Time Series: Lab material

Time Series Using R

Introduction

Many datasets are available in `R`, in the `datasets` library. We will use several functions from traditional packages. Install those if needed.

```
library(stats)
library(datasets)
```

We will start by working on the dataset of level of the Lake Huron. Check it and its description.

```
LakeHuron
```

```
## Time Series:
## Start = 1875
## End = 1972
## Frequency = 1
##  [1] 580.38 581.86 580.97 580.80 579.79 580.39 580.42 580.82 581.40 581.32
## [11] 581.44 581.68 581.17 580.53 580.01 579.91 579.14 579.16 579.55 579.67
## [21] 578.44 578.24 579.10 579.09 579.35 578.82 579.32 579.01 579.00 579.80
## [31] 579.83 579.72 579.89 580.01 579.37 578.69 578.19 578.67 579.55 578.92
## [41] 578.09 579.37 580.13 580.14 579.51 579.24 578.66 578.86 578.05 577.79
## [51] 576.75 576.75 577.82 578.64 580.58 579.48 577.38 576.90 576.94 576.24
## [61] 576.84 576.85 576.90 577.79 578.18 577.51 577.23 578.42 579.61 579.05
## [71] 579.26 579.22 579.38 579.10 577.95 578.12 579.75 580.85 580.41 579.96
## [81] 579.61 578.76 578.18 577.21 577.13 579.10 578.25 577.91 576.89 575.96
## [91] 576.80 577.68 578.38 578.52 579.74 579.31 579.89 579.96
```

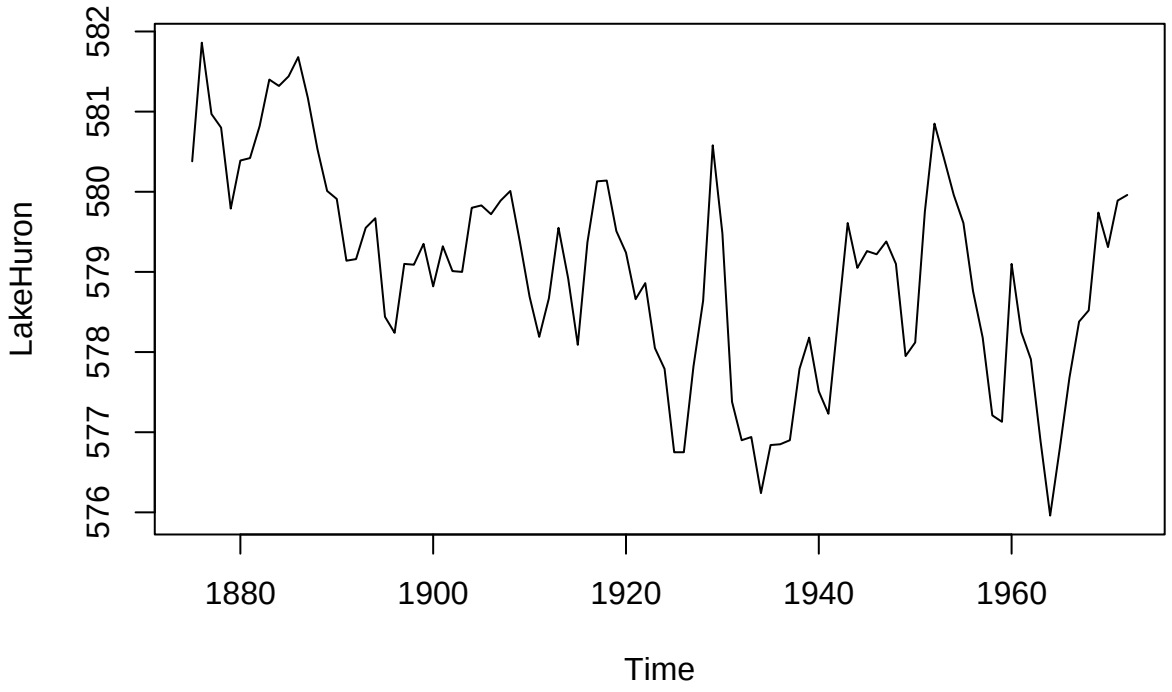
```
?LakeHuron
```

Note that the format of this object is "Time Series" meaning that it is already coerced to be represented with time (it is defined with an array of values, a start and end date and a frequency).

The function `ts` can be used to create a time series, check it and its associated functions. In particular you can have a look at the `ts.intersect` and `ts.union` functions.

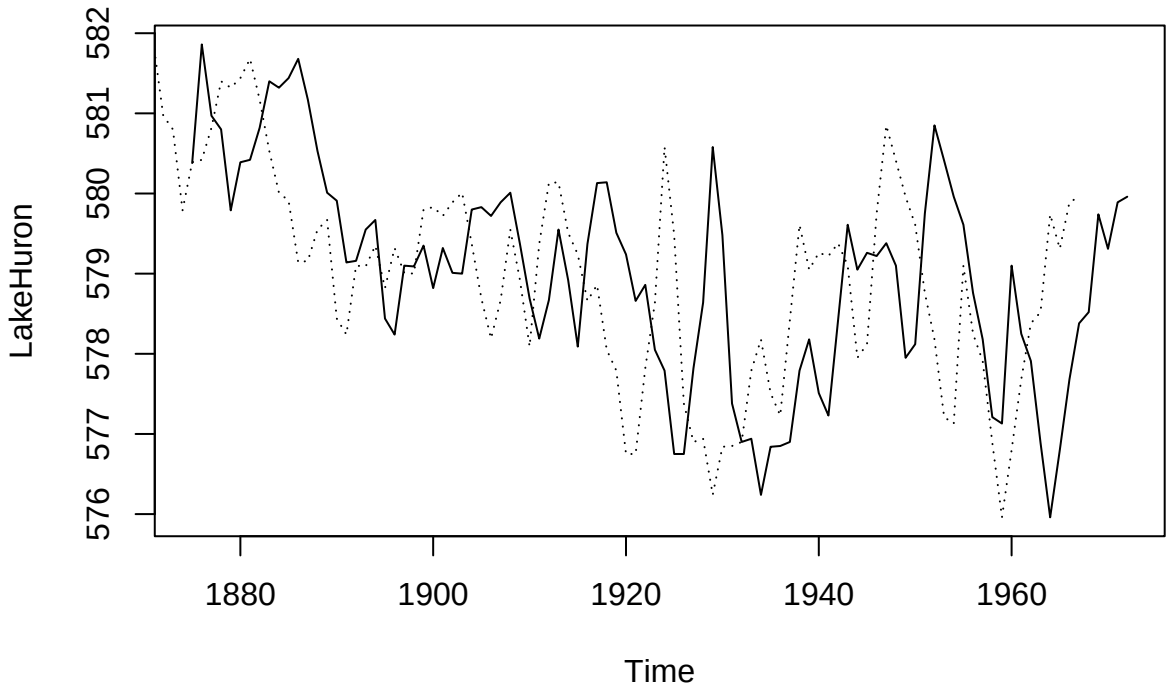
You can plot a time series using the standard `plot` function.

```
plot(LakeHuron)
```



