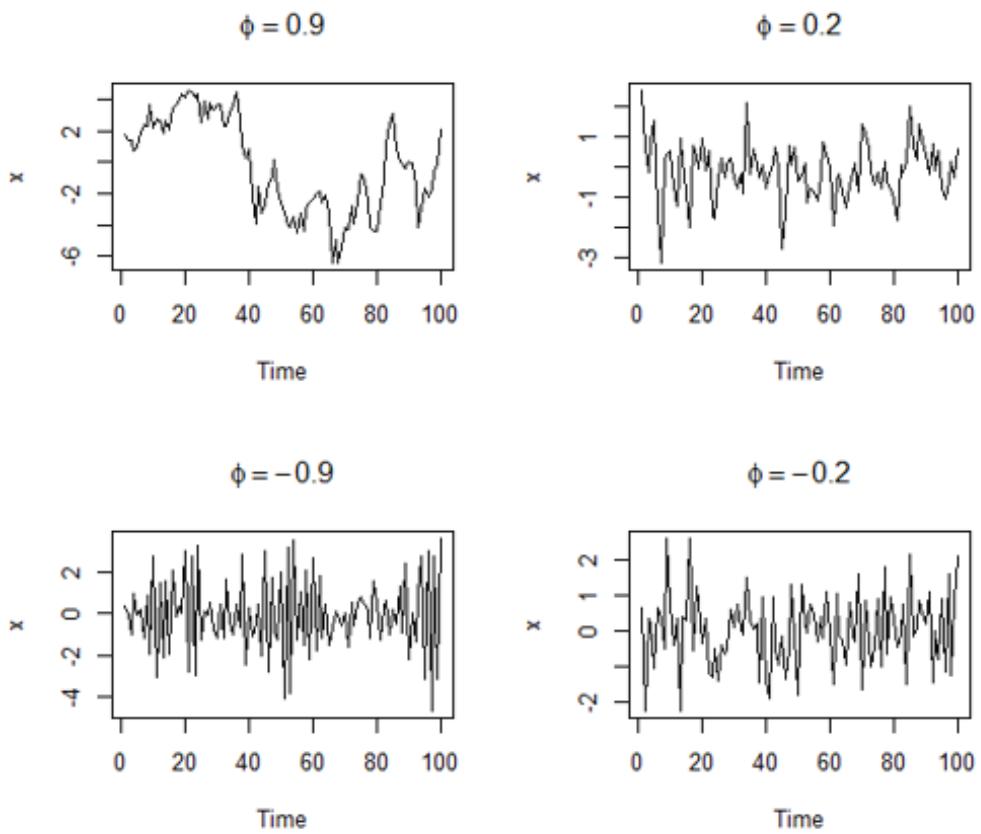
$$\phi(B) = 1 - \phi_1 B - \dots - \phi_p B^p$$

- 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)

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

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

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

Hence,

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

Also,

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

# AR(1)

## whiteboard

- **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

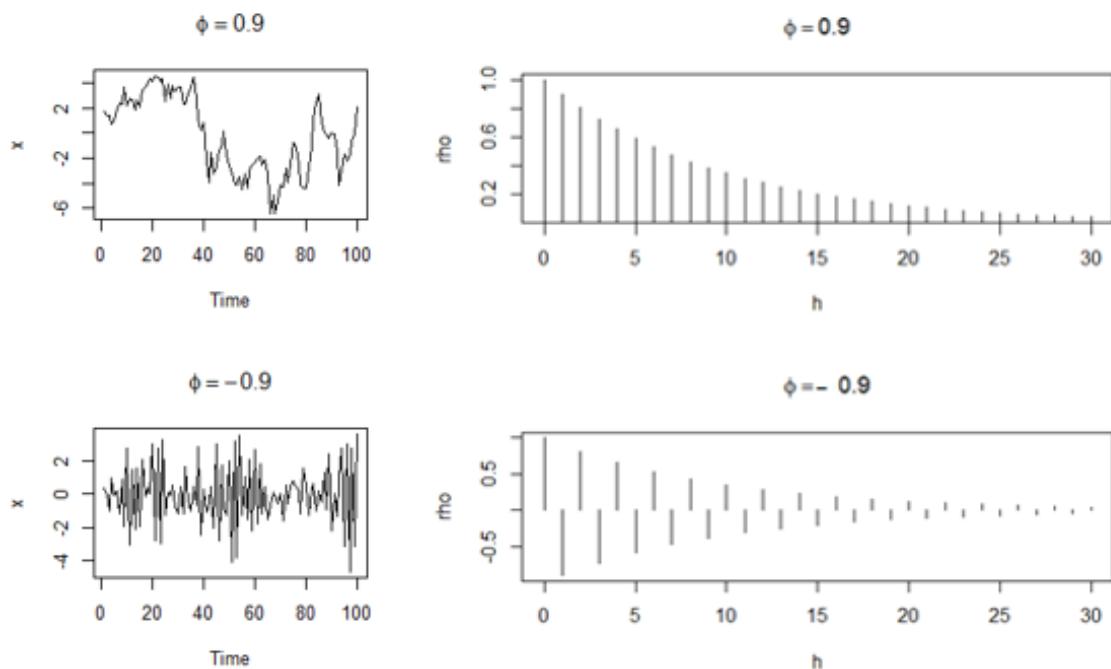
$$\gamma(h) = \frac{\sigma_w^2 \phi^h}{1 - \phi^2} \quad \rho(h) = \phi^h$$

for  $h \geq 0$ .

## Example: AR(1)

Autocovarince 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  $\sum_{j=0}^{\infty} |\psi_j| < \infty$ .

## MA equivalence

## Whiteboard

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

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

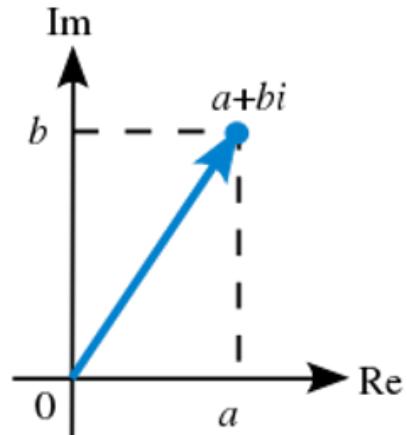
$$x_t - 0.5x_{t-1} = w_t - 0.5w_{t-1}$$

Looks like ARMA(1,1)!

## Reminder: complex numbers

- Imaginary unit  $i^2 = -1$
- Complex number  $z = a + bi$
- 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

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



## 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.  $|z'| > 1$

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

# Coefficient matching whiteboard

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

## Lecture 4

# Time Series Analysis

## Lecture 4: ARIMA models-1, Estimation

**Tohid Ardesthiri**

Linköping University  
Division of Statistics and Machine Learning

September 13, 2019

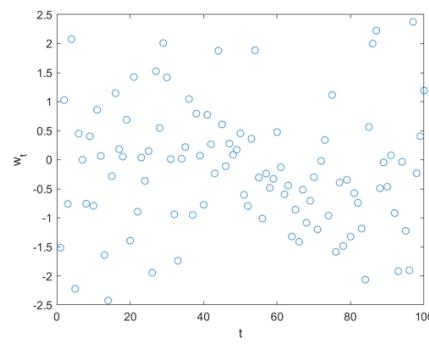


# White noise

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

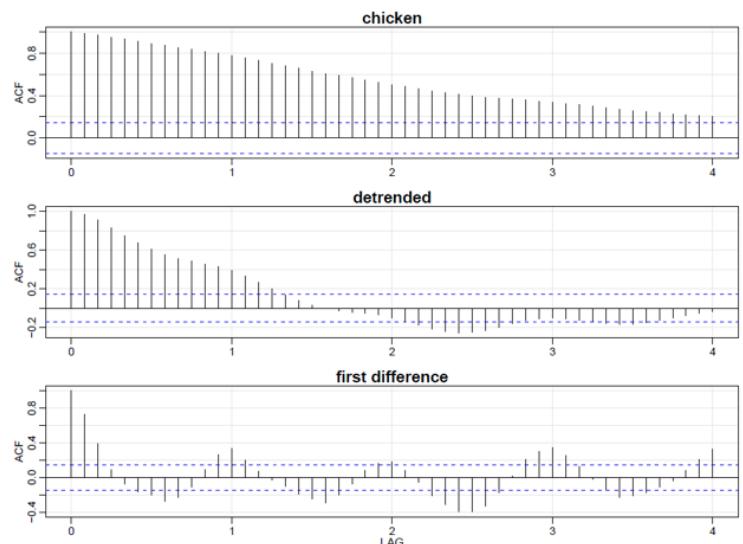
# Sample ACF

**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)**

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

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

- ▶  $w_t \sim wn(0, \sigma_w^2)$
- ▶  $\theta_1, \dots, \theta_q$  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
- ▶  $w_t \sim wn(0, \sigma_w^2)$
- ▶  $\phi_1, \dots, \phi_p$  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)

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

- ▶  $\phi_p \neq 0, \theta_q \neq 0$
- ▶ 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

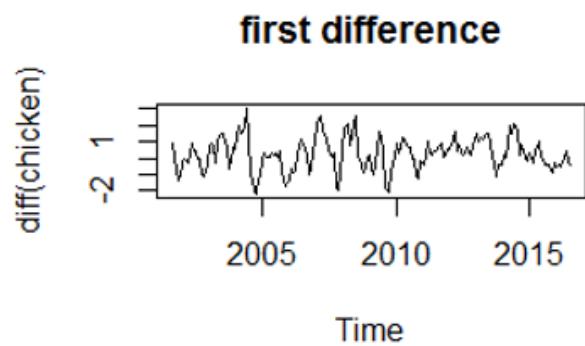
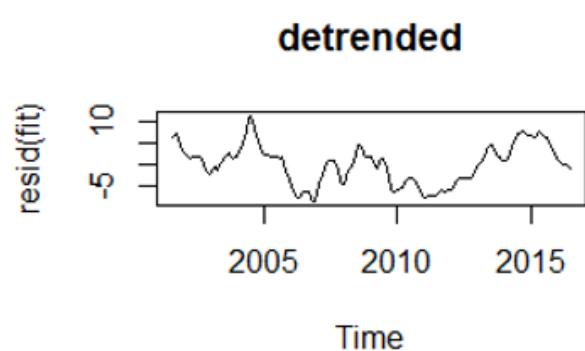
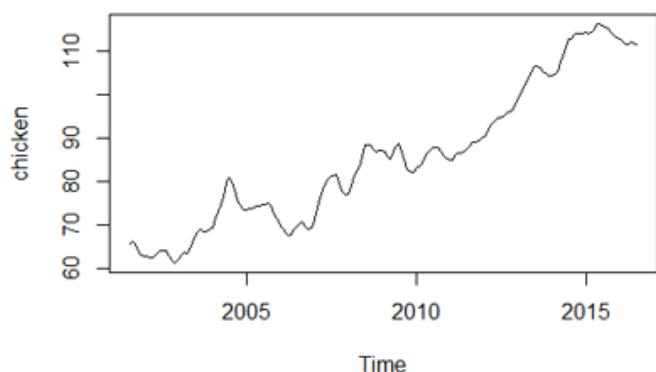
Assume  $x_t = \mu_t + y_t$  and  $y_t$  stationary

Differencing gives

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

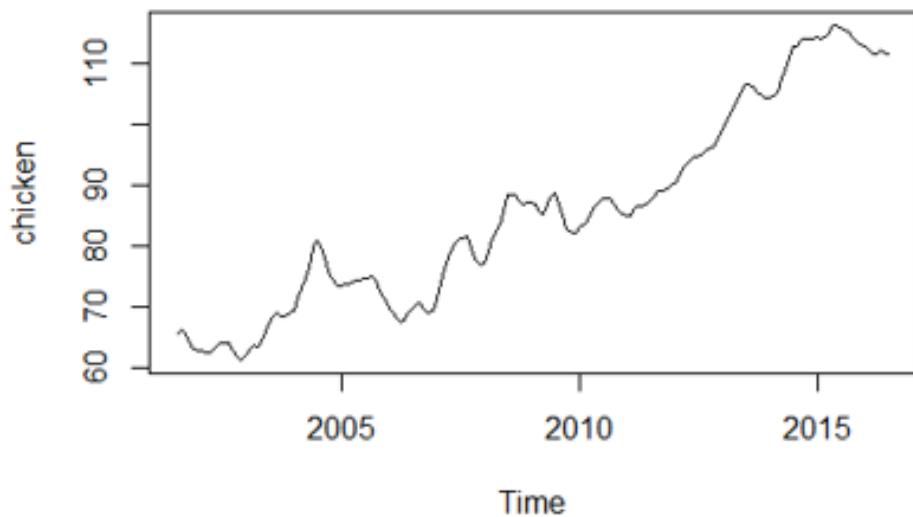
Also,

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



# 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  $\phi_1, \dots, \phi_p$  and  $\theta_1, \dots, \theta_q$
  - ▶ variance  $\sigma_w^2$  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  $\alpha$  given observations  $x_1, \dots, x_t$  is

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

- Maximum likelihood: Optimal  $\alpha$

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

- 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  $\alpha$  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  $\alpha$  given observations  $x_1, \dots, x_t$  is

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

- Maximum likelihood: Optimal  $\alpha$

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

- 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  $\alpha$

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

## 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

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

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

# ML for AR(1)

## Whiteboard

- 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  $\sigma^2$  explicit
  - ▶ Otherwise numerical optimization (unconstrained optimization)

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

# 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

## Lecture 5

# Time Series Analysis

## Lecture 5: ARIMA models-2 Estimation, PACF, Forecasting

**Tohid Ardestiri**

Linköping University  
Division of Statistics and Machine Learning

September 16, 2019



# Maximum likelihood estimation: reminder

**Time series data are NOT independent**

- Likelihood of  $\alpha$  given observations  $x_1, \dots, x_t$  is

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

- Maximum likelihood: Optimal  $\alpha$

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

- 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  $\alpha$

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

# 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

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

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

# ML for AR(1)

## Whiteboard

- 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  $\sigma^2$  explicit
  - ▶ Otherwise numerical optimization (unconstrained optimization)

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

# ARMA

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

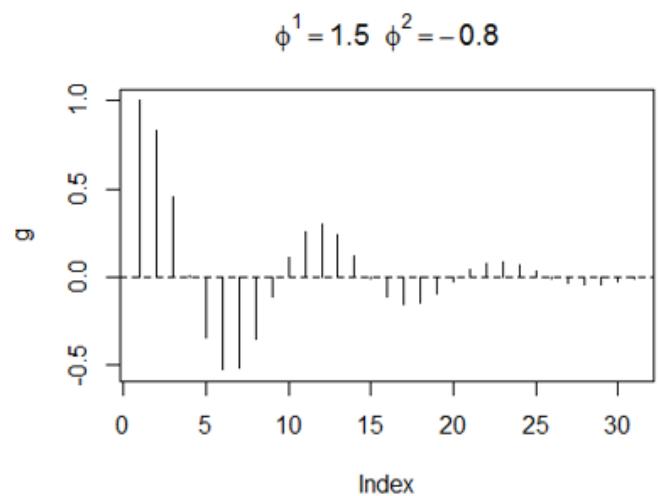
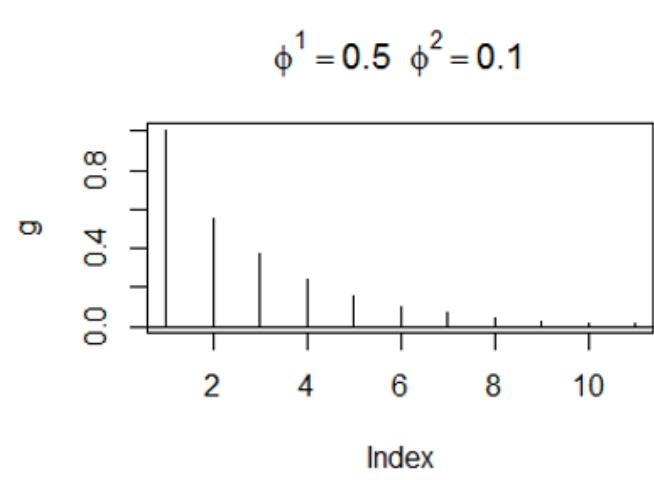
$$x_t = \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + w_t + \theta_1 w_{t-1} + \dots + \theta_q w_{t-q}$$

- ▶  $\phi_p \neq 0, \theta_q \neq 0$
- ▶ 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)



## 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  $\psi_j$  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