You can extract a subpart of the time series with the `window` function, check its arguments. You can shift a time series with the `lag` function.

```
plot(LakeHuron)
lines(lag(LakeHuron, k=5), lty=3)
```



The `diff` function takes the difference between a series and its lagged value. Check it and its arguments.

Polynomial regression

Consider a polynomial regression model of the form

$$X_t = s(t) + w_t = \beta_0 + \beta_1 t + \cdots + \beta_k t^k + w_t$$

where w is a stationary process. The parameters β's can be estimated with least squares, minimizing

$$\sum_{t=1}^{n} (x_t - s(t))^2 = \sum_{t=1} (x_t - (\beta_0 + \beta_1 t + \cdots + \beta_k t^k))^2,$$

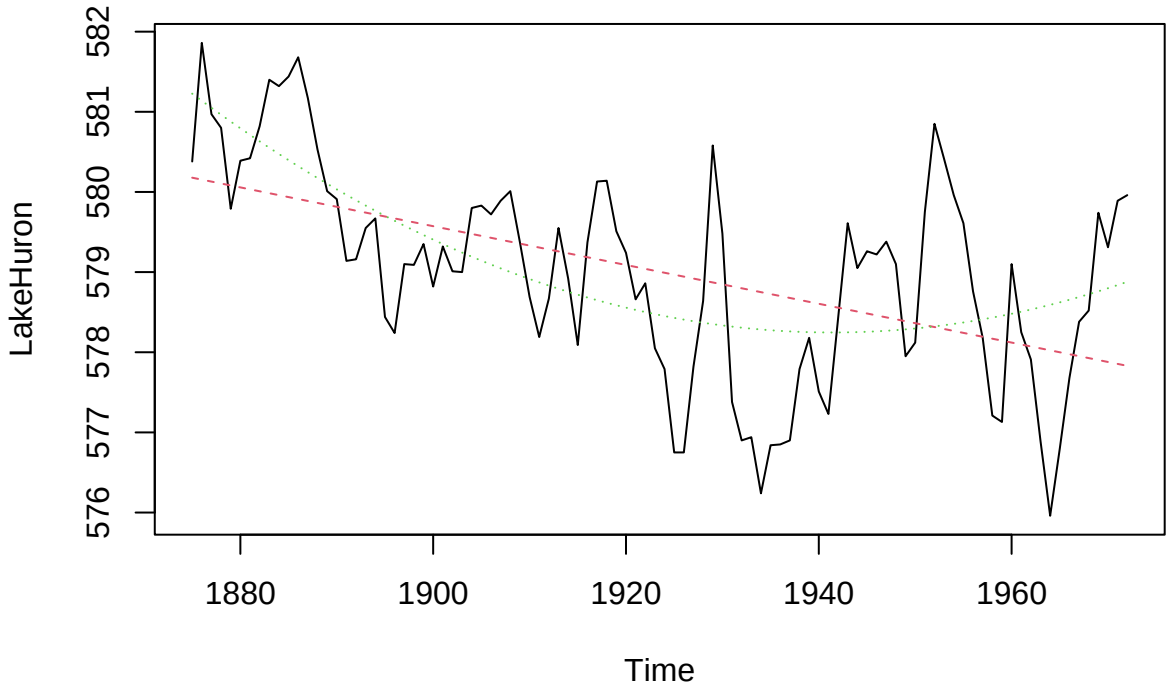
leading to an estimator (homework: do the computation) $\hat{\beta} = (X^T X)^{-1} X^T y$, with $y^T = (y_1, \ldots, y_n)$ and X is the matrix of the $(t^{j_i})_{i,j}$.

Comments:

- sensitivity to points at the extremities of the time span;
- polynomials are quite limited;
- may require to use a family of orthogonal polynomials (see Wikipedia) for numerical reasons;
- easily copes with missing values and unequally spaced observations.

One way to run this inference in `R` is to use the `lm` function:

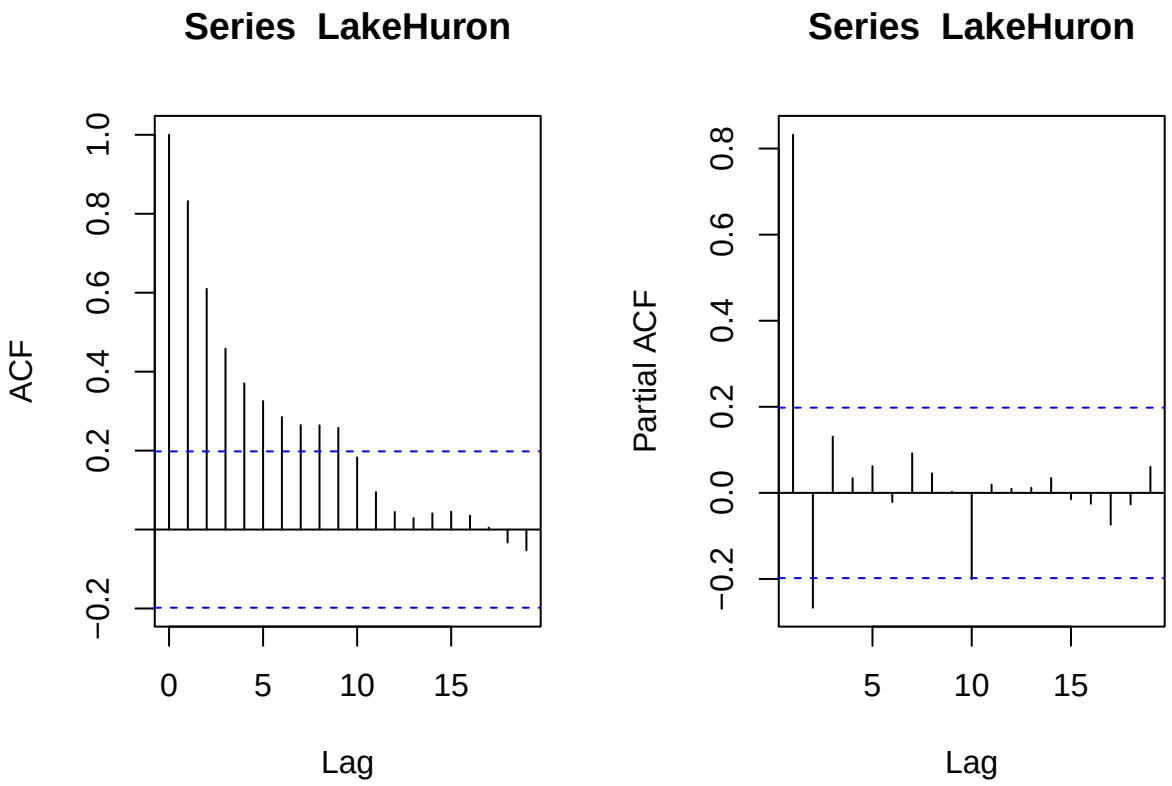
```
plot(LakeHuron)
linfit <- ts(fitted(lm(LakeHuron ~ c(1:98))), frequency = 1,
             start = start(LakeHuron)[1], end = end(LakeHuron)[1])
lines(linfit,lty=2,col=2)
polyfit <- ts(fitted(lm(LakeHuron ~ poly(c(1:98),2))), frequency = 1,
              start = start(LakeHuron)[1], end = end(LakeHuron)[1])
lines(polyfit,lty=3,col=3)
```



Second-order summaries

The `acf` function compute and plots the c_k and r_k, the estimated auto-covariance and -correlation, see notes and help. The `pacf` function performs the same for partial autocorrelation and autocovariance.

```r
par(mfrow=c(1,2))
acf(LakeHuron)
pacf(LakeHuron)
```



- What do the horizontal lines represent?
- Can you identify some useful models for this dataset?

The Ljung-Box test statistic at lag 3 can be obtained with:

```r
Box.test(LakeHuron, type="Ljung-Box", lag=3)
```

```
##
##  Box-Ljung test
##
## data:  LakeHuron
## X-squared = 129.56, df = 3, p-value < 2.2e-16
```

What can you conclude from that?

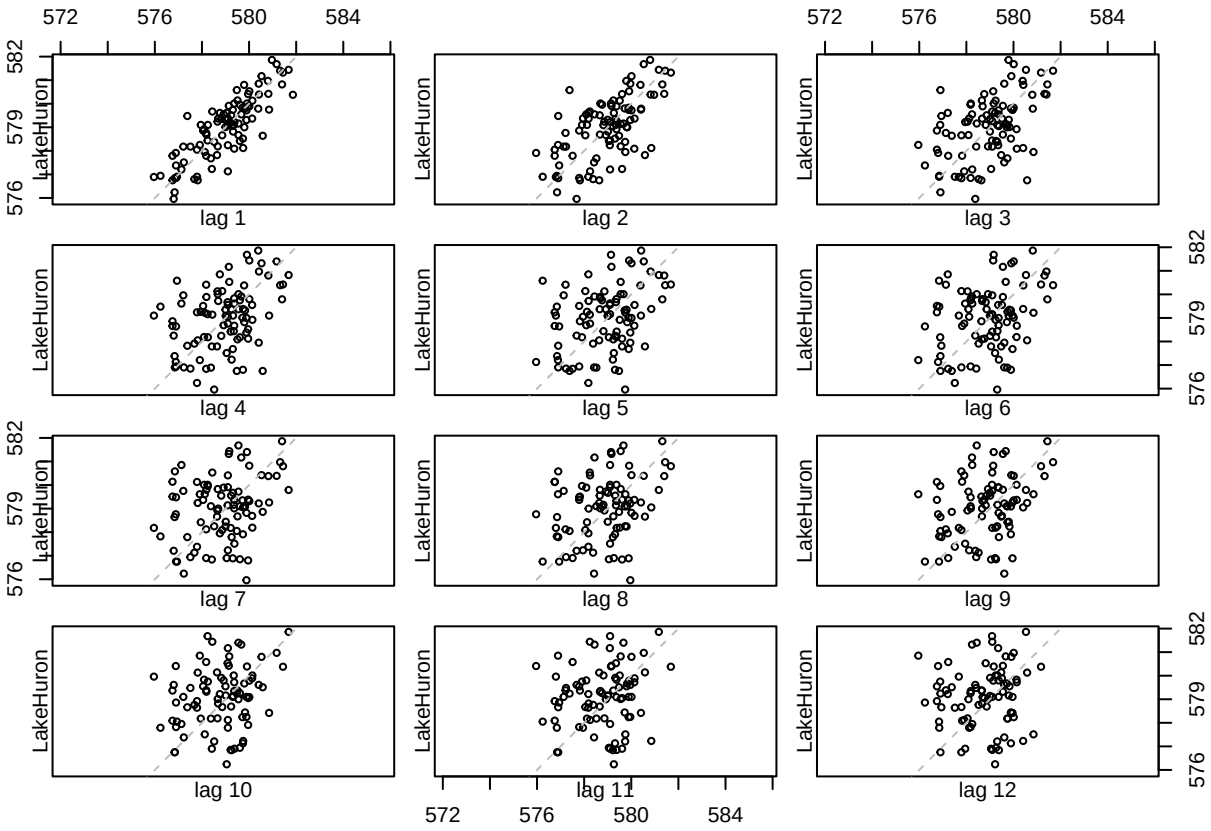
If you want to obtain those value for several lags, you can use the `sapply` function:

```r
sapply(2:15, function(lag) Box.test(LakeHuron, type="Ljung-Box", lag)$p.value)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Another way to visualize the autocorrelation is via the `lag.plot` function.