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

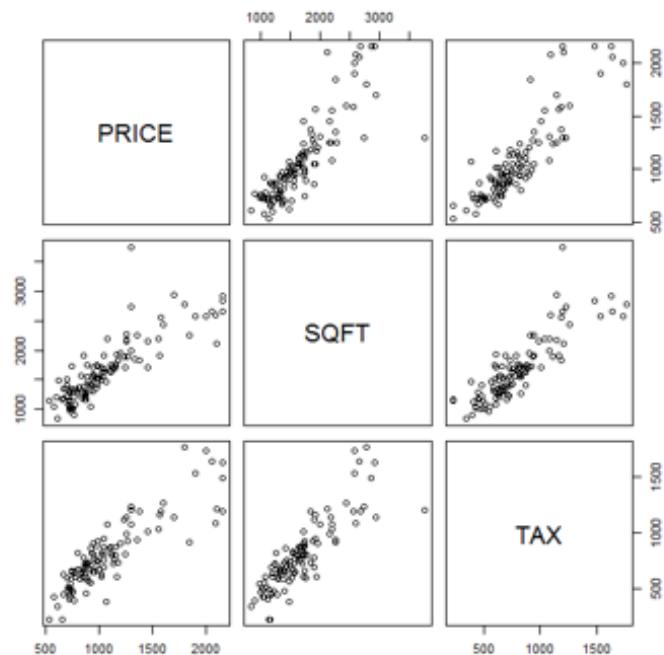
- Also,

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

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

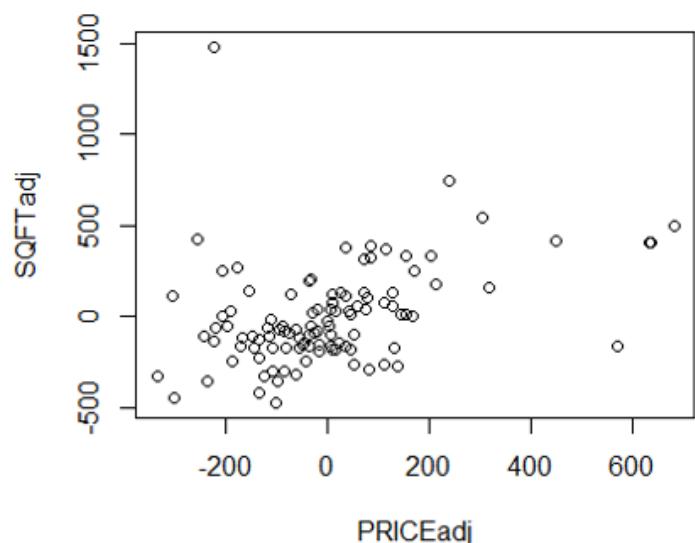
# Partial autocorrelation

- **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  $\alpha$ ,  $\beta$  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)

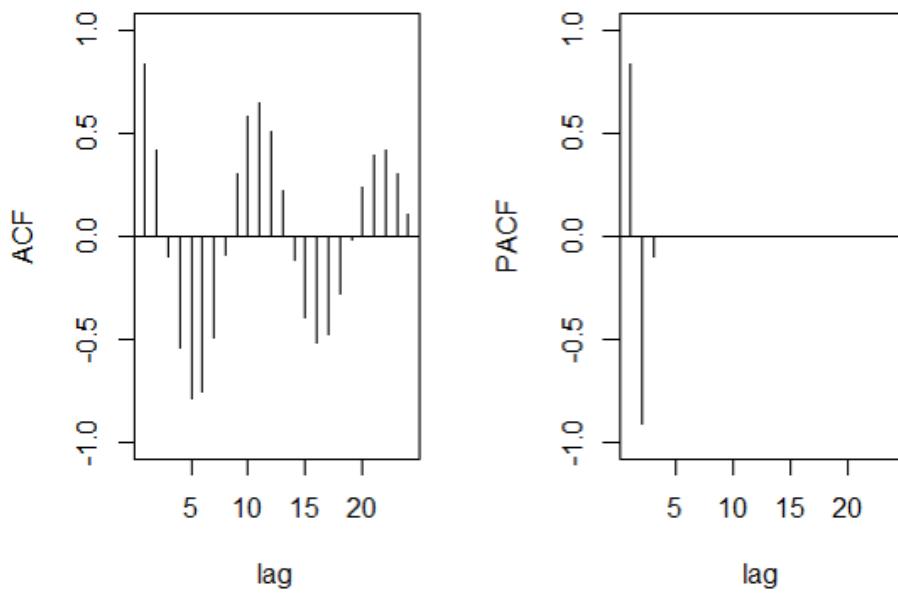
$$x_t = \sum_{j=1}^p \phi_j x_{t-j} + w_t$$

- 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)  $\phi_1 = 1.5$ ,  $\phi_2 = -0.75$ ,  $\phi_3 = -0.1$

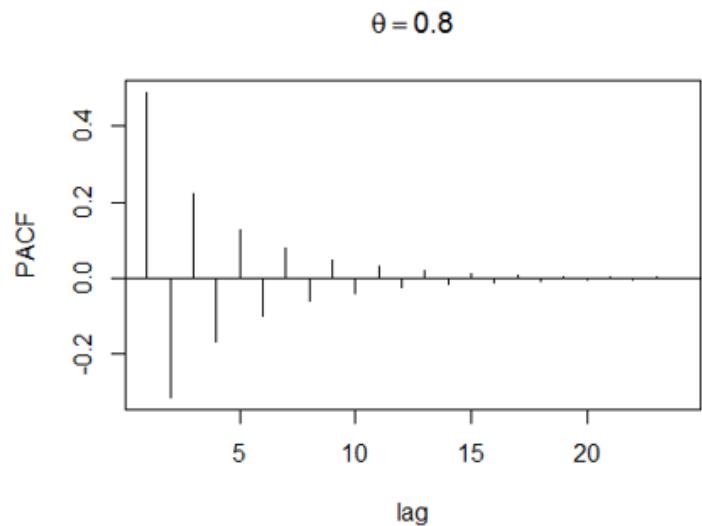


## PACF for MA(1)

- 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

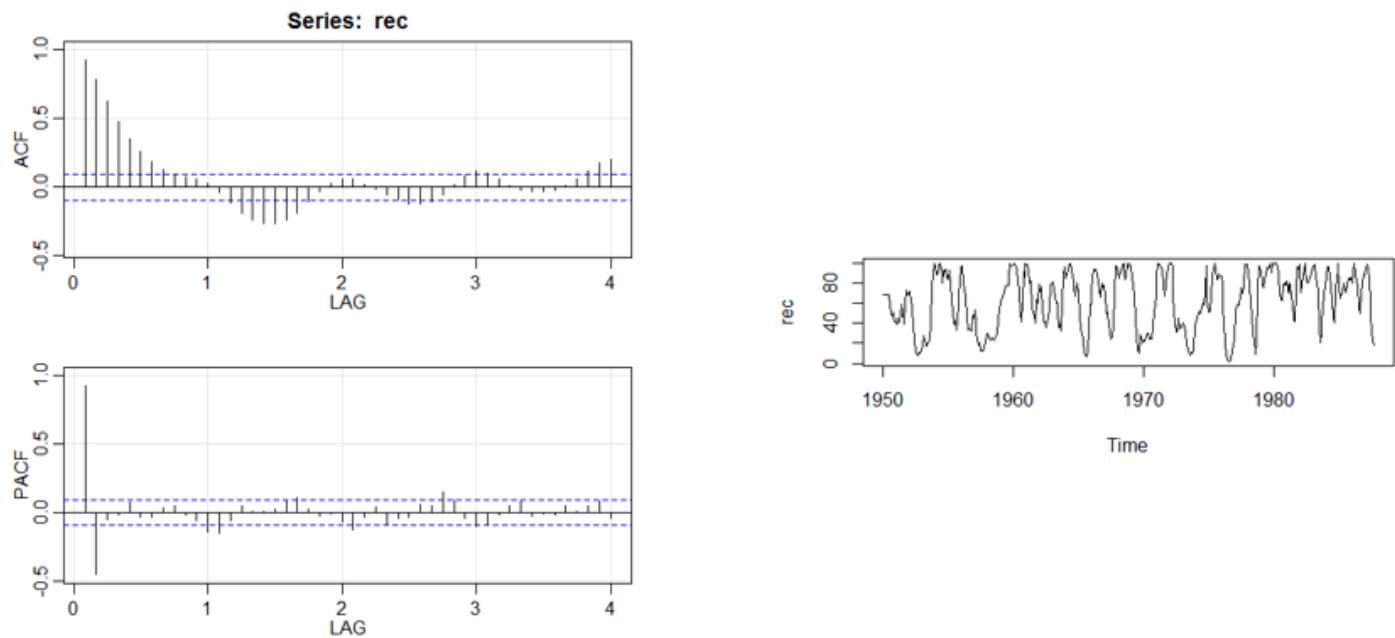
## Empirical ACF (EACF)

- $k > p$  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 o

2 o o x x o 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 o

4 x x o 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 o

6 x x x o 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

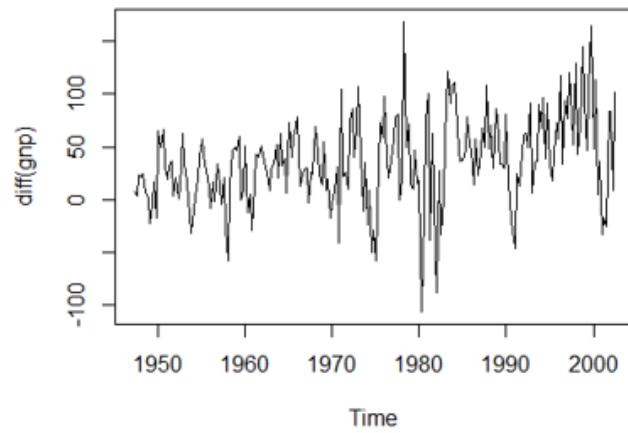
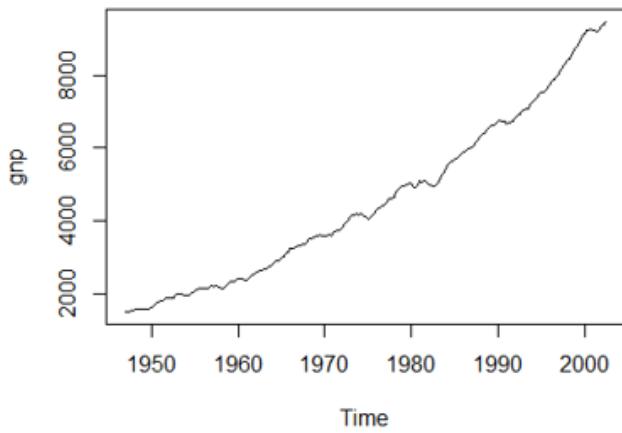
- Which model is suitable?
  - ▶ What is  $p, d, q$  in ARIMA(p,d,q)?
  - ▶  $d$  is defined before!
  - ▶
- Step 1: Check ACF, PACF and EACF to define a few tentative models

# 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

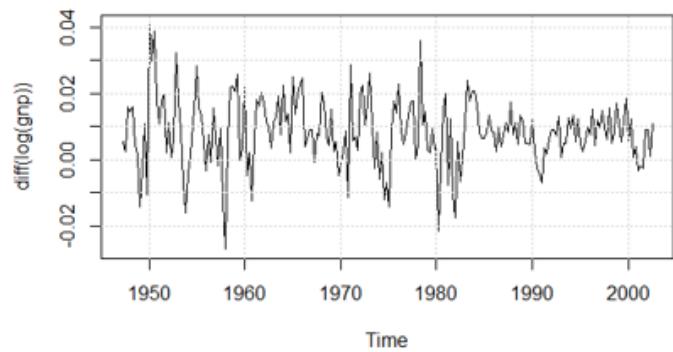
# Model selection

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



# Model selection

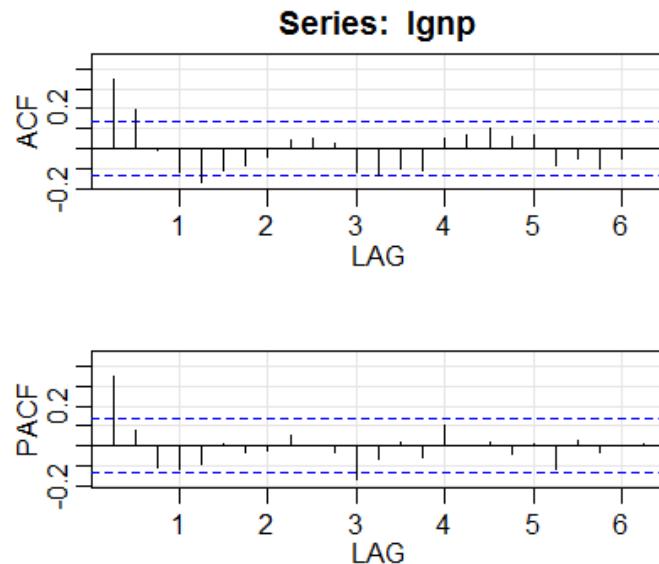
- **Example:** GNP data
  - ▶ Taking log and then differencing → still not perfect, but ... keep it as is.



```
> 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

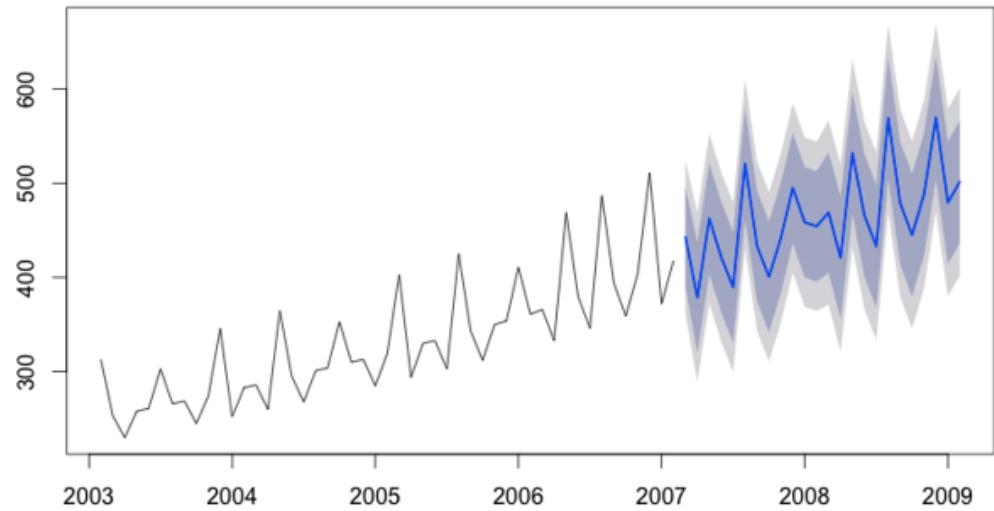
```
> TSA::eacf(lgnp)
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 x x o o x o o o o o o o o o o
1 x x o o o o o o o o o o o o o
2 x x o o o o o o o o o o o o o
3 x o x o o o o o o o o o o o o
4 x o x o o o o o o o o o o o o
5 o x x o x o o o o o o o o o o
6 x x x x o o o o o o o o o o o
7 x x o x o o x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

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

- Prediction equations
  - ▶ Find  $\alpha$ '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  $\Gamma_n$  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  $\phi_{nn}$  by solving  $\Gamma_n \phi_n = \gamma_n$

# One-step-ahead

- Mean square prediction error (MSPE)

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

- Confidence intervals for  $x_{n+1}$

$$\hat{x}_{n+1}^n \pm \alpha \sqrt{P_{n+1}^n}$$

- 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  $m \rightarrow \infty$ ?

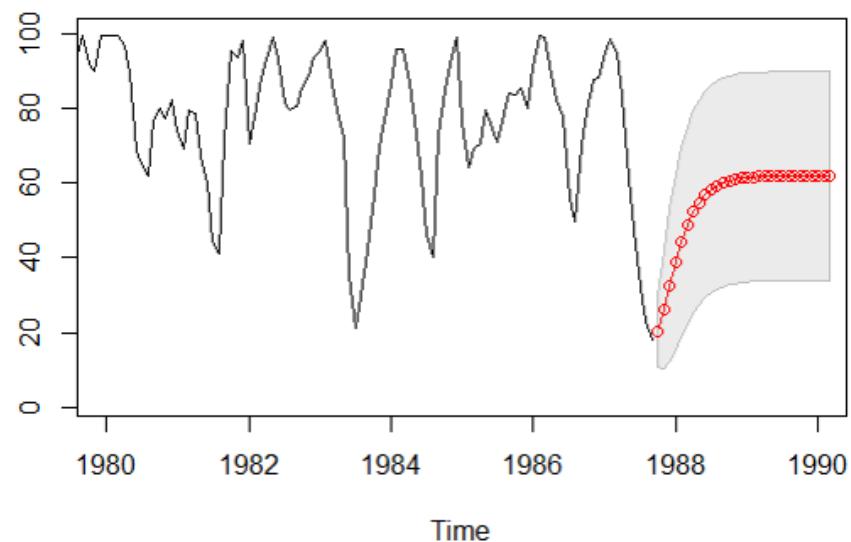
$$\tilde{x}_{n+m} \rightarrow 0(\text{or } \mu)$$