```r
lag.plot(LakeHuron, type="p", lag=12, do.lines=FALSE)
```



Fitting and assessing AR models

Likelihood inference If you don't remember what is a likelihood, check your previous years course or ask Wikipedia.

We write L the likelihood and ℓ the log likelihood. A natural estimator for the parameters θ will then be $\hat{\theta} := argmax_\theta(\ell(x;\theta))$. We remind that if the parameters are in higher dimension, you might have to compute the Hessian H of the likelihood and check its definite-positiveness.

A useful property, when the number of samples is large and the model is *regular* is that if θ^0

$$\hat{\theta} \sim \mathcal{N}(\theta^0, H(\hat{\theta})^{-1}).$$

From this we can write an approximate confidence interval at level α as

$$\hat{\theta}_r \pm z_\alpha v_{rr}^{1/2}$$

where the $v_{i,j}$ are the elements of H and z_α is the quantile of the normal distribution.

To implement this procedure in `R`:

- Implement $-\ell$ the negative log likelihood (usually the most complex part);
- minimise $-\ell$ numerically, with a method that returns the Hessian H;
- Compute $H(\hat{\theta})^{-1}$ and to get $v_{rr}^{-1/2}$ as the error for the corresponding $\hat{\theta}_r$.

Likelihood ratio statistic

To compare two models A and B several methods can be used, one of the particular case is when a model is included in the other (for example: model A is a particular case of B). In the case of time series, the

white noise model $(X_1, \ldots, X_n) \sim \mathcal{N}(\mu, \sigma^2)^{\otimes n}$ is nested into the Gaussian AR(1) model, as the first one corresponds to the second with $\alpha = 0$.

Remark: it is always true that $max\ell_B \geq max\ell_A$, as the space of parameters for B includes the space of parameters for A.

We can then compare the models using the likelihood ratio:

$$D = 2(max\ell_B - max\ell_A).$$

If the simpler (A) model is true and B has q more parameters than A, then

$$D \sim \chi^2_q .$$

Fitting AR models with `arima` The function `arima` fits an ARIMA model to a time series. It is possible to change the order of the fit, and thus to fit ARMA, AR and MA models with it.

For example, to fit an AR(2) model (with constant),

```
fitar2 <- arima(x=LakeHuron, order=c(2, 0, 0))
fitar2
```

```
##
## Call:
## arima(x = LakeHuron, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       1.0436  -0.2495   579.0473
## s.e.  0.0983   0.1008     0.3319
##
## sigma^2 estimated as 0.4788:  log likelihood = -103.63,  aic = 215.27
```

Note that the constant fitted corresponds to the estimate of the constant mean, and not to the constant describe in the lectures.

To assess the fit, you can extract the residuals with `fitar2$residuals` and then

- plot of the residuals
- check the correlograms of the residuals
- examine the Ljung-Box stats of the residuals

Using one or more of those methods, what can you say on this fit?

Simulating an AR process The function `arima.sim` can be used for that purpose:

```
ar1 <- arima.sim(n=100, model=list(ar=0.9))
ma1 <- arima.sim(n=100, model=list(ma=0.8))
```

Apply the second order summaries function to these two processes.

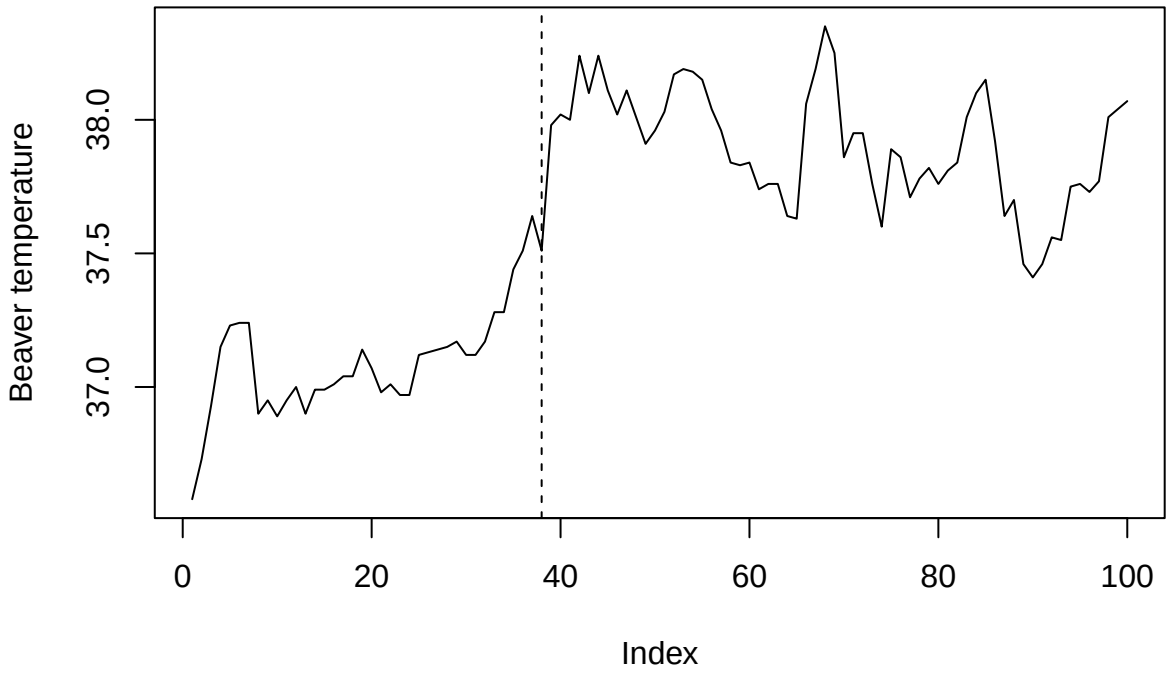
Application: Statistical model for beaver data Practical: Statistical models for beaver data

The data relates to 100 consecutive telemetric measurements on the body temperature of a female Canadian beaver, Castor canadensis, taken at 10-minute intervals. The animal remained in its lodge for the first 38 recordings and then moved outside, at which point there was a sustained temperature rise.

Download the data from Learn and load them in `R`.