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

## 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

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

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

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

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

## Lecture 6

# Time Series Analysis

Lecture 6: ARIMA models summary  
State space models

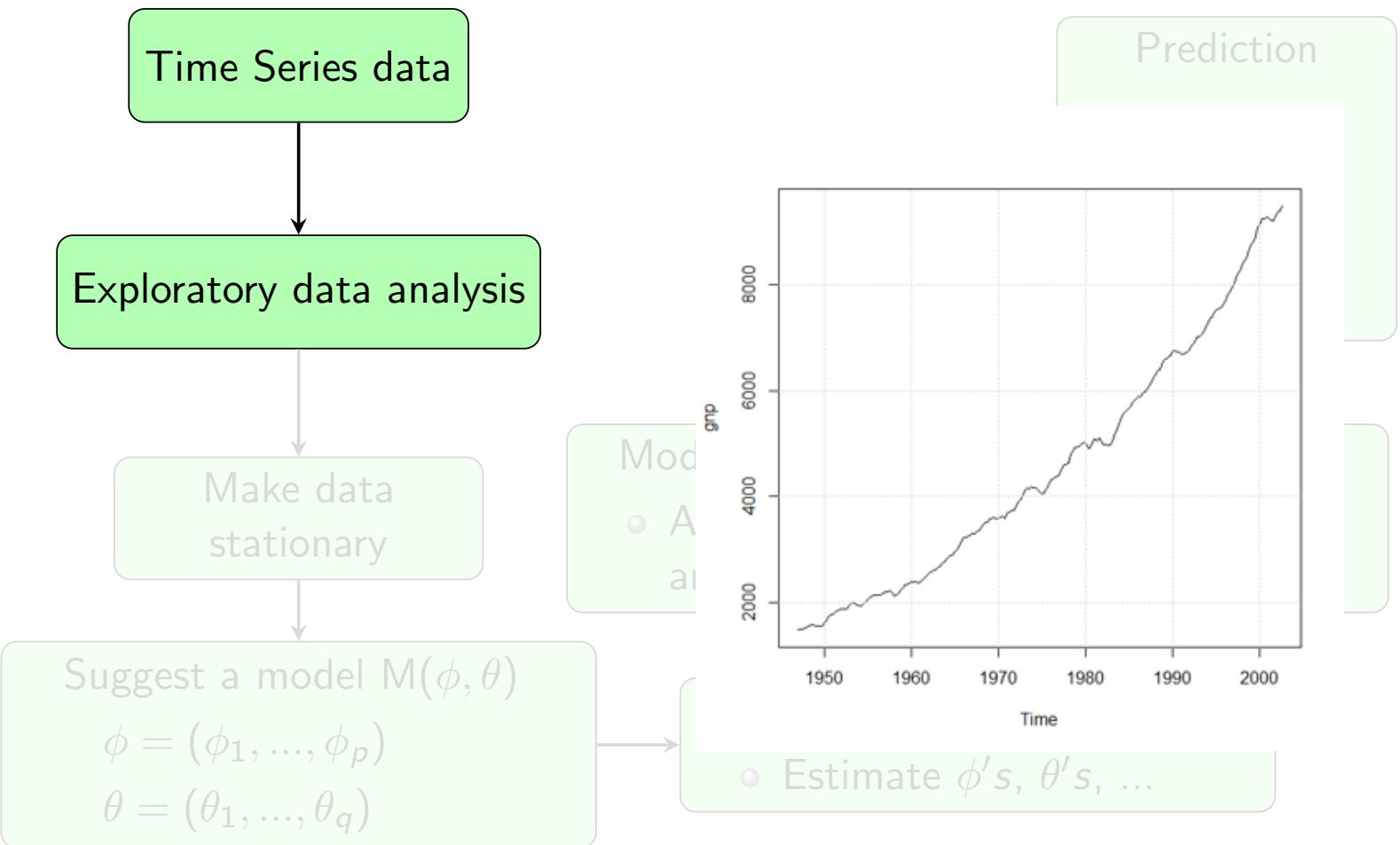
**Tohid Ardesthiri**

Linköping University  
Division of Statistics and Machine Learning

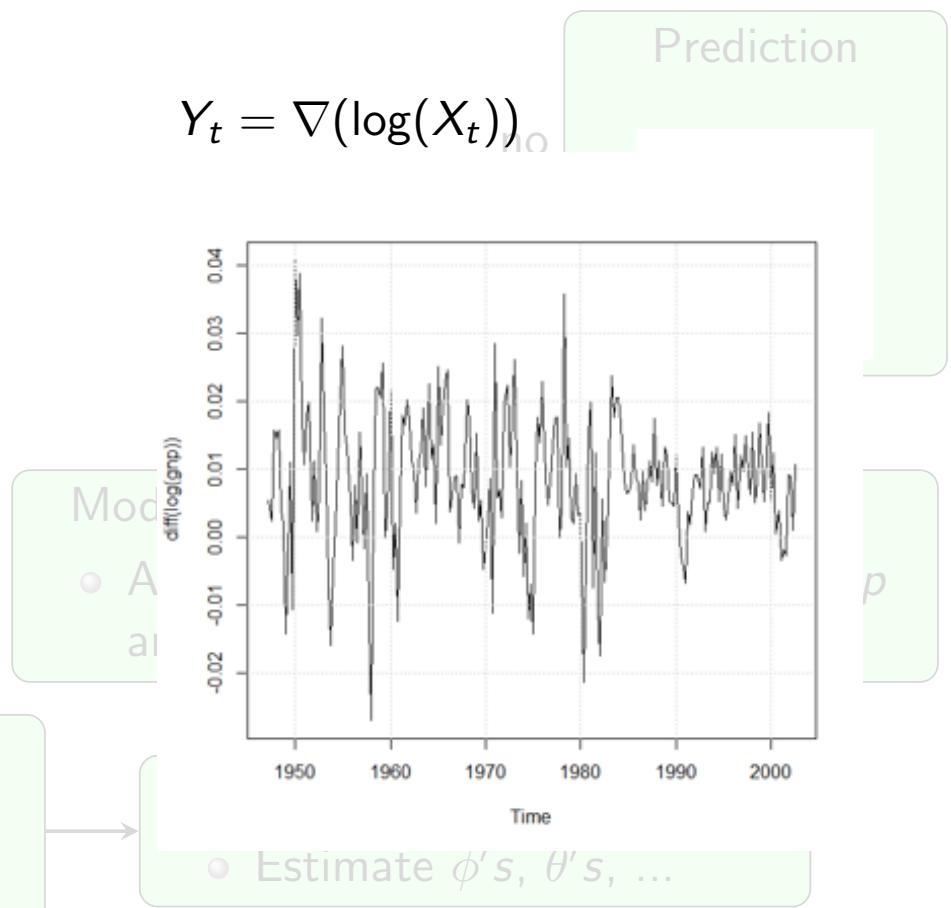
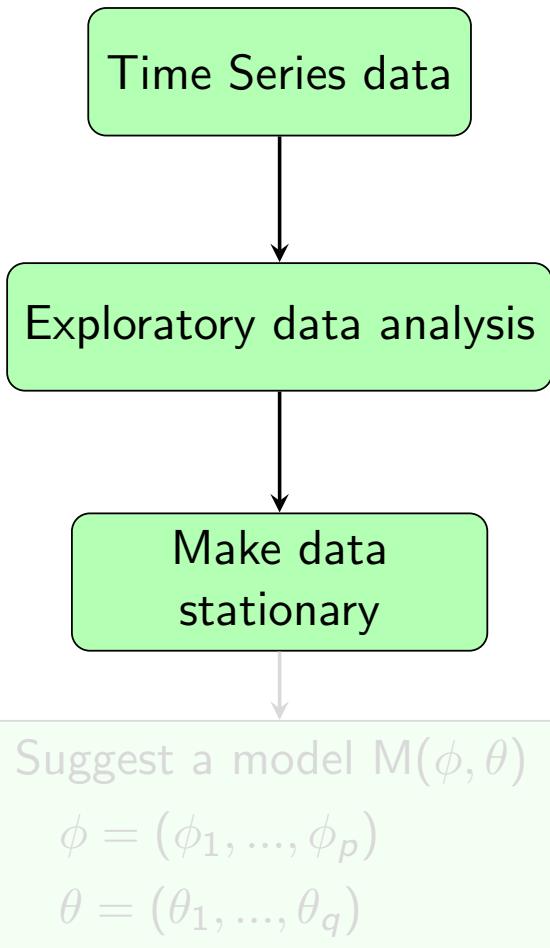
September 27, 2019



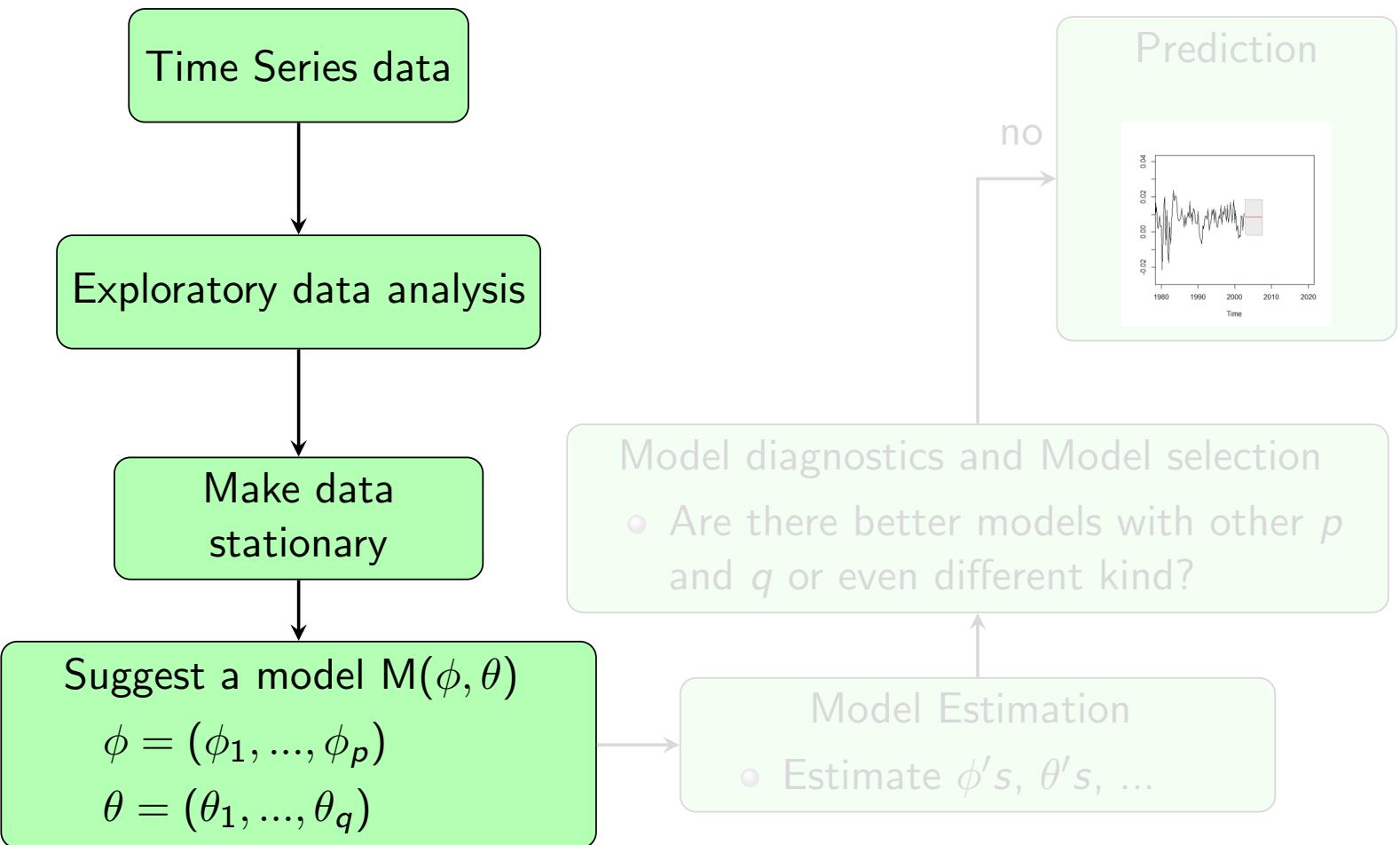
# Time domain: The Big Picture



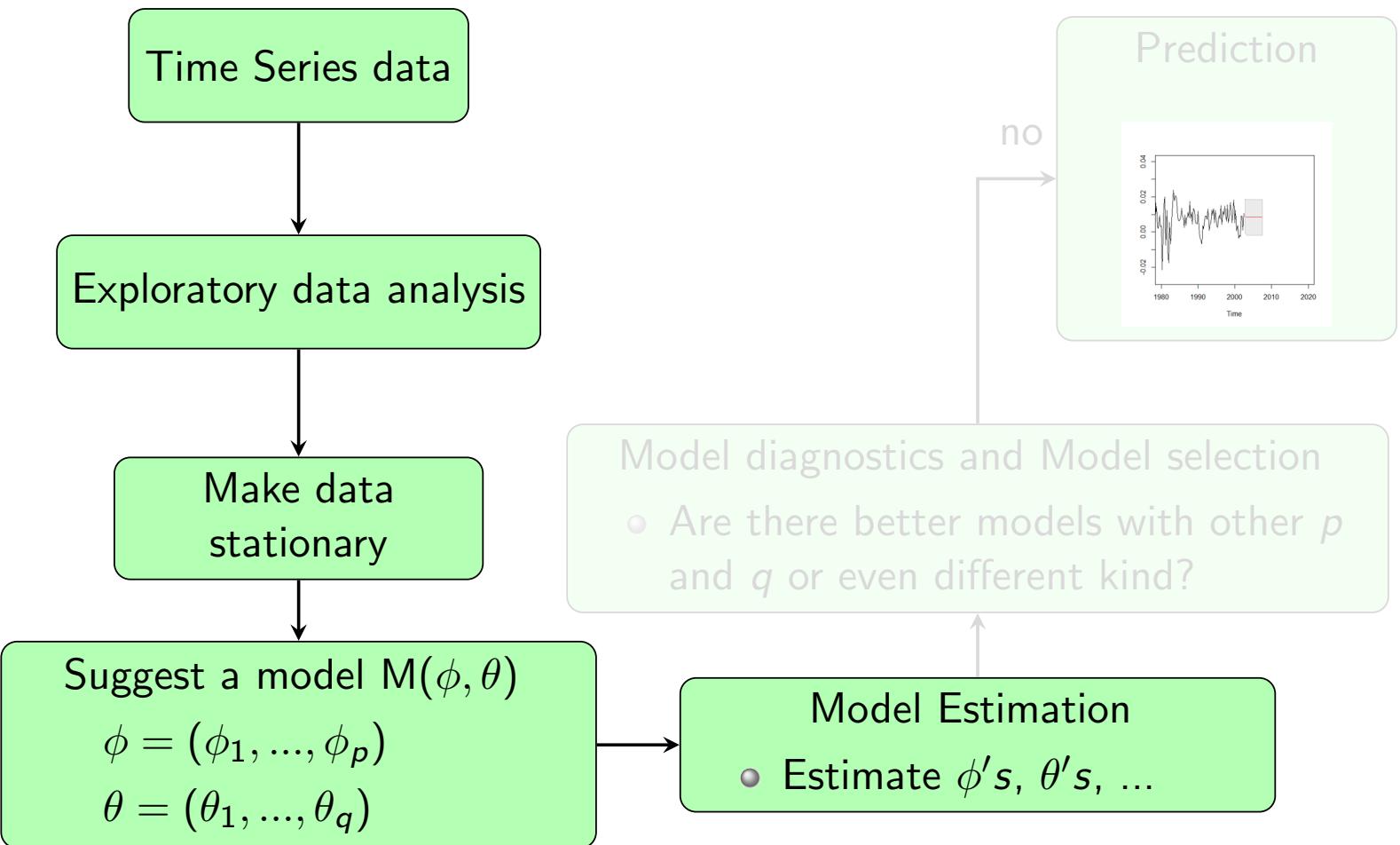
# Time domain: The Big Picture



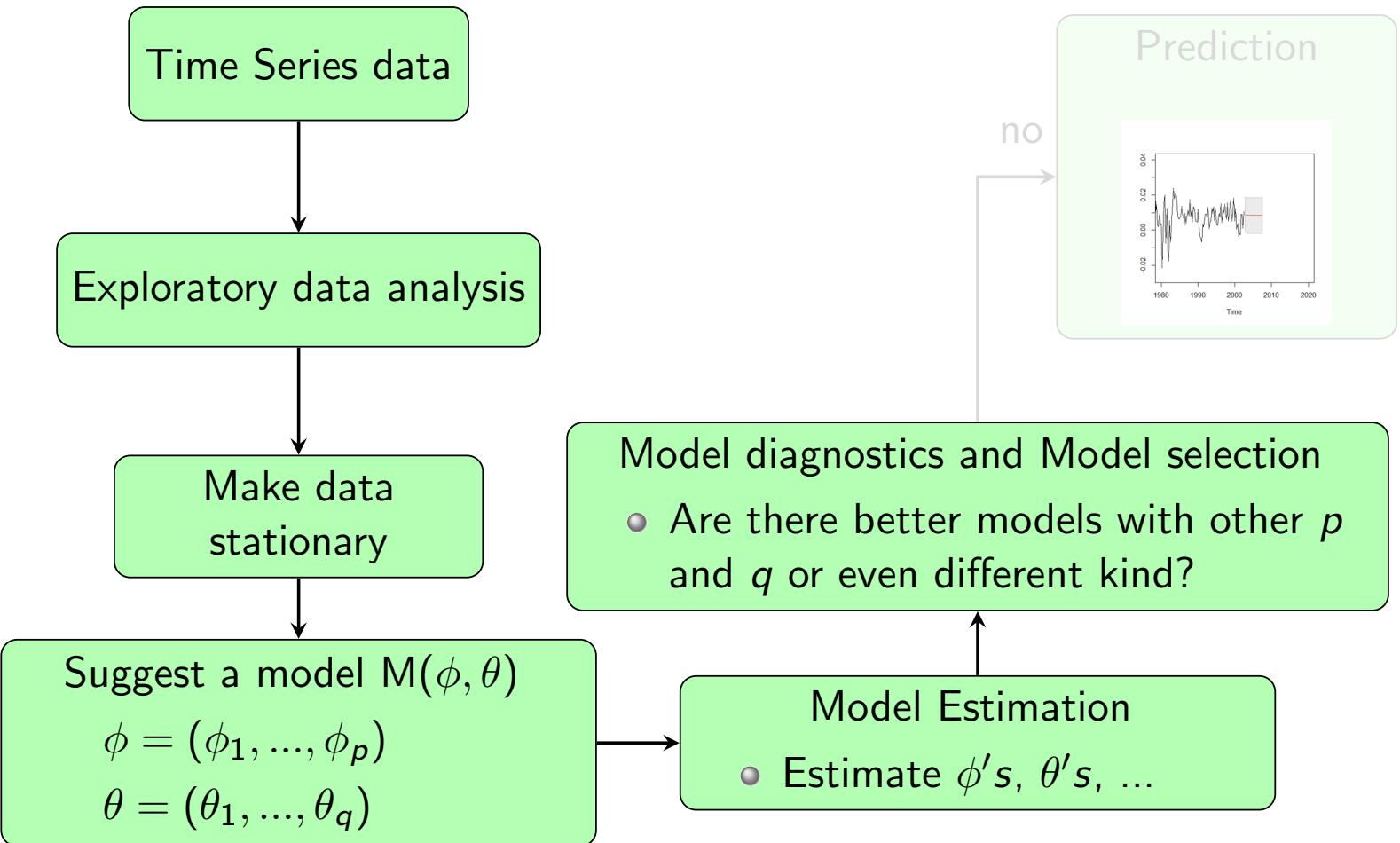
# Time domain: The Big Picture



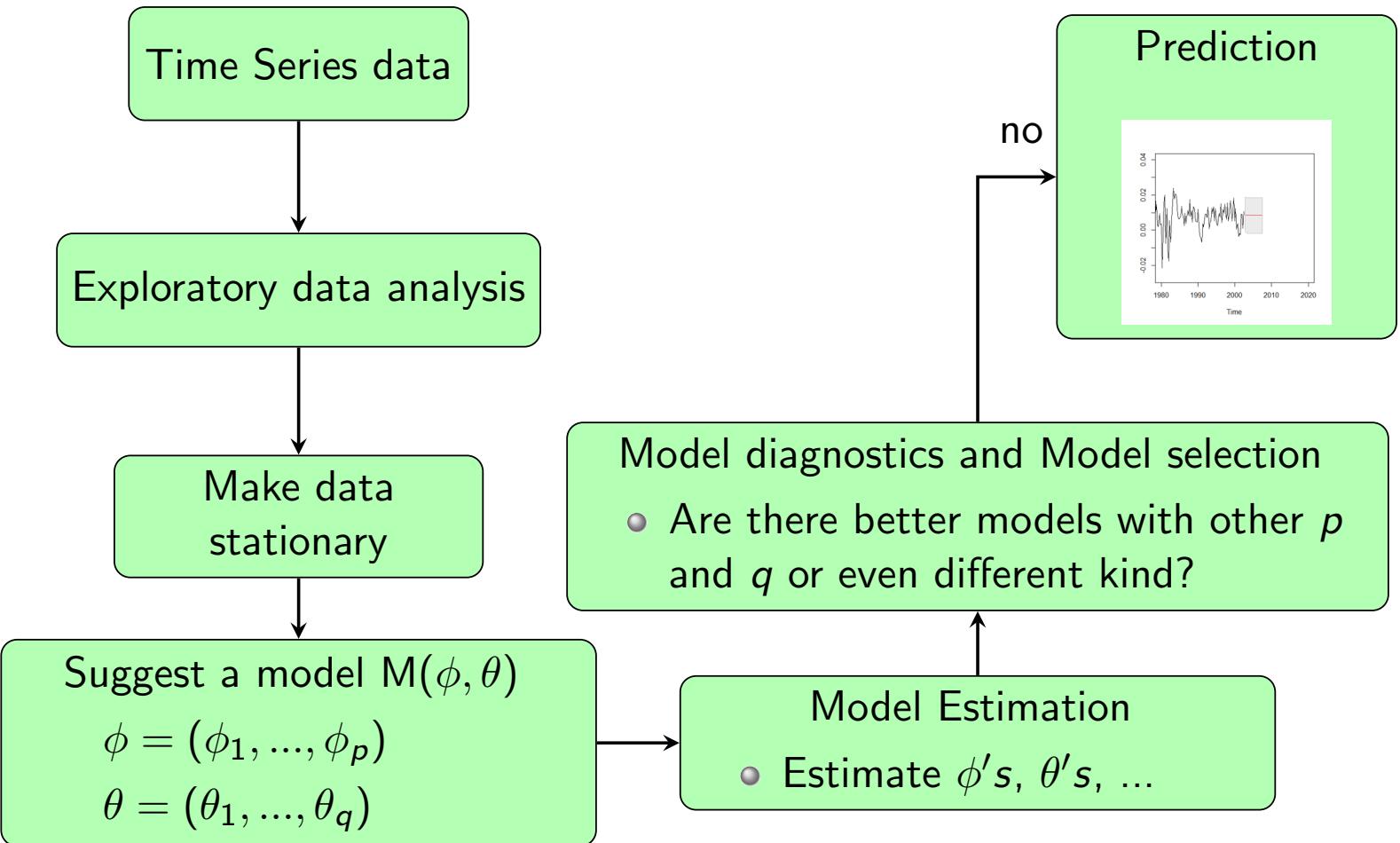
# Time domain: The Big Picture



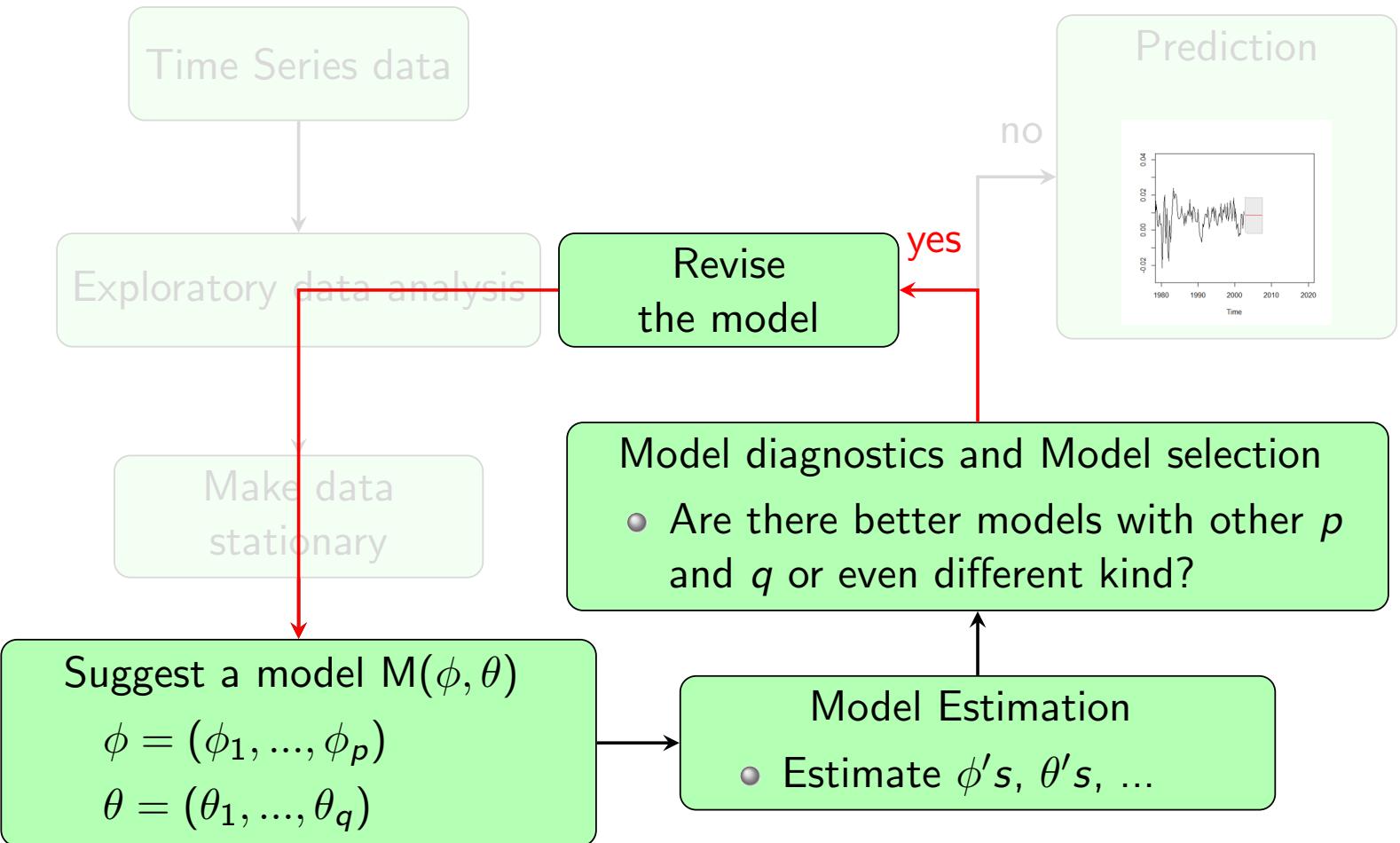
# Time domain: The Big Picture



# Time domain: The Big Picture



# Time domain: The Big Picture



# Model selection

Fit the tentative models, compare them

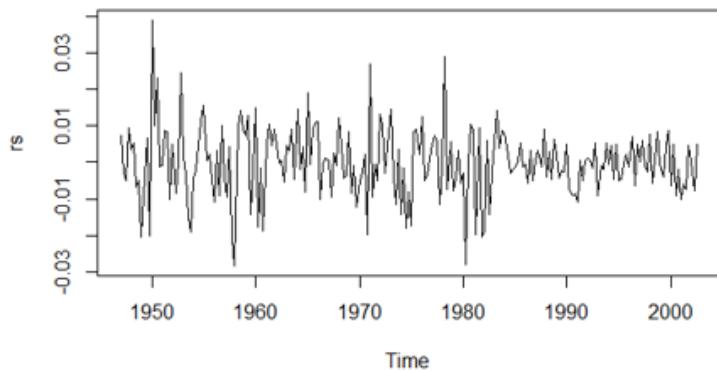
- Analytical measures: AIC, BIC
  - ▶ Penalize models with many parameters → simpler models
- Residual analysis

# Residual analysis

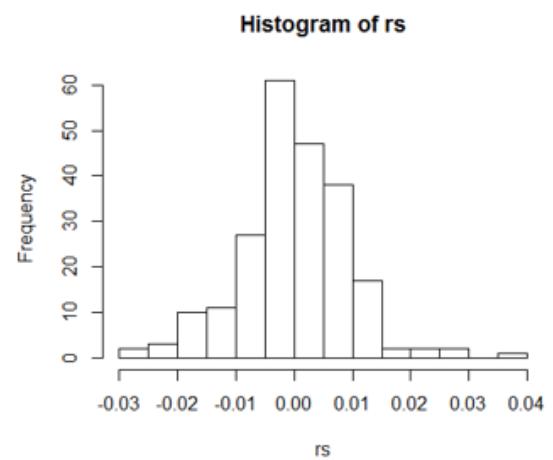
- Residuals  $r_t = x_t - \hat{x}_t^{t-1}$ ? they are innovations
  - ▶ Note: computed from one-step-ahead predictions!
  - ▶ Measures predictive quality of the model (compare OLS)
- Residual analysis
  - ▶ Visual inspection: stationary? Patterns?
  - ▶ Histograms, Q-Q plots
  - ▶ ACF, PACF
  - ▶ Runs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

# Residual analysis - Visual inspection

## Histogram and visual inspection

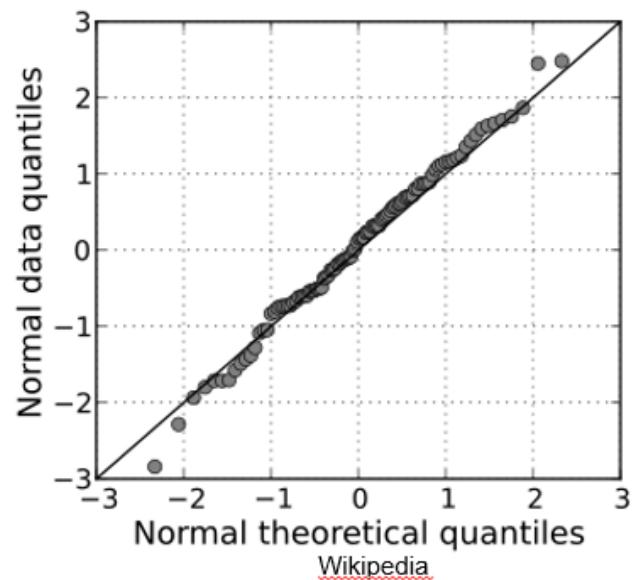
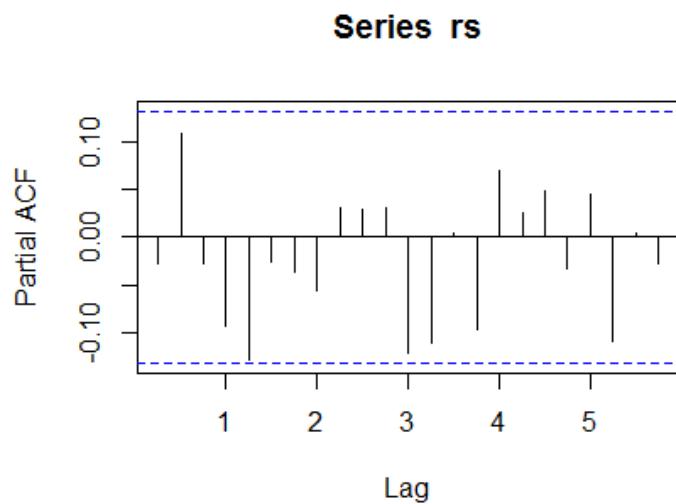


If looks white is good



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

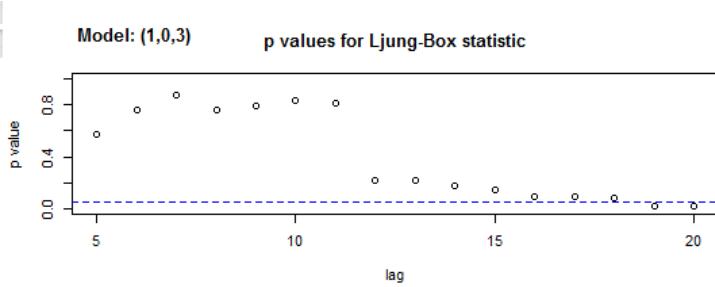
Tests are used to test independence

## Runs test

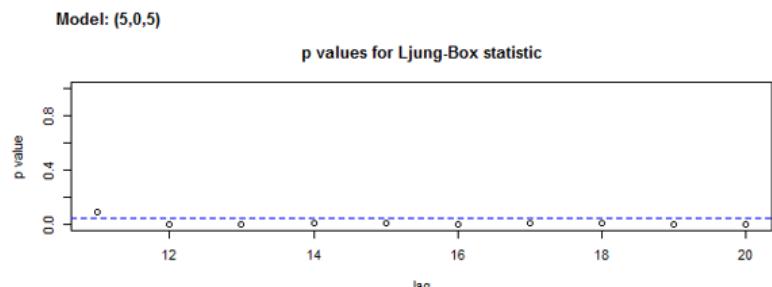
- $H_0$  :  $x_t$  values are i.i.d. **p-value NOT small**
- $H_a$  :  $x_t$  values are not i.i.d. **p-value small**

## Box-Ljung test

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



GOOD



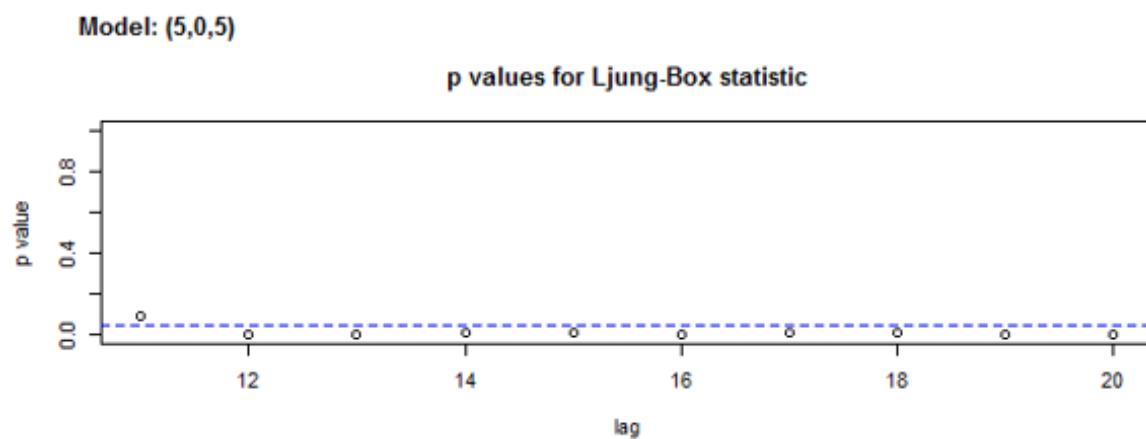
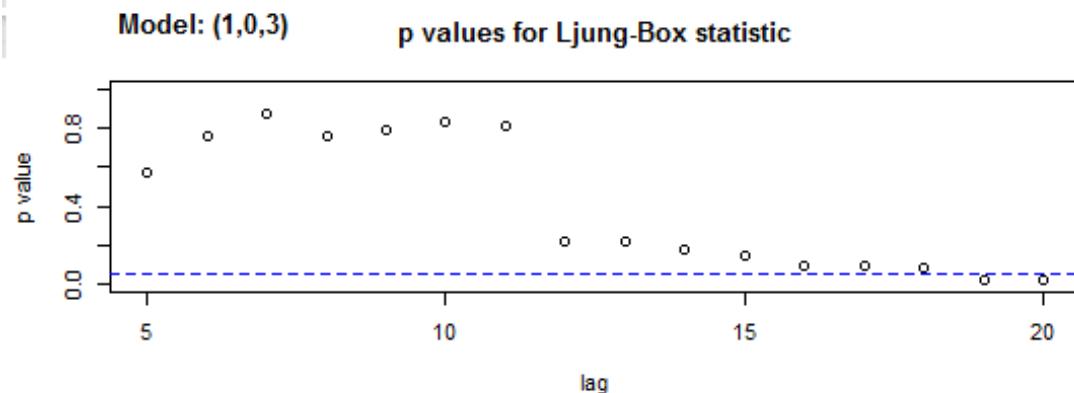
BAD