```r
beaver=beaver2

plot(beaver[,3], ylab="Beaver temperature", type="l")
abline(v=38, lty=2)                          # change point
```



We will consider 4 models on this data:

- M1: independent normal observations;
- M2: indepdent normal observations with different means before and after the change point but same variance;
- M3: AR(1) with constant mean;
- M4: AR(1) with different constant means on each side of the change point.

Write mathematically these models.

Note that x_0 is a required parameter of the AR models if we want to include all the data points. We have several ways to deal with this:

- choose an arbitrary value for x_0
- use the stationary distribution $\mathcal{N}(\mu, \sigma^2/(1-\alpha^2))$
- drop the likelihood term of the first observation.

```r
## negative log likelihood
beaver <- as.data.frame(beaver)
logL1 <- function(th, y) -sum(dnorm(y$temp, mean=th[1], sd=th[2], log=T))

## initial values for mu, sigma
init1 <- c(mean(beaver$temp),sd(beaver$temp))

## use minimisation routine nlm
fit1 <- nlm(f=logL1, p = init1, iterlim=500, hessian=T, y=beaver[-1,])

## components of the fitted object fit1
```

```r
names(fit1)

## parameter estimates
fit1$estimate

## standard errors from square root of diagonal of inverse Hessian
sqrt(diag(solve(fit1$hessian)))
```

Fitting M1

```r
logL2 <- function(th, y) {
    mu <-  th[1]+y$activ*th[3]
    -sum(dnorm(y$temp, mean=mu, sd=th[2], log=T))
}
init2 <- c(init1,0.5)

fit2 <- nlm(f=logL2, p = init2, iterlim=500, hessian=T, y=beaver[-1,])
fit2$minimum
fit2$estimate
sqrt(diag(solve(fit2$hessian)))

# likelihood ratio statistic to compare M2 and M1
(w21 <- 2*(fit1$minimum-fit2$minimum))
```

Fitting M2

```r
logL3 <- function(th, y) {
    mu <- th[1]+th[3]*(y$temp[-100]-th[1])
    -sum(dnorm(y$temp[-1], mean=mu, sd=th[2], log=T))
}
init3 <- c(init1,0.5)

fit3 <- nlm(f=logL3, p = init3, iterlim=500, hessian=T, y=beaver)
fit3$minimum
fit3$estimate

sqrt(diag(solve(fit3$hessian)))
(w31 <- 2*(fit1$minimum-fit3$minimum))
```

Fitting M3

```r
logL4 <- function(th, y) {
    mu <- th[1]+th[4]*y$activ[-1]+th[3]*(y$temp[-100]-th[1]-th[4]*y$activ[-100])
    - sum(dnorm(y$temp[-1], mean=mu, sd=th[2], log=T))
}
init4 <- c(init1,0.5,0.4)

fit4 <- nlm(f=logL4, p = init4, iterlim=500, hessian=T, y=beaver)
fit4$minimum
fit4$estimate
```

```r
sqrt(diag(solve(fit4$hessian)))
(w43 <- 2*(fit3$minimum-fit4$minimum))
```

Fitting M4 For each of the models, find the number of parameters and the value of $\hat{\ell} = max\ell$.

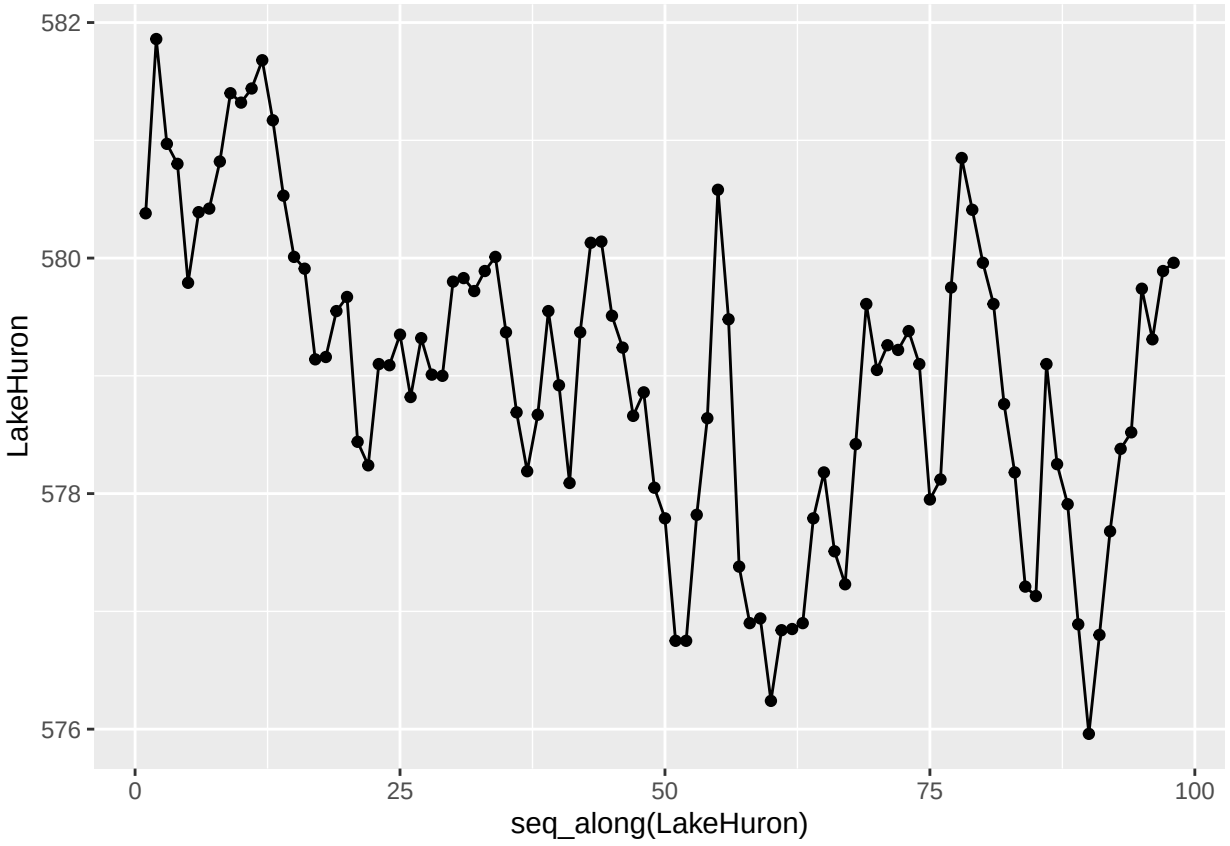
Give all the estimates and the standard errors of the parameters of each models.

Time Series plots

Basics As stated previously, you can use the standard plot functions. You can also use `ggplot` if you are used to it