# 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

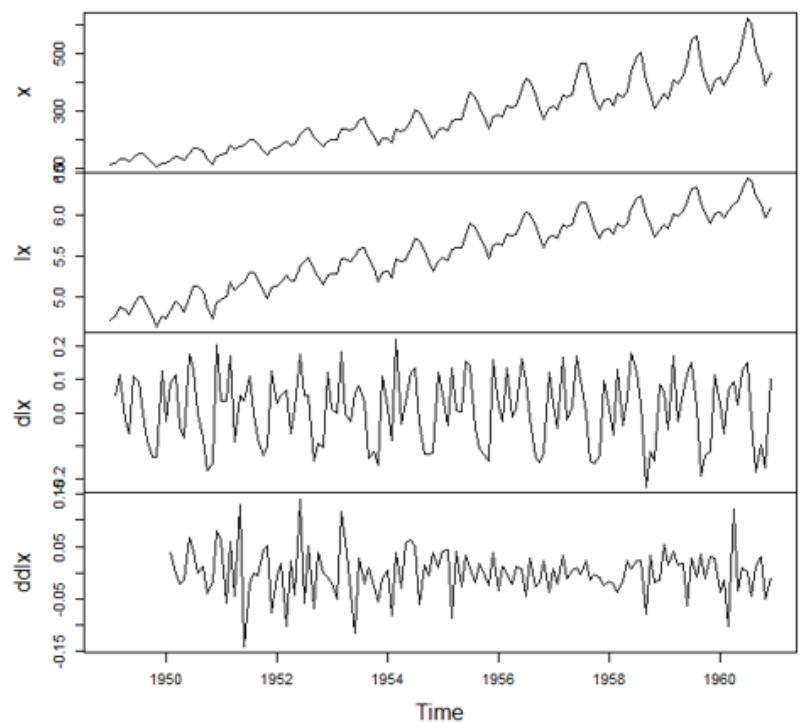
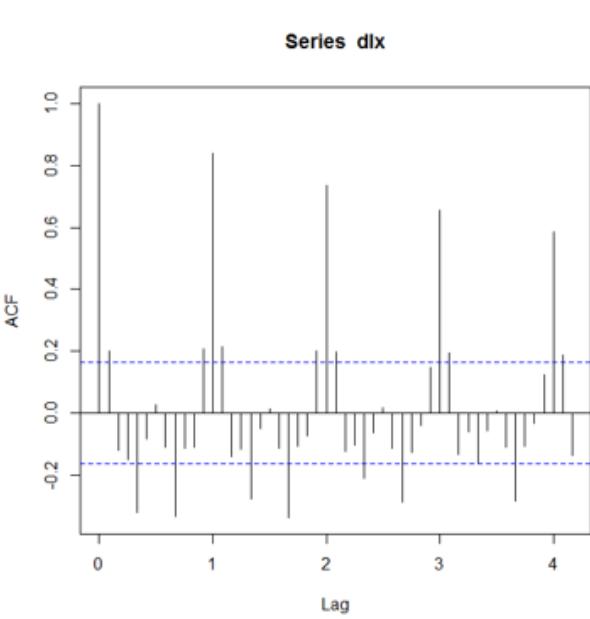
# Overfitting

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



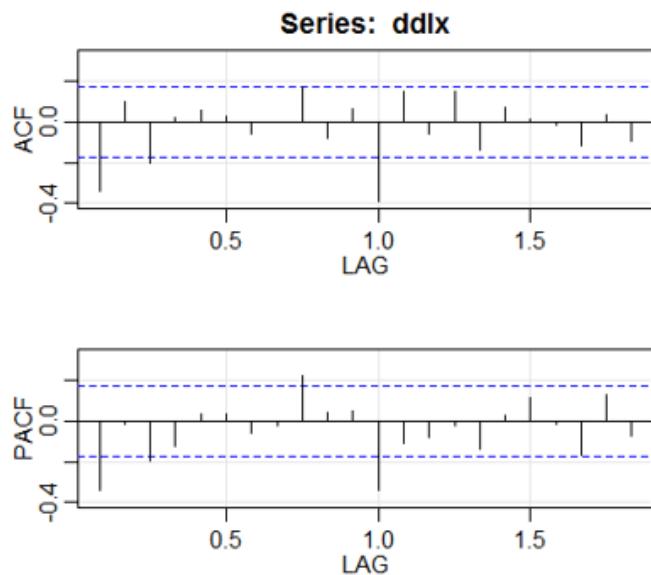
# SARIMA - Air passengers

- Example: Air passengers



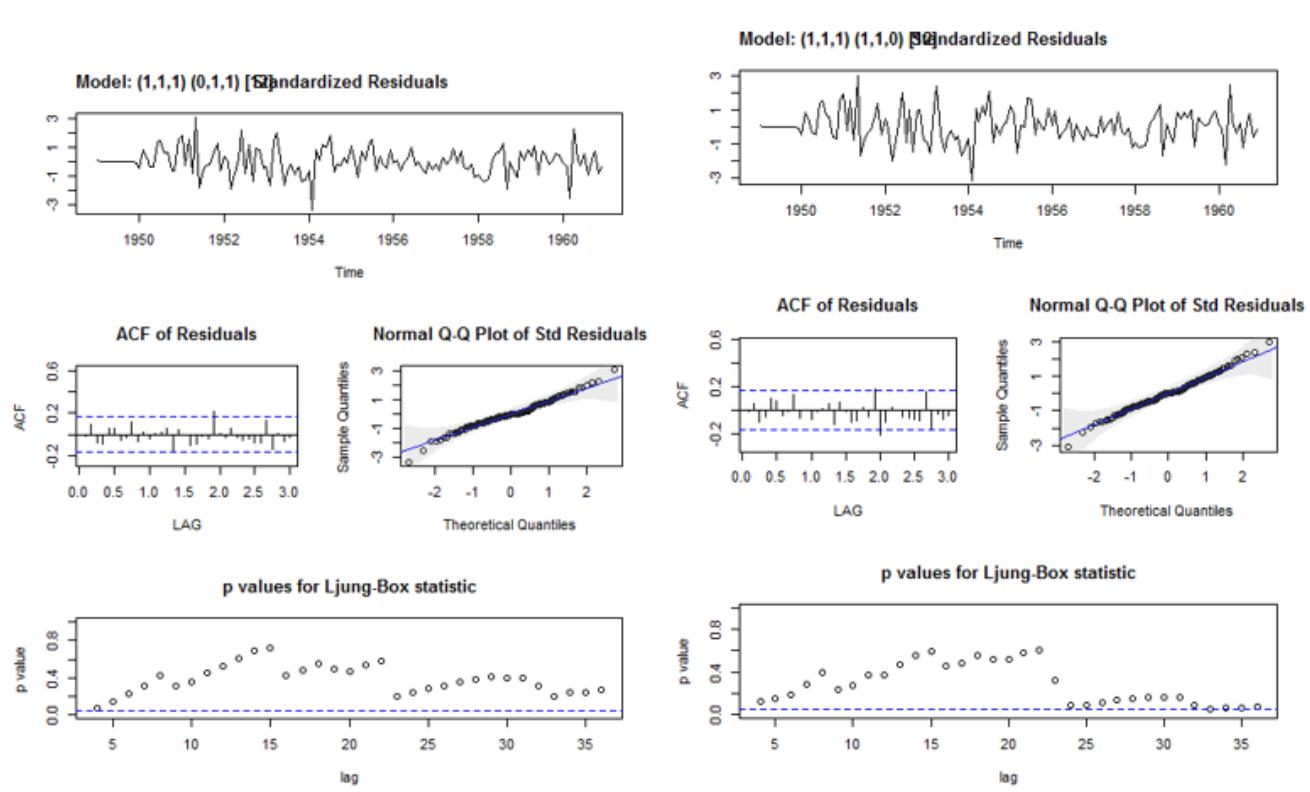
# SARIMA - Air passengers

- Example: Air passengers



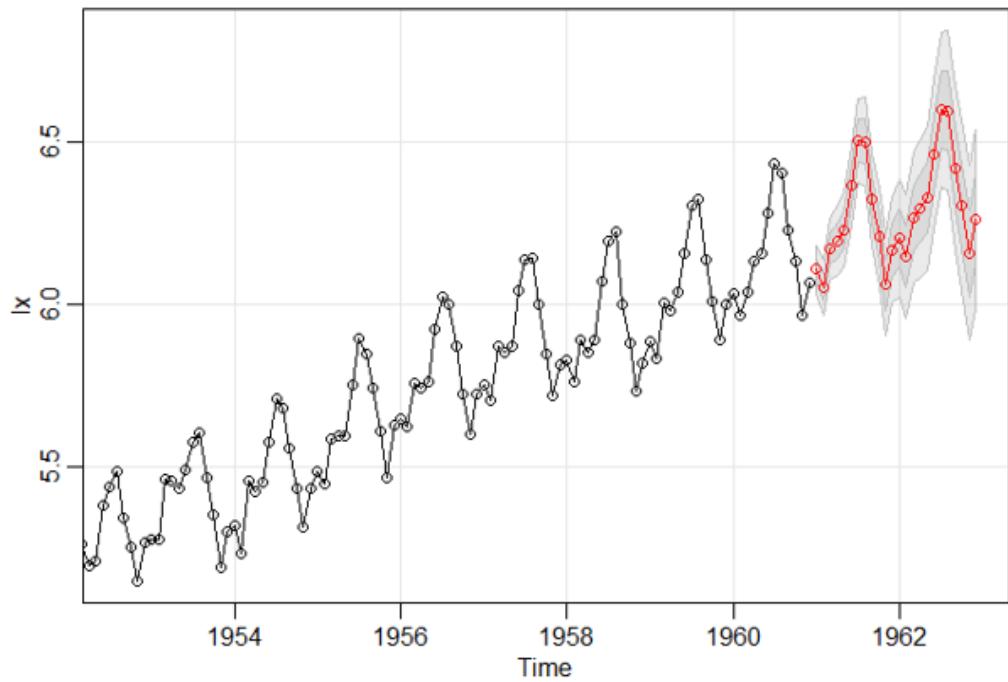
ARIMA(0, 1, 1)<sub>12</sub> or  
ARIMA(1, 1, 0)<sub>12</sub>

# SARIMA - Air passengers



# 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.

## Vectorized AR(2) model

## Whiteboard

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:

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

# ARIMA models in State Space form

Whiteboard

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

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

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

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

# ARIMA models in State Space form

**Whiteboard**

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

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

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

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

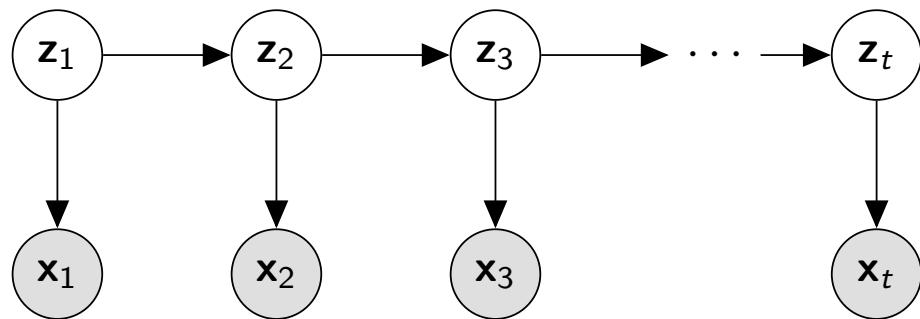
$$x_t = [1 \ \theta_1 \ \theta_2 \ \cdots \ \theta_r] z_t$$

# State Space models - graphical models

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

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

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

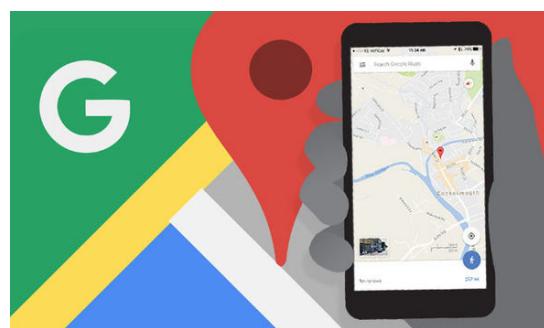
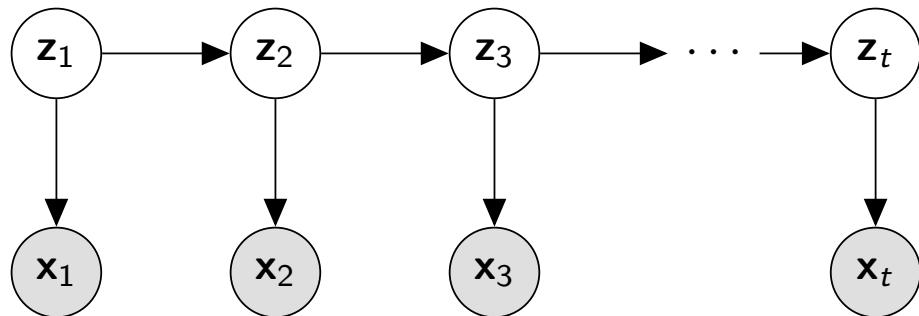


The main tool here is the probability Calculus; Bayes rule and marginalization.

# Dynamical systems - more general case

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

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

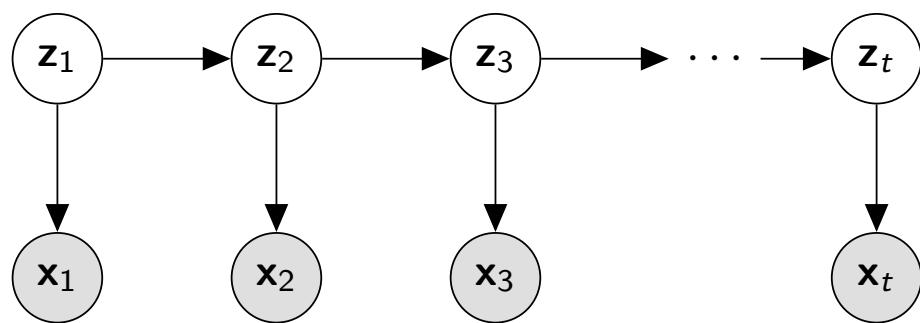


# State Space models - Linear and Gaussian

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.

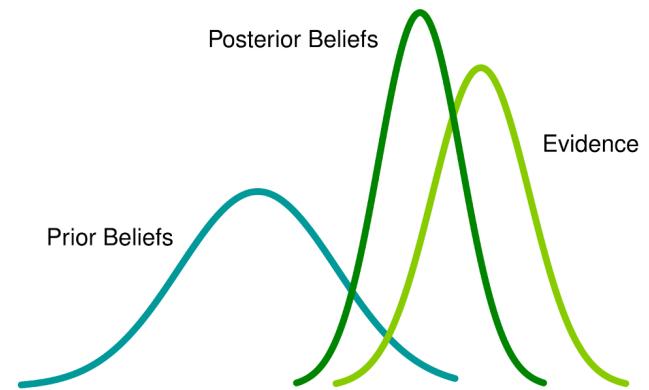
## Example: likelihood update

$$f(\mathbf{z}|\mathbf{x}) \propto f(\mathbf{x}|\mathbf{z})f(\mathbf{z})$$

## Probability Calculus

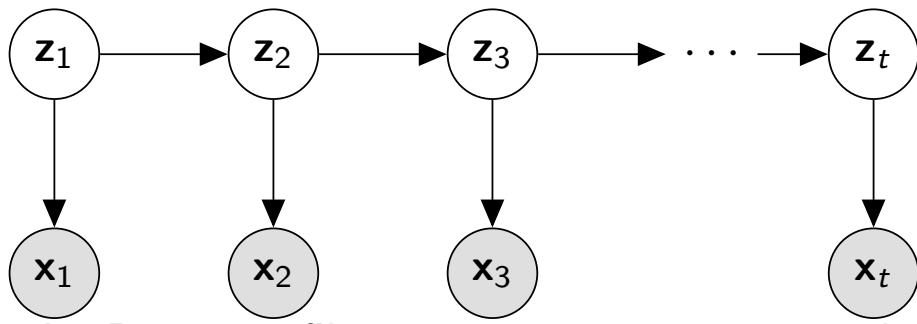
$$f(\mathbf{z}, \mathbf{x}) = f(\mathbf{z}|\mathbf{x})f(\mathbf{x})$$

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



# 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  $\mu$  and covariance  $\Sigma$ , 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