```r
library(ggplot2)
p <- ggplot(data=as.data.frame(LakeHuron), aes(y = LakeHuron, x = seq_along(LakeHuron)))
p + geom_line() + geom_point()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



As per usual graph representations, aspect ratio and other graphical parameters are crucial to the understanding of the data. Try changing it in the plotting function you chose.

ARMA, ARIMA and consorts

Parameter space of causal AR(2) processes

Reminder: the recurrence equation defining the AR(2) process writes:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + w_t, \qquad (w_t) \sim (0, \sigma^2).$$

To determine the space of parameters for which the process is causal we need to check the roots of the associated polynomial Φ.

If we write ρ_1 and ρ_2 the autocorrelation function at lag 1 and 2 respectively, show that:

$$\rho_1 = \phi_1 + \phi_2\rho_1;$$
$$\rho_2 = \phi_1\rho_1 + \phi_2.$$

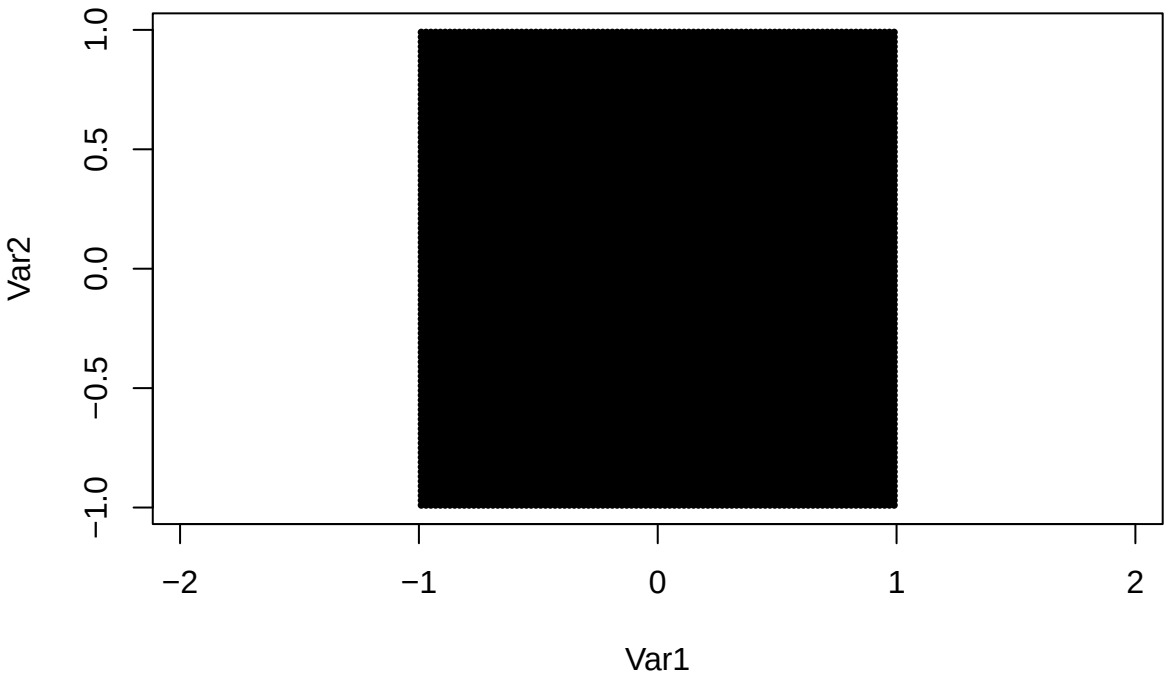
The system of equations is written in a matrix form $A\underline{\phi} = \underline{\rho}$.

Therefore, we can solve this equation:

$$\phi_= \frac{\rho_1 - \rho_1\rho_2}{1-\rho_1^2}, \qquad \phi_2 = \frac{\rho_2 - \rho_1^2}{1-\rho_1^2}.$$

From this, we can see that conditions on (ϕ_1, ϕ_2) induce conditions on (ρ_1, ρ_2). And we know that $-1 \leq \rho_\cdot \leq 1$. From that we can see which parameters allow for this to be true.

```
rho    <- seq(-0.99,0.99, len=100)
rhomat <- expand.grid(rho,rho)
plot(rhomat, asp=1, cex=0.5, pch=16)
```



We can then compute for each point of the grid the value of $\phi_\cdot$, then we can check if the roots of Φ have modulus greater than 1. The roots write

$$\frac{-\phi_1 \pm (\phi_1^2 + 4\phi_2)^{1/2}}{2\phi_2}.$$

We can then write a function that performs all of these operations:

```
library(dplyr)
arparspace <- function(rho1, rho2)
```

```
{
    phi1 <- (rho1-rho1*rho2)/(1-rho1^2)
    phi2 <- (rho2-rho1^2)/(1-rho1^2)
    r1   <- as.complex((-phi1 + sqrt(as.complex(phi1^2 + 4*phi2)))/(2*phi2))
    r2   <- as.complex((-phi1 - sqrt(as.complex(phi1^2 + 4*phi2)))/(2*phi2))
    cond <- Mod(r1) > 1 & Mod(r2) > 1
    return(data.frame(phi1, phi2, rho1, rho2, r1, r2, cond))
}
```