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

## Lecture 7

# Time Series Analysis

Lecture 7: State Space Models, Kalman filtering  
Kalman Smoothing

**Tohid Ardestiri**

Linköping University  
Division of Statistics and Machine Learning

September 30, 2019



# Remaining Course topics

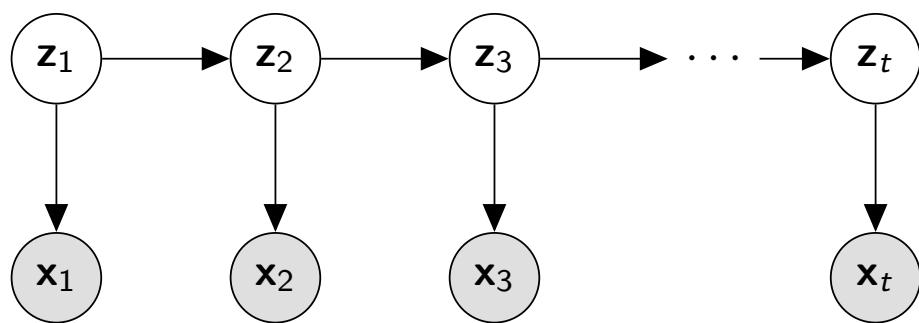
- ARIMA models
- State space models (2 lectures, 1 teaching session with hand-in, 1 computer lab with short report)
  - ▶ Linear and Gaussian state space models (Chapter 6.1)
  - ▶ Kalman filtering, Kalman smoothing and Forecasting (Chapter 6.2)
  - ▶ Maximum likelihood estimate of the state space models (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

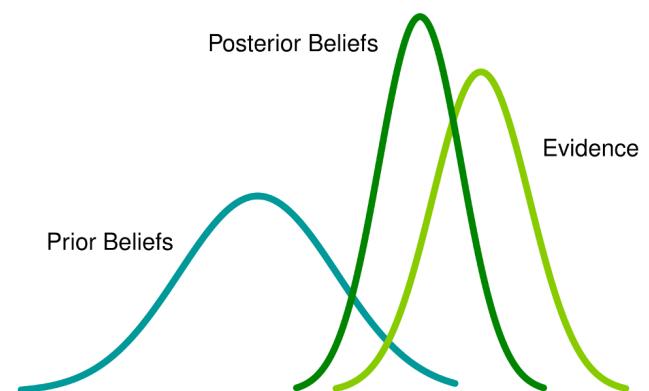
$$f(z|x) \propto f(x|z)f(z)$$

## Probability Calculus

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

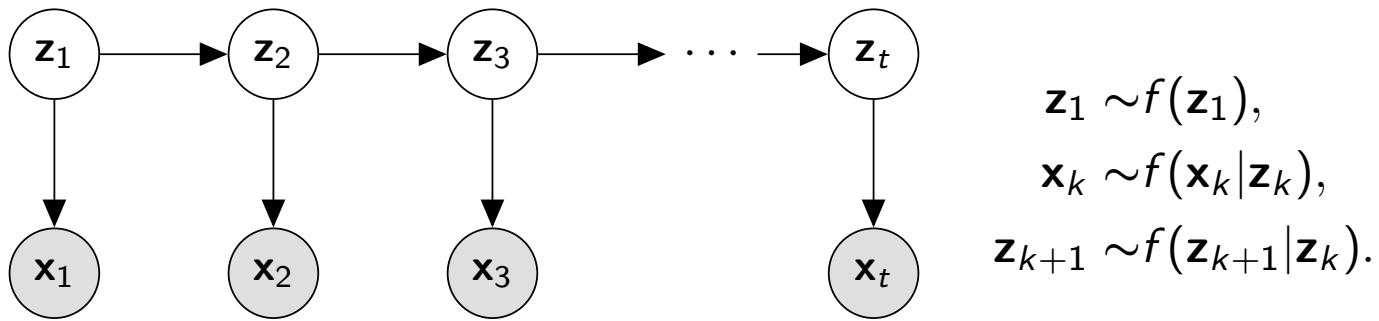
$$f(z, x) = f(x|z)f(z)$$

$$f(z) = \int f(z, x) dx$$



# Online recursive algorithms

Consider a stochastic dynamical system represented by the following recursion



The Bayesian filtering recursion corresponds to computing the posterior distributions  $f(\text{z}_k | \text{x}_{1:k})$ :

$$f(\text{z}_k | \text{x}_{1:k}) = \frac{f(\text{z}_k | \text{x}_{1:k-1}) f(\text{x}_k | \text{z}_k)}{\int f(\text{z}_k | \text{x}_{1:k-1}) f(\text{x}_k | \text{z}_k) d\text{z}_k}$$

The density  $f(\text{z}_k | \text{x}_{1:k-1})$  in the numerator which is called the predicted density of  $\text{z}_k$  and is obtained by integration as in

$$f(\text{z}_k | \text{x}_{1:k-1}) = \int f(\text{z}_k | \text{z}_{k-1}) f(\text{z}_{k-1} | \text{x}_{1:k-1}) d\text{z}_{k-1}.$$

# 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.

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

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



# The Kalman Filter's Foundation

Let  $\mathbf{z}$  have a normal prior distribution with mean  $\mu$  and covariance  $\Sigma$ , 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:**  $f(\mathbf{y}_1)f(\mathbf{y}_2|\mathbf{y}_1) = f(\mathbf{y}_1, \mathbf{y}_2)$

$$N(\mathbf{y}_1; \mu, \Sigma)N(\mathbf{y}_2; C\mathbf{y}_1, R) = N\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}; \begin{bmatrix} \mu \\ C\mu \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma C^T \\ C\Sigma & C\Sigma C^T + R \end{bmatrix}\right)$$

**Property 2: marginalization and conditioning**

If  $\mathbf{y}_1, \mathbf{y}_2$  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

$$\mu_t = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{t|t} \\ Am_{t|t} \\ CAm_{t|t} \end{bmatrix}$$

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  $f(\mathbf{z}_1) = N(\mathbf{z}_1; m_0, P_0)$

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

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

---

1: **Inputs:**  $A_t$ ,  $C_t$ ,  $Q_t$ ,  $R_t$ ,  $m_0$ ,  $P_0$  and  $\mathbf{x}_{1:T}$ .

*initialization*

2:  $m_{1|0} \leftarrow m_0$ ,  $P_{1|0} \leftarrow P_0$

3: **for**  $t = 1$  to  $T$  **do**

*observation update step*

4:  $K_t \leftarrow P_{t|t-1}C_t^T(C_tP_{t|t-1}C_t^T + R_t)^{-1}$

5:  $m_{t|t} \leftarrow m_{t|t-1} + K_t(\mathbf{x}_t - C_t m_{t|t-1})$

6:  $P_{t|t} \leftarrow (I - K_t C_t)P_{t|t-1}$

*prediction step*

7:  $m_{t+1|t} \leftarrow A_t m_{t|t}$

8:  $P_{t+1|t} \leftarrow A_t P_{t|t} A_t^T + Q_{t+1}$

9: **end for**

10: **Outputs:**  $m_{t|t}$ ,  $P_{t|t}$  for  $t = 1 : T$

# Bayesian Smoothing

The purpose of Bayesian smoothing is to compute the marginal posterior distribution of  $x_t$  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**

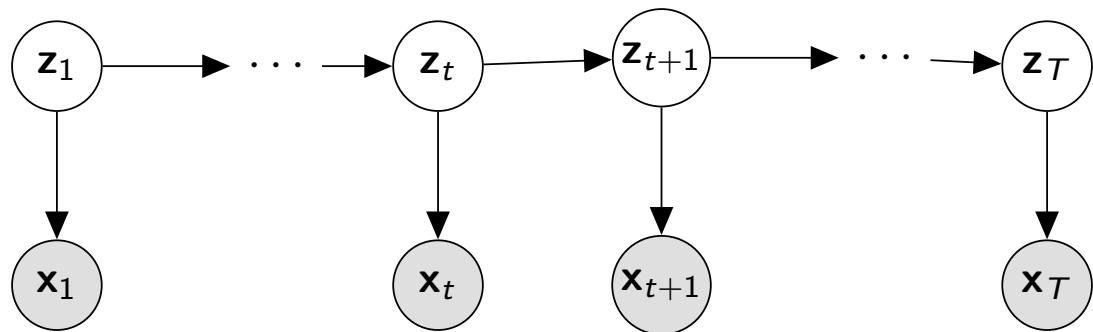
$$\mathbf{z}_t = A_{t-1} \mathbf{z}_{t-1} + e_t$$

## 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:

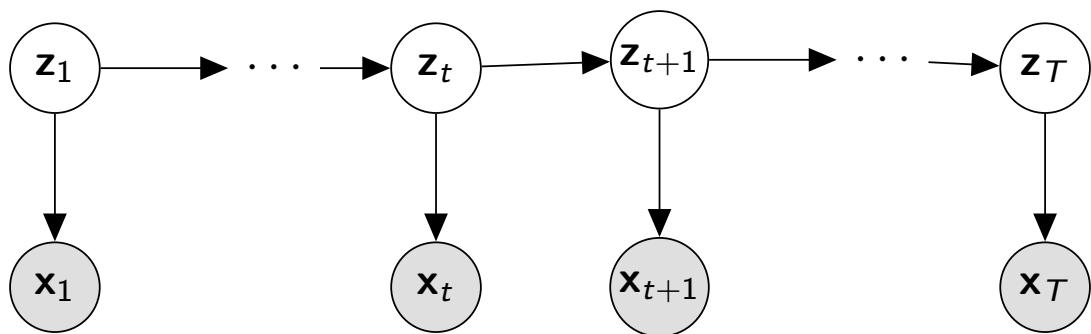
$$f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t}) = N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t)$$

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

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

Recall that

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

## RTS Smoother's derivation

where

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

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

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

Hence,

$$\begin{aligned} f(\mathbf{z}_{t+1}, \mathbf{z}_t | \mathbf{x}_{1:T}) &= N(\mathbf{z}_{t+1}; m_{t+1|T}, P_{t+1|T}) N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t) \\ &= N\left(\left[\begin{array}{c} \mathbf{z}_t \\ \mathbf{z}_{t+1} \end{array}\right], \left[\begin{array}{c} \cdot \\ m_{t+1|T} \end{array}\right], \left[\begin{array}{cc} \cdot & \cdot \\ \cdot & P_{t+1|T} \end{array}\right]\right) \end{aligned}$$

# RTS smoother's backwards recursion

Prove the backwards recursion of the RTS smoother for following state space model with initial prior on the state  $f(\mathbf{z}_1) = N(\mathbf{z}_1; m_0, P_0)$

$$\begin{aligned}\mathbf{z}_t &= A_{t-1}\mathbf{z}_{t-1} + e_t, & e_t &\sim N(0, Q_t) \\ \mathbf{x}_t &= C_t\mathbf{z}_t + \nu_t, & \nu_t &\sim N(0, R_t)\end{aligned}$$

---

- 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:    $m_{t|\tau} \leftarrow m_{t|t} + G_t(m_{t+1|\tau} - A_t m_{t|t})$
  - 5:    $P_{t|\tau} \leftarrow P_{t|t} + G_t(P_{t+1|\tau} - P_{t+1|t})G_t^T$
  - 6: **end for**
  - 7: **Outputs:**  $m_{t|\tau}, P_{t|\tau}$
-

# Read home

- Shumway and Stoffer, Chapters 6.1 and 6.2

## Lecture 8

# Time Series Analysis

## Lecture 8: State Space Model Stochastic Volatility

**Tohid Ardesthiri**

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

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

- **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

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

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

# Stochastic Volatility

In the following:

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

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

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

$$\begin{aligned}\mathbf{z}_t &= \phi \mathbf{z}_{t-1} + w_t \\ r_t &= \beta \exp(\mathbf{z}_t/2) \epsilon_t\end{aligned}$$

## Stochastic Volatility

## Whiteboard

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

$$\begin{aligned}\mathbf{z}_t &= \phi \mathbf{z}_{t-1} + w_t \\ r_t &= \beta \exp(\mathbf{z}_t/2) \epsilon_t\end{aligned}$$

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  $\alpha$  to the state equation and rewrite it as

$$\begin{aligned}\mathbf{z}_t &= \phi_0 + \phi_1 \mathbf{z}_{t-1} + w_t \\ \mathbf{x}_t &= \mathbf{z}_t + \nu_t\end{aligned}$$

where the  $\phi_0$  is called the leverage effect.

# Stochastic Volatility

The distribution of  $\nu_t$  is not Gaussian because

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

Hence,  $\nu$  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,

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

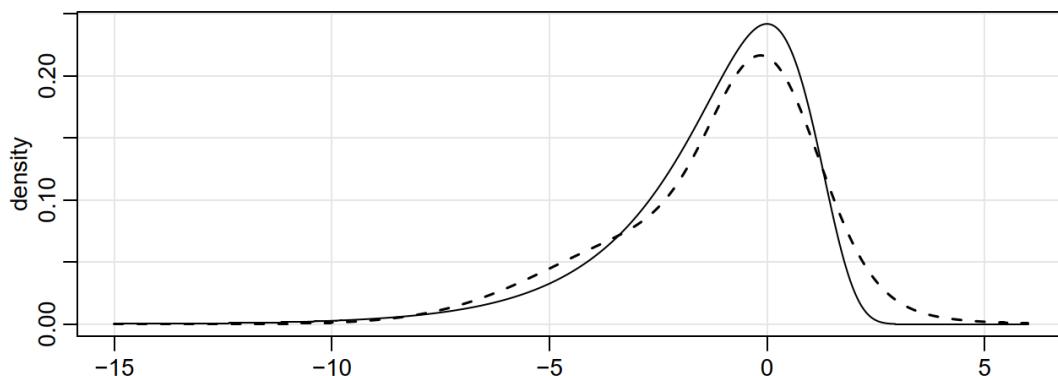
$$n_{t1} \sim N(\mu_1, \sigma_1^2)$$

## 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

$$\begin{aligned}\mathbf{z}_t &= \phi_0 + \phi_1 \mathbf{z}_{t-1} + w_t \\ \mathbf{x}_t &= \mathbf{z}_t + \eta_t\end{aligned}$$

and

$$\begin{aligned}w_t &\sim iidN(0, \sigma_w^2) \\ \eta_t &\sim \pi_0 N(0, \sigma_0^2) + \pi_1 N(\mu_1, \sigma_1^2)\end{aligned}$$

where  $\pi_0 + \pi_1 = 1$

## Gaussian sum filter - derivation

Whiteboard

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

$$\begin{aligned}\mathbf{z}_t &= A\mathbf{z}_{t-1} + w_t \\ \mathbf{x}_t &= C\mathbf{z}_t + \eta_t\end{aligned}$$

and

$$\begin{aligned}w_t &\sim iidN(0, Q) \\ \eta_t &\sim \pi_0 N(\mu_0, R_1) + \pi_1 N(\mu_1, R_2)\end{aligned}$$

where  $\pi_0 + \pi_1 = 1$

# Read home

- Shumway and Stoffer, Chapter 6.11

## Lecture 9



# 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:**

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

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

**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,  $\theta$  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

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

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

That is,

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

## Neural network – graphical illustration

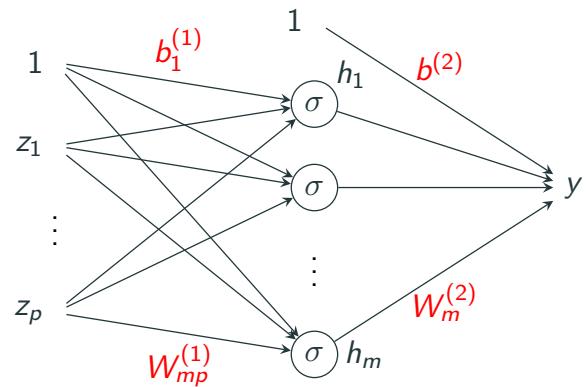
The equations

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

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

can be illustrated graphically.

Input variables	Hidden units	Output
-----------------	--------------	--------

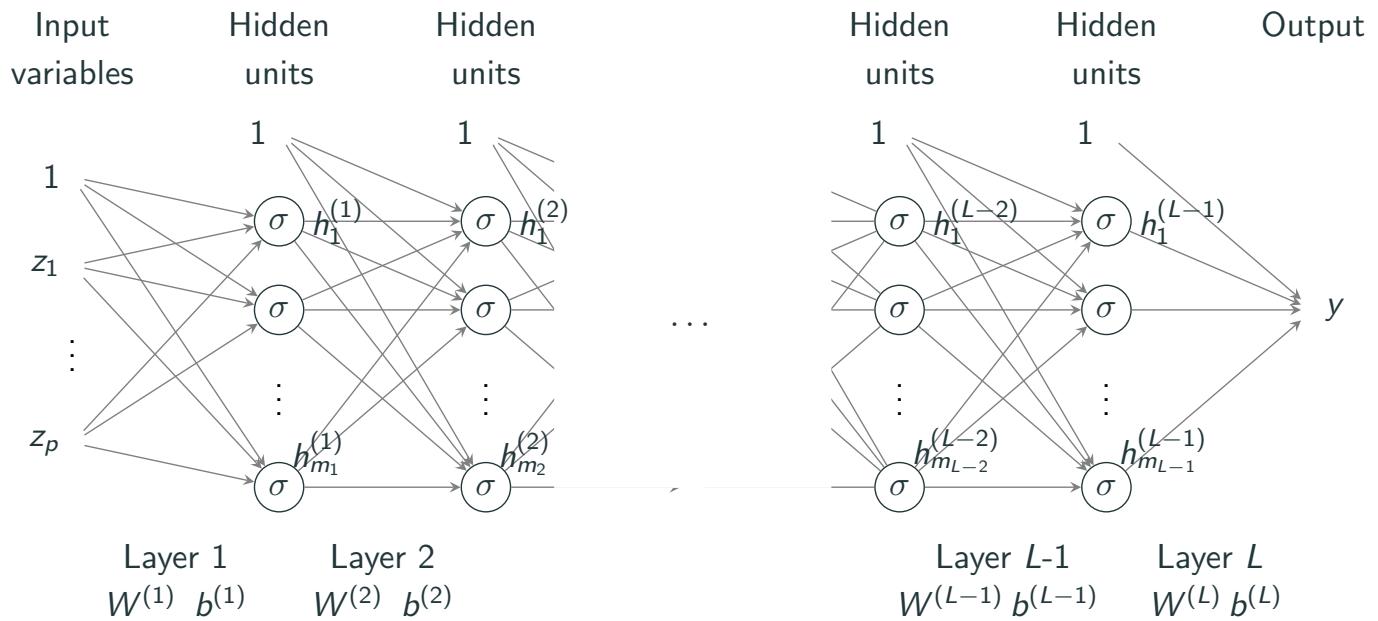


- The variables  $\mathbf{h} = (h_1, \dots, h_m)$  are referred to as a **hidden layer**.
- The function  $\sigma(\cdot)$  is an element-wise nonlinearity, referred to as an **activation function**. Typical choices are

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

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

# 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.)

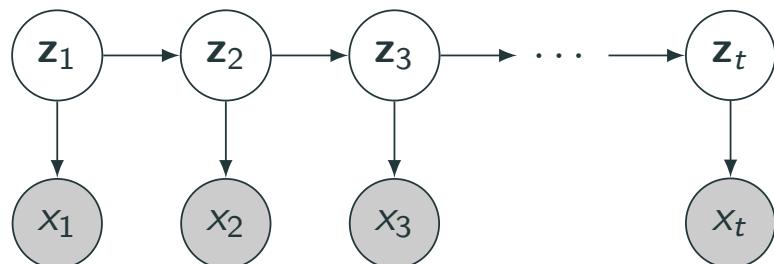
**Proof.** Let  $\mathbf{h}_t = m_{t|t-1}$ , the Kalman predictive mean.

## Innovation form

Original form:

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

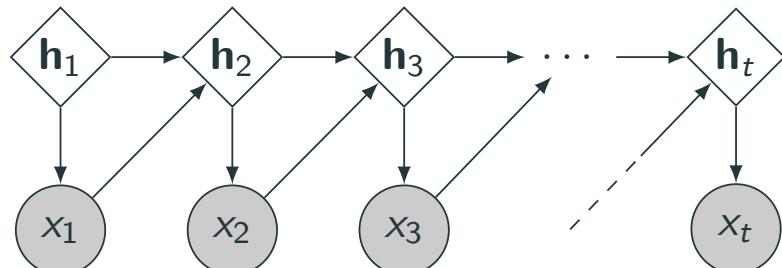
$$x_t = C\mathbf{z}_t + \nu_t.$$



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...?

$$\begin{aligned}\mathbf{h}_t &= W\mathbf{h}_{t-1} + Ux_{t-1}, \\ x_t &= C\mathbf{h}_t + \nu'_t,\end{aligned}$$

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:

$$\begin{aligned}\mathbf{h}_t &= \sigma(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}x_{t-1} + \mathbf{b}), \\ x_t &= \mathbf{C}\mathbf{h}_t + \mathbf{c} + \nu'_t,\end{aligned}$$

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  $\theta$  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  $\sigma$ .

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

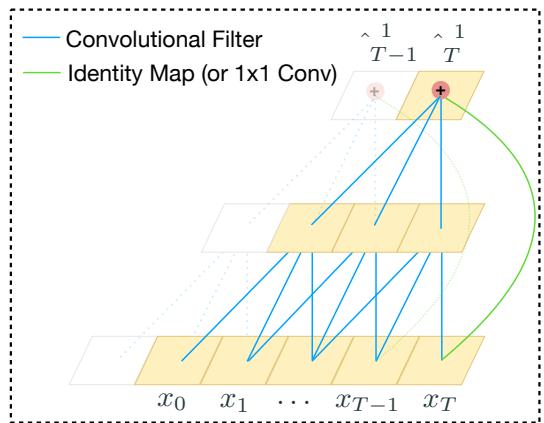
We can address these issues with a **multi-layer network architecture!**

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

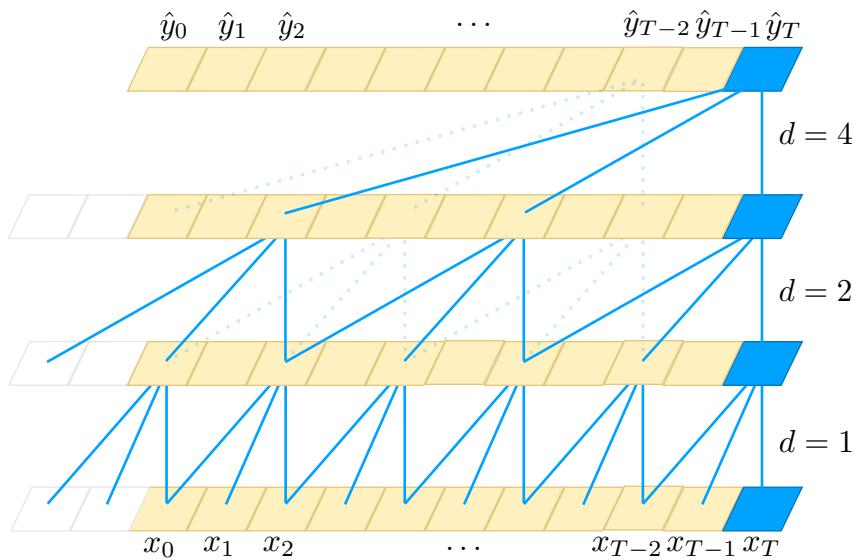


Can extend to multiple hidden layers,  $h_t^{(1)}, h_t^{(2)}, \dots$

- ▲ 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.

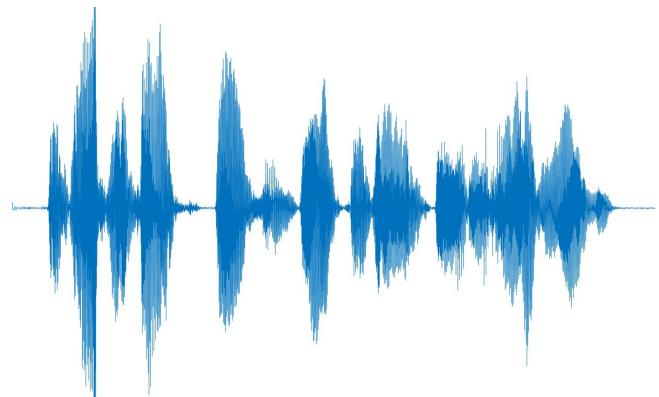
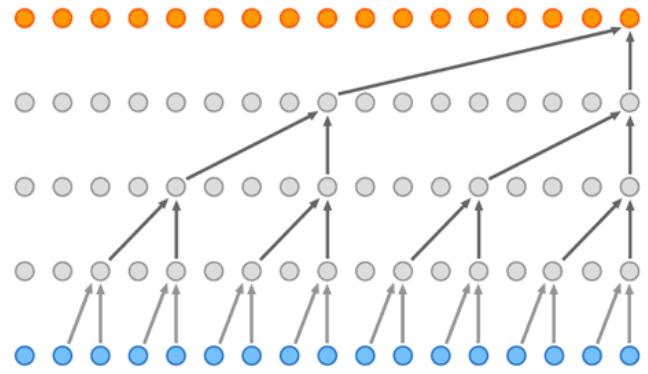
... and have other advantages too: ... but also some disadvantages:

- 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.

## **Teaching Session II**

# 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

$$\Phi_p(B^s) = 1 - \Phi_1(B^{(1\cdot s)}) - \dots - \Phi_p B^{P\cdot s}$$

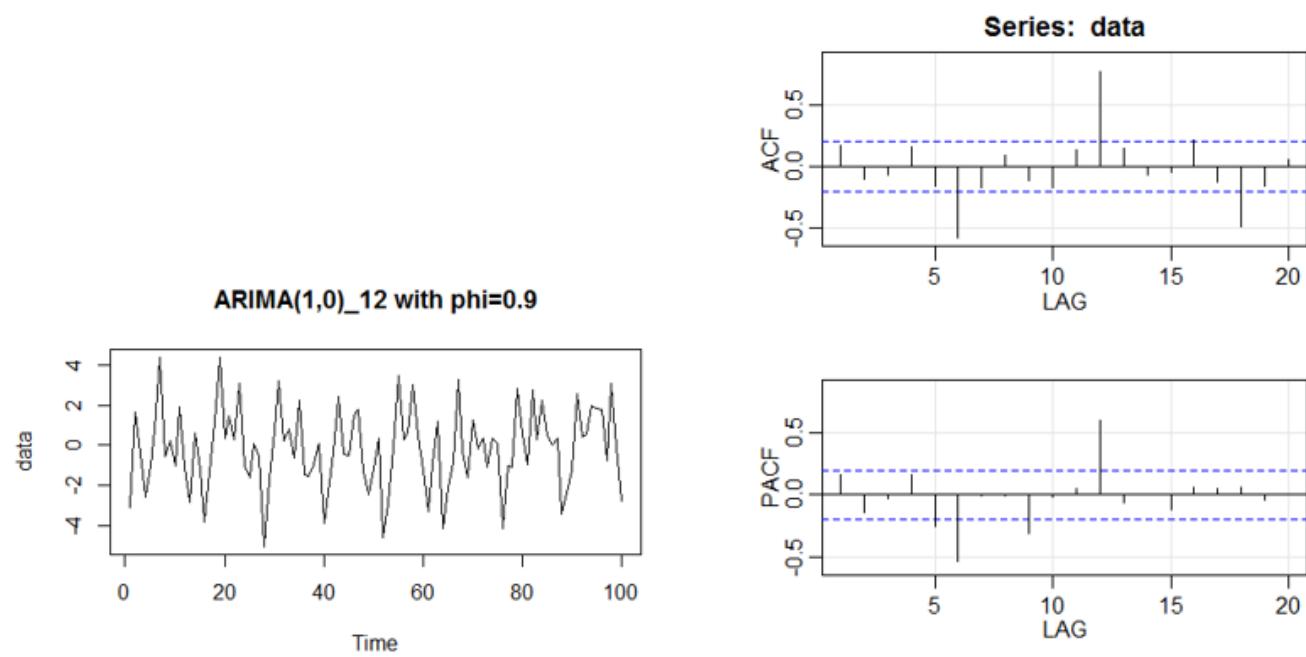
- Seasonal moving average operator

$$\Theta_Q(B^s) = 1 + \Theta_1(B^{(1\cdot s)}) + \dots + \Theta_Q B^{Q\cdot s}$$

- Same principles for causality and invertibility
- Example:  $ARMA(1, 0)_{12}$  and  $ARMA(0, 1)_{12}$ 
  - ▶ Autocovariance

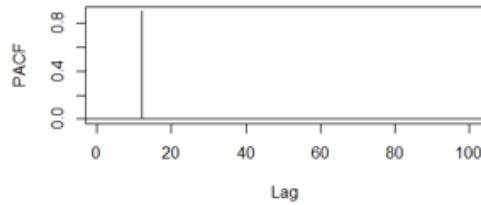
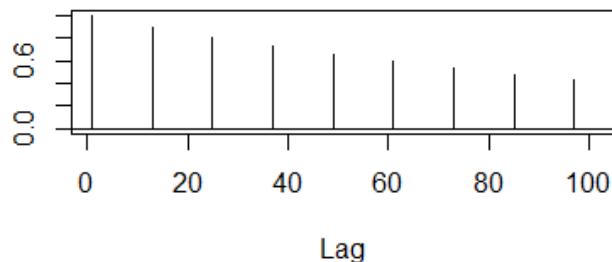
# Seasonal ARMA

- **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 $ks$

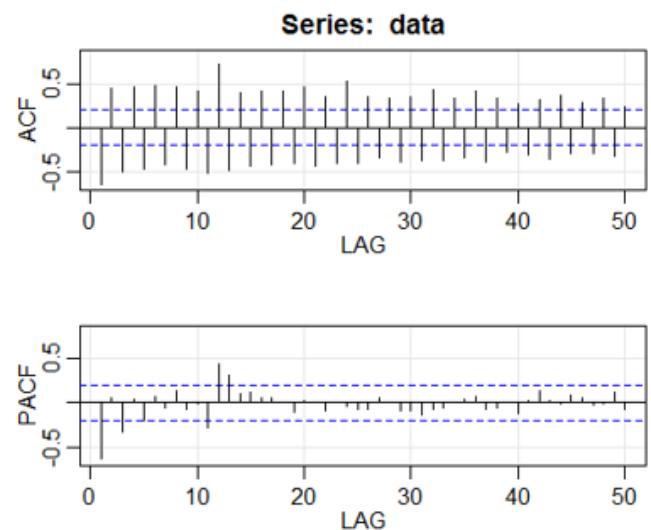
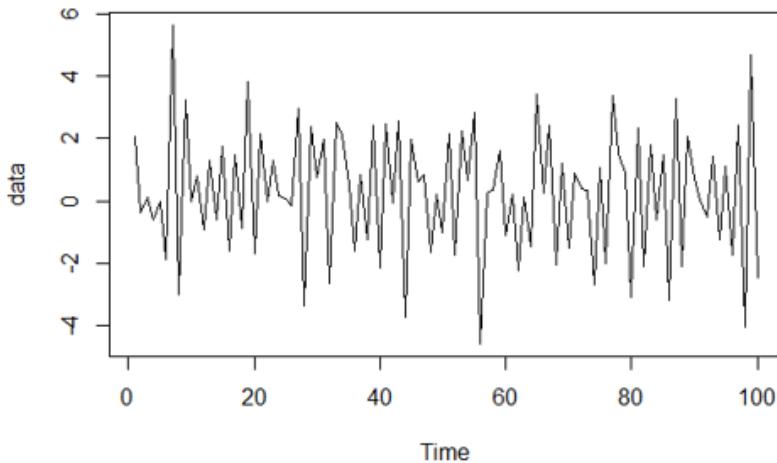
\*The values at nonseasonal lags  $h \neq ks$ , for  $k = 1, 2, \dots$ , are zero.

# Multiplicative seasonal ARMA

- **Problem:** in real data, it is hard to assume  $x_t$  is dependent on  $x_{t-kh}$  only...
  - ▶ Combinational 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( $p, q$ )  $\times (P, Q)_s$

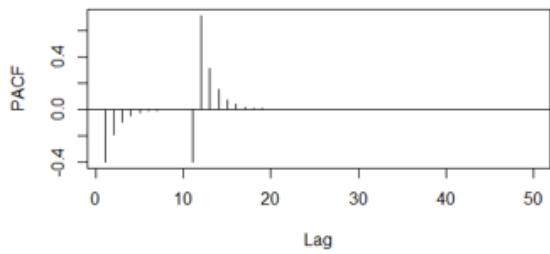
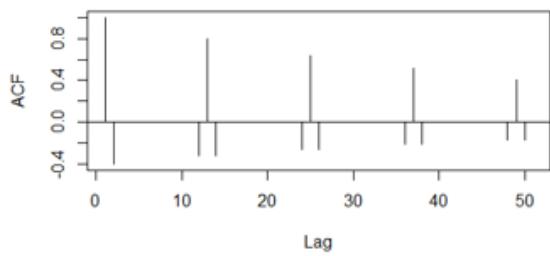
# Multiplicative seasonal ARMA

- Example  $x_t = 0.8x_{t-12} + w_t - 0.5w_{t-1}$



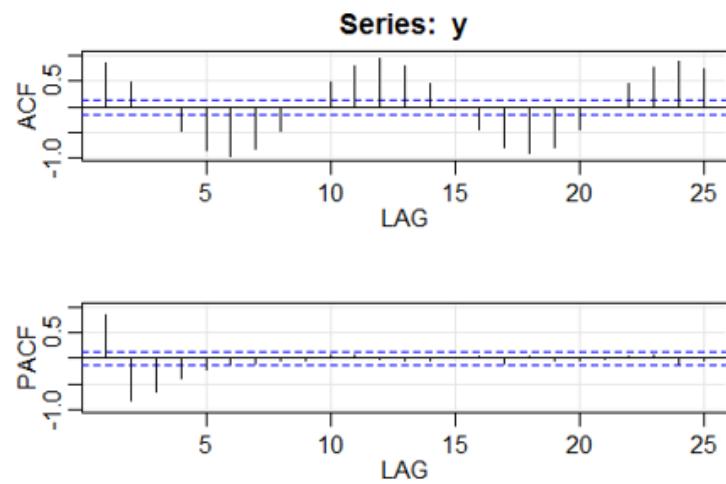
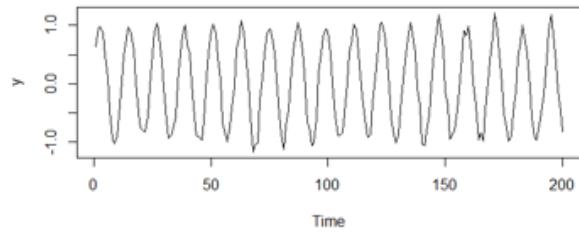
# Multiplicative seasonal ARMA

- 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  $ARIMA(p, d, q) \times (P, D, Q)_s$