Next, we apply this function to the grid of points

```
phimat <- arparspace(rhomat[,1], rhomat[,2])
```

We can then filter the results depending on the nature of the correlation:

```
phimat <- phimat %>% mutate(negneg=(rho1 < 0 & rho2 < 0),
                            posneg=(rho1 > 0 & rho2 < 0),
                            negpos=(rho1 < 0 & rho2 > 0),
                            pospos=(rho1 > 0 & rho2 > 0)) %>%
    filter(cond==1)
```

And then, we can plot the results. Note that the grid is irregular because of the transformations.

```
library(dplyr)
library(ggplot2)
library(gridExtra)

p1 <- ggplot(phimat) +
    geom_point(aes(x = phi1, y = phi2, colour=negneg), alpha=0.5 ) +
    geom_abline(intercept=1, slope=1)+
    geom_abline(intercept=1, slope=-1)+
    geom_abline(intercept=-1, slope=0)+
    theme_classic()

p2 <- ggplot(phimat) +
    geom_point(aes(x = phi1, y = phi2, colour=posneg), alpha=0.5 )  +
    geom_abline(intercept=1, slope=1)+
    geom_abline(intercept=1, slope=-1)+
    geom_abline(intercept=-1, slope=0)+
    theme_classic()

p3 <- ggplot(phimat) +
    geom_point(aes(x = phi1, y = phi2, colour=negpos), alpha=0.5 )  +
    geom_abline(intercept=1, slope=1)+
    geom_abline(intercept=1, slope=-1)+
    geom_abline(intercept=-1, slope=0)+
    theme_classic()

p4 <- ggplot(phimat) +
    geom_point(aes(x = phi1, y = phi2, colour=pospos), alpha=0.5 )  +
    geom_abline(intercept=1, slope=1)+
    geom_abline(intercept=1, slope=-1)+
    geom_abline(intercept=-1, slope=0)+
    theme_classic()

a <- grid.arrange(p1, p2, p3, p4, nrow=2)
```

Harder exercise: Partial autocovariance and geometry

Computing the Autocovariance of ARMA(p, q) processes

`R` allows to compute theoretical ACF or PACF. For details on how second order summaries are computed, see Chapter 3.3 of Brockwell and Davis (1991).

From Example 3.3.1 in Brockwell and Davis (1991): consider the ARMA$(2, 1)$ process given by

$$(1 - B + B^2/4)X_t = (1 + B)w_t, \qquad (w_t) \sim (0, \sigma^2)$$

The autocovariance of the process is:

$$\gamma_X(k) = \sigma^2 2^{-k}(32/3 + 8k),$$

and the autocorrelation

$$\rho_X(k) = 2^{-k}(1 + 3k/4).$$

The function `ARMAacf` can be used to compute the values of ρ_X for all k.

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

Compare those results with the theoretical ones.

Discuss the following plots of autocorrelation in relation to the analysis in section 2.1

```
par(mfrow=c(3,2))
acf1 <- ARMAacf(ar=c(1/2, 1/3), lag.max=10)
acf2 <- ARMAacf(ar=c(3/2, -2/3), lag.max=10)
acf3 <- ARMAacf(ar=c(-1/2, -1/3), lag.max=10)
acf4 <- ARMAacf(ar=c(-1/2, -1/2), lag.max=10)
acf5 <- ARMAacf(ar=c(-3/2, -2/3), lag.max=10)
acf6 <- ARMAacf(ar=c(+1/2, -1/2), lag.max=10)
## ------------------------------------------
plot(acf1, type="h", ylab="autocorrelation")
points(2:3, acf1[2:3], pch=16, cex=2, col="red")
plot(acf2, type="h", ylab="autocorrelation")
points(2:3, acf2[2:3], pch=16, cex=2, col="red")
plot(acf3, type="h", ylab="autocorrelation")
points(2:3, acf3[2:3], pch=16, cex=2, col="red")
plot(acf4, type="h", ylab="autocorrelation")
points(2:3, acf4[2:3], pch=16, cex=2, col="red")
plot(acf5, type="h", ylab="autocorrelation")
points(2:3, acf5[2:3], pch=16, cex=2, col="red")
plot(acf6, type="h", ylab="autocorrelation")
points(2:3, acf6[2:3], pch=16, cex=2, col="red")
```

Fitting the ARMA$(1, 1)$ model to `LakeHuron` data Reminder: we can the the `arima` function.

Fit an ARMA$(1, 1)$ to the data.

```
mod <- arima(x = LakeHuron, order = c(1, 0, 1))
mod
```

What are the estimates of the parameters? Build confidence intervals for those parameters. Precise the assumptions you are making.

Solving an AR(p) model Consider a stationary AR(p) process (X_t). If the process is initialised at $X_0 = x_0, \dots, X_k = x_k$ with $x_1, \dots, x_k \in \mathbb{R}$ then we can solve by substitution the stochastic difference equation

$$\Phi(B)X_t = w_t.$$

This difference equation determines the one step transition probability of the time series process since

$$\mathbb{P}(X_t \leq y \mid \mathbf{X_{t,k}} = \mathbf{x_{t,k}}) = \mathbb{P}(\mathbf{w_t} \leq \mathbf{y} - \sum_{\mathbf{i=1}}^{\mathbf{p}} \phi_\mathbf{i} \mathbf{x_{t-1}}),$$

where $\mathbf{X_{t,k}} := (\mathbf{X_{t-1}}, \dots, \mathbf{X_{t-k}})$, and $\mathbf{x_{t,k}} := (\mathbf{x_{t-1}}, \dots, \mathbf{x_{t-k}})$. For the step t transition probability, we can propose a solution of the form

$$X_t = \rho_t X_t 0 + M_t, \qquad t = 1, 2, \dots$$

where ρ is a sequence of real valued scalars and M is a real valued time series independent of X_0. We have

$$\begin{aligned} M_t = X_t - \rho_t X_0 &= ((1 - \Phi(B))X_t - \rho_t X_0) + \Phi(B)X_t \\ &= ((1 - \Phi(B))X_t - \rho_t X_0) + w_t \\ &= \sum_i \phi_i \rho_{t-i} X_0 + \sum_i \phi_i M_{t-i} - \rho_t X_0 + w_t \\ &= X_0 \sum_i \phi_i \rho_{t-i} + \sum_i \phi_i M_{t-i} - \rho_t X_0 + w_t \end{aligned}$$