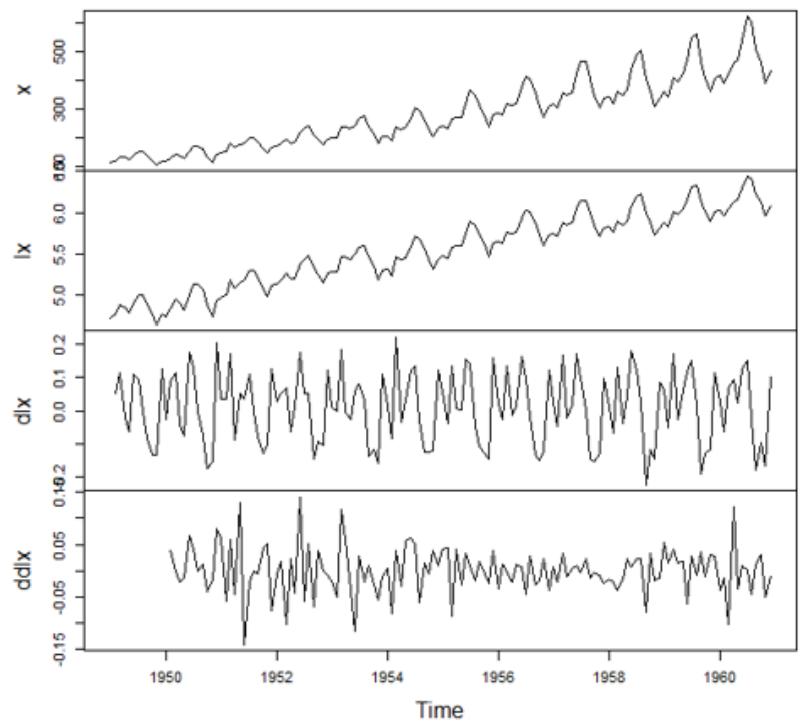
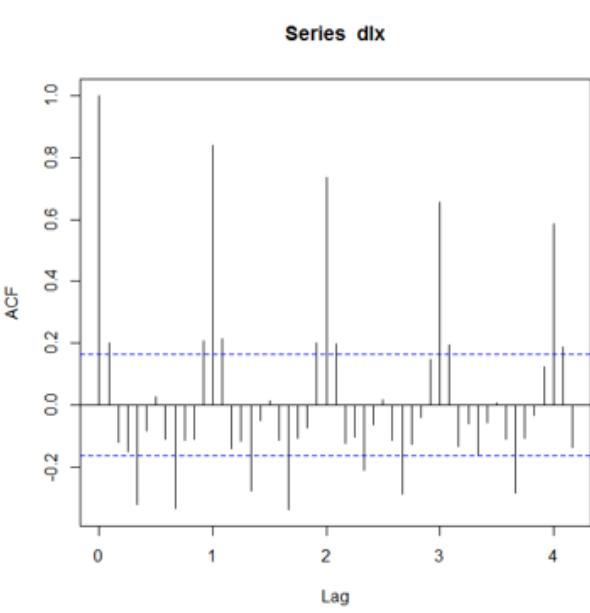
$$\Phi_p(B^s)\phi(B)\nabla_s^D\nabla^d x_t = \delta + \Theta_Q(B^s)\theta(B)w_t$$

$$\nabla_s^D = (1 - B^s)^D$$

- How to identify SARIMA?
  - ① Perform differencing first (trend)
  - ② Investigate ACF  $\rightarrow$  slowly decays at peaks?
    - ① Yes  $\rightarrow$  Additional differencing by  $\nabla_s^D$
  - ③ Model non-seasonal part
  - ④ Model seasonal part (check peaks), check ACF and PACF of residuals

# SARIMA

- Example: Air passengers



# SARIMA

- Remove AR term!

Is one model much better than the other one?

$(1, 1, 1) \times (1, 1, 0)_{12}$

```
> 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

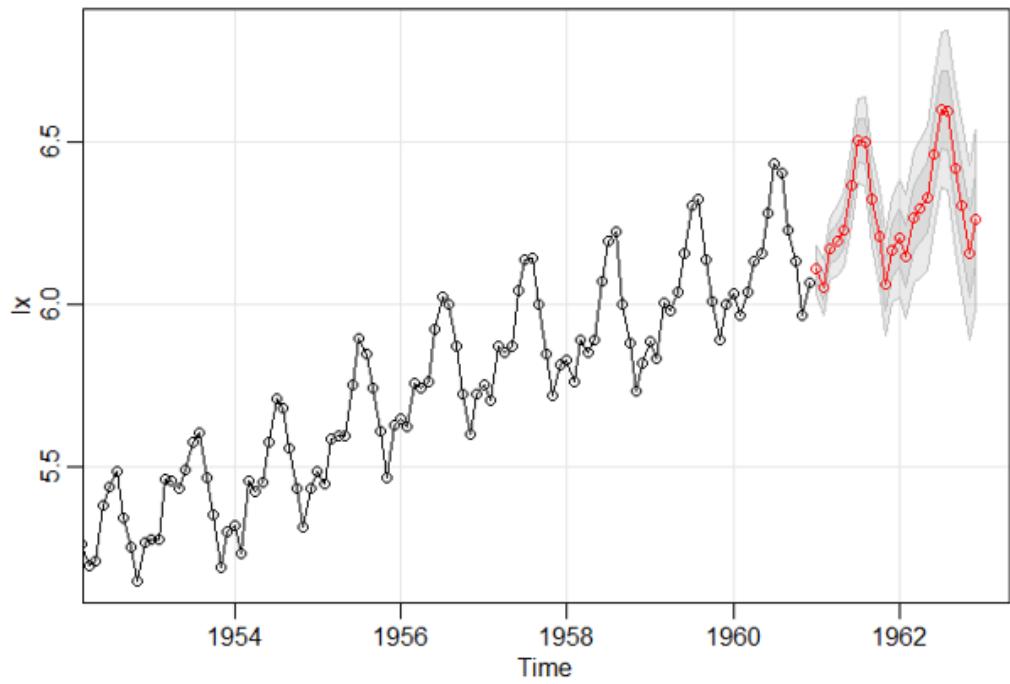
sigma^2 estimated as 0.001425:  log likelihood = 241.73,  aic = -475.47
> m2$fit
Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
REPORT = 1, reltol = tol))

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
```

# SARIMA

- Forecasting



## Read home

- Shumway and Stoffer, section 3.9
- R code: sarima, sarima.for, runs

## **Teaching Session III**

# 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

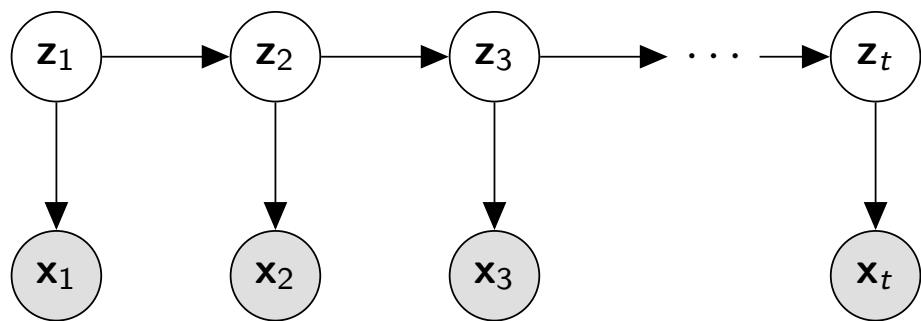
- ARIMA models
- State space models (2 lectures, 1 teaching session with hand-in, 1 computer lab with short report)
  - ▶ Linear and Gaussian state space models (Chapter 6.1)
  - ▶ Kalman filtering, Kalman smoothing and Forecasting (Chapter 6.2)
  - ▶ Maximum likelihood estimate of the state space models (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

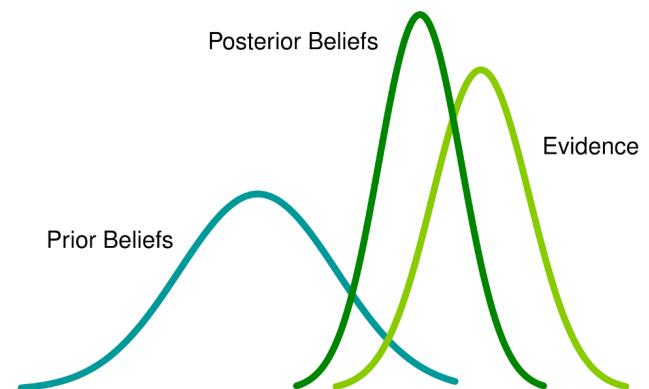
$$f(z|x) \propto f(x|z)f(z)$$

## Probability Calculus

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

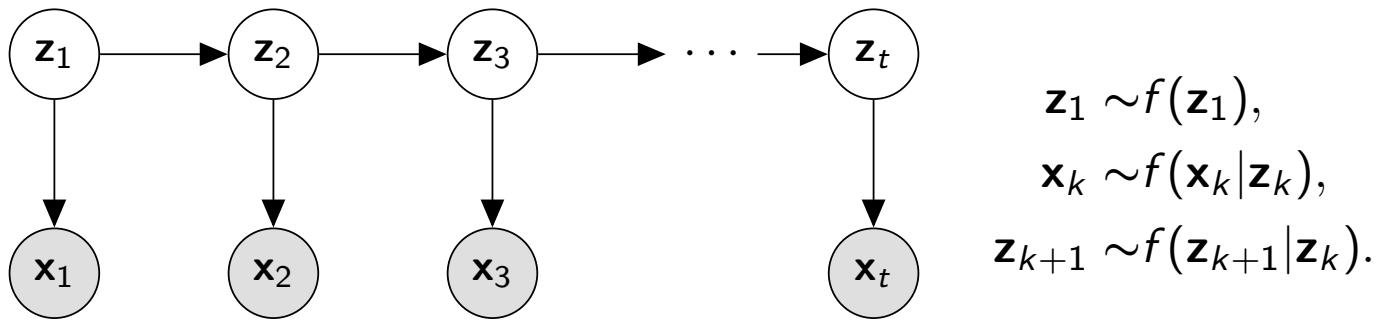
$$f(z, x) = f(x|z)f(z)$$

$$f(z) = \int f(z, x) dx$$



# Online recursive algorithms

Consider a stochastic dynamical system represented by the following recursion



The Bayesian filtering recursion corresponds to computing the posterior distributions  $f(\mathbf{z}_k | \mathbf{x}_{1:k})$ ;

$$f(\mathbf{z}_k | \mathbf{x}_{1:k}) = \frac{f(\mathbf{z}_k | \mathbf{x}_{1:k-1}) f(\mathbf{x}_k | \mathbf{z}_k)}{\int f(\mathbf{z}_k | \mathbf{x}_{1:k-1}) f(\mathbf{x}_k | \mathbf{z}_k) d\mathbf{z}_k}$$

The density  $f(\mathbf{z}_k | \mathbf{x}_{1:k-1})$  in the numerator which is called the predicted density of  $\mathbf{z}_k$  and is obtained by integration as in

$$f(\mathbf{z}_k | \mathbf{x}_{1:k-1}) = \int f(\mathbf{z}_k | \mathbf{z}_{k-1}) f(\mathbf{z}_{k-1} | \mathbf{x}_{1:k-1}) d\mathbf{z}_{k-1}.$$

# 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.

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

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



# The Kalman Filter's Foundation

Let  $\mathbf{z}$  have a normal prior distribution with mean  $\mu$  and covariance  $\Sigma$ , 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:**  $f(\mathbf{y}_1)f(\mathbf{y}_2|\mathbf{y}_1) = f(\mathbf{y}_1, \mathbf{y}_2)$

$$N(\mathbf{y}_1; \mu, \Sigma)N(\mathbf{y}_2; C\mathbf{y}_1, R) = N\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}; \begin{bmatrix} \mu \\ C\mu \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma C^T \\ C\Sigma & C\Sigma C^T + R \end{bmatrix}\right)$$

**Property 2: marginalization and conditioning**

If  $\mathbf{y}_1, \mathbf{y}_2$  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

$$\mu_t = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} m_{t|t} \\ Am_{t|t} \\ CAm_{t|t} \end{bmatrix}$$

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  $f(\mathbf{z}_1) = N(\mathbf{z}_1; m_0, P_0)$

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

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

---

1: **Inputs:**  $A_t$ ,  $C_t$ ,  $Q_t$ ,  $R_t$ ,  $m_0$ ,  $P_0$  and  $\mathbf{x}_{1:T}$ .

*initialization*

2:  $m_{1|0} \leftarrow m_0$ ,  $P_{1|0} \leftarrow P_0$

3: **for**  $t = 1$  to  $T$  **do**

*observation update step*

4:  $K_t \leftarrow P_{t|t-1}C_t^T(C_tP_{t|t-1}C_t^T + R_t)^{-1}$

5:  $m_{t|t} \leftarrow m_{t|t-1} + K_t(\mathbf{x}_t - C_t m_{t|t-1})$

6:  $P_{t|t} \leftarrow (I - K_t C_t)P_{t|t-1}$

*prediction step*

7:  $m_{t+1|t} \leftarrow A_t m_{t|t}$

8:  $P_{t+1|t} \leftarrow A_t P_{t|t} A_t^T + Q_{t+1}$

9: **end for**

10: **Outputs:**  $m_{t|t}$ ,  $P_{t|t}$  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**

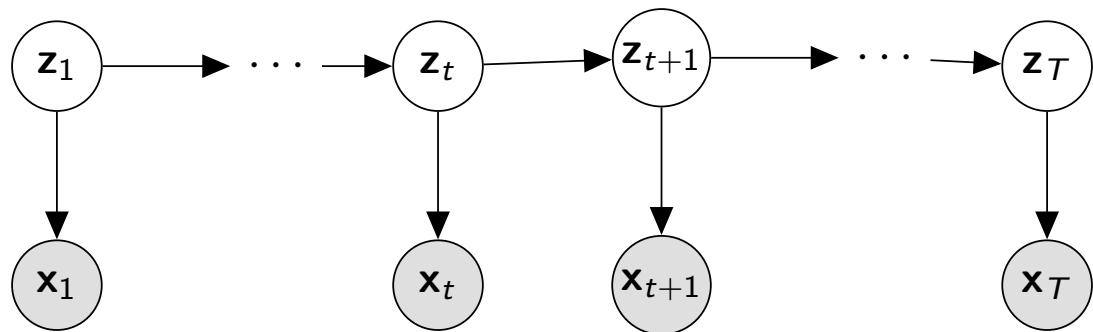
$$\mathbf{z}_t = A_{t-1} \mathbf{z}_{t-1} + e_t$$

## 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:

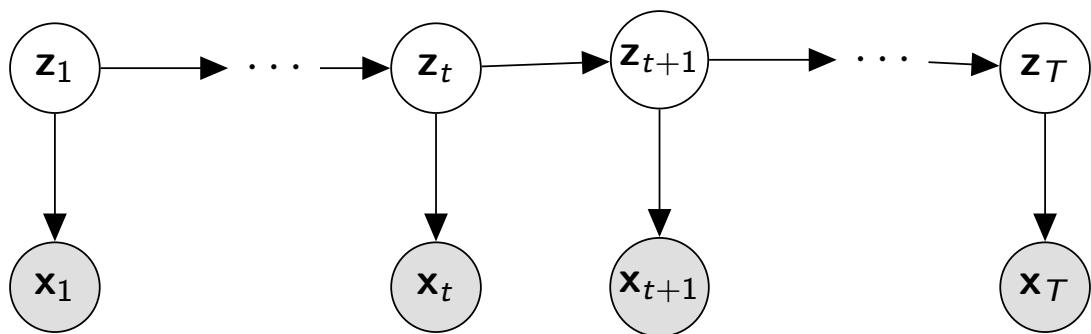
$$f(\mathbf{z}_t | \mathbf{z}_{t+1}, \mathbf{x}_{1:t}) = N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t)$$

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

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

Recall that

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

## RTS Smoother's derivation

where

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

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

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

Hence,

$$\begin{aligned} f(\mathbf{z}_{t+1}, \mathbf{z}_t | \mathbf{x}_{1:T}) &= N(\mathbf{z}_{t+1}; m_{t+1|T}, P_{t+1|T}) N(\mathbf{z}_t; \tilde{m}_t, \tilde{P}_t) \\ &= N\left(\left[\begin{array}{c} \mathbf{z}_t \\ \mathbf{z}_{t+1} \end{array}\right], \left[\begin{array}{c} \cdot \\ m_{t+1|T} \end{array}\right], \left[\begin{array}{cc} \cdot & \cdot \\ \cdot & P_{t+1|T} \end{array}\right]\right) \end{aligned}$$

# RTS smoother's backwards recursion

Prove the backwards recursion of the RTS smoother for following state space model with initial prior on the state  $f(\mathbf{z}_1) = N(\mathbf{z}_1; m_0, P_0)$

$$\begin{aligned}\mathbf{z}_t &= A_{t-1}\mathbf{z}_{t-1} + e_t, & e_t &\sim N(0, Q_t) \\ \mathbf{x}_t &= C_t\mathbf{z}_t + \nu_t, & \nu_t &\sim N(0, R_t)\end{aligned}$$

---

- 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:    $m_{t|\tau} \leftarrow m_{t|t} + G_t(m_{t+1|\tau} - A_t m_{t|t})$
  - 5:    $P_{t|\tau} \leftarrow P_{t|t} + G_t(P_{t+1|\tau} - P_{t+1|t})G_t^T$
  - 6: **end for**
  - 7: **Outputs:**  $m_{t|\tau}, P_{t|\tau}$
-

# Read home

- Shumway and Stoffer, Chapters 6.1 and 6.2

## **Summary**

# 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)

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

- $w_t \sim wn(0, \sigma_w^2)$
- $\theta_1, \dots, \theta_q$  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
- $w_t \sim wn(0, \sigma_w^2)$
- $\phi_1, \dots, \phi_p$  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.  $|z'| > 1$

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