From the last equality, the only independent solution that satisfies the initial hypothesis (independence of M_t and X_0) is:

$$\forall t, \qquad \rho_t = \sum_i \phi_i \rho_{t-i}.$$

This results in a unique solution:

$$\begin{aligned} X_t &= \rho_t X_0 + M_t, \qquad X_0 \text{ independant from } M \\ M_t &= \sum_i \phi_i M_{t-i} + w_t. \end{aligned}$$

Interestingly, for large t, the process forgets the initial value. This comes from the stationarity, as ρ_t tails off to 0, leading to:

$$X_t = M_t + o_p(1)$$

where $M_t \sim \text{AR}(p)$.

Most of the works on stationary Time Series are based on this forgottenability of the past values: independence over long time spans. Yet, in many applications, this dependency over long times is not negligible. One of the solutions to deal with these problems is to extend the concept of ARIMA(p, d, q) to non-integer d through fractional differencing.

Central to this development is the concept of Hurst exponent. However, as discussed by Bhattacharya et al. (1983), apparent long-range dependence in data cannot be distinguished from a non-stationary mean or trend. An alternative approach to constructing covariance functions with such behaviour is therefore to use a two-stage model, where the local behaviour is treated separately from the long-range behaviour.

Yule-Walker equations

We remind that the AR(p) recurrence equation writes:

$$X_s = \sum_{i=1}^{p} \phi_i X_{s-i} + w_s.$$

Note that X is nearly linear in $X_{s-1}, \ldots, X_{s-p}$. The AR(p) model can be written more compactly

$$\Phi(B) X_s = w_s.$$

Remind yourself of the stationarity condition under this writing.

Assuming stationarity and multiplying each side by X_{s-t} we get

$$\begin{aligned} E[X_s X_{s-t}] &= E\left[\left(\sum_{i=1}^{p} \phi_i X_{s-i+w_t}\right) X_{s-t}\right] \\ &= \sum_{i=1}^{p} \phi_i E[X_{s-i} X_{s-t}] + 0. \end{aligned}$$

Leading to:

$$\forall t, \qquad \gamma_X(t) = \sum_{i=1}^{p} \phi_i \gamma_{t-i}.$$

using p of those relations, we get a linear equation system that can be used to compute the γ_X values.

Vector of AR processes We can extend the notion of time series to vector. In the case of an AR process the equation becomes:

$$\mathbf{X_t} = \sum_{\mathbf{i=1}}^{\mathbf{p}} \mathbf{A_i X_{t-i}} + \mathbf{G w_t}$$

where the A_i's and G are square matrices, while $\mathbf{w}$ is a white noise vector of same dimension as $\mathbf{X}$.

We can also write a scalar AR(p) process as a vector AR(1), for example with AR(2):

$$\begin{pmatrix} X_t \\ X_{t-1} \end{pmatrix} = \begin{pmatrix} \alpha_1 & \alpha_2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} X_{t-1} \\ X_{t-2} \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} w_t \\ w_{t-1} \end{pmatrix}.$$

This is similar to writing an higher order ODE as a first order system of ODE.

ARIMA models

In the `arima` function, p, d, q are entered in the `order` argument. The downside is that it is not possible to include a drift term (a constant) when fitting an ARIMA model

$$(W_t - \mu) = \alpha_1(W_{t-1} - \mu) + \cdots + \alpha_p(W_{t-p} - \mu) + w_t - \theta_1 w_{t-1} - \cdots - \theta_q w_{t-q};$$
$$W_t = \Delta^d X_t$$

unless $d = 0$, where $(w_t) \sim (0, \sigma^2)$.

See also: http://robjhyndman.com/hyndsight/arimaconstants/.

The package `forecast` allows fitting ARIMA with a constant, with the `Arima` function.

Brownian mothin as a scaling limit of a random walk

Consider the simple symmetric random walk on $\mathbb{Z}$:

$$Y_0 = 0, \qquad Y_{t+1} = Y_{t-1} - 1\text{w.p. } 1/2;\ Y_{t-1} + 1\text{w.p. } 1/2.$$

Simulate a represent such a stochastic process.

Brownian motion can be seen as the limit of such a process, where infinitesimal jumps occurs with infinite frequency. It is not possible to simulate this process but we can approximate it with random walks.

Let $Z_t = Y_t - Y_{t-1}$, so that $E[Z_t] = 0$ and $Var(Z_t) = 1$, with Z_i independent of Z_j for $i \neq j$. Then, we can write $E[Y_t] = 0$ and $var(Y_t) = \sum_{i=1}^{t} var(Z_i) = t$.

The standard deviation after ct steps is therefore of order $c^{1/2}$. This motivates a scaled down and speeded up version of Y:

$$X_t^{(c)} := \frac{Y_{[ct]}}{\sqrt{c}}$$

where $[a]$ represents the nearest integer of a. Using scaling c, the variance is independant of the speed-up factor c. The limit when $c \to \infty$ of this process is called Brownian motion, a.k.a. Weiner process.

Note the relation of Z to Y, Z is a form of ''derivative'' of the random walk. In SDE, White Noise often designates the derivative of the Brownian motion (with the idoneous definition).

```
## Visualisation: Brownian motion as scaling limit of random walk
set.seed(111)
N <- 2^(0:14)
p <- 1/2
X <- rbinom(N[length(N)],1,p)
X[X==0] <- -1                    #+-1 spins, w.p. 1/2

x11()
for(k in 1:(length(N)-1))
{
    print(paste("k is:", k))
    l <- 1
    for(i in seq(N[k],N[k+1],len=100))
    {
        if(k > 1) l<-sqrt(N[k])
        plot(1:N[k],l*cumsum(X[1:N[k]]),type="l",
```

```
            col=1,lwd=2,xlim=c(1,i),ylim=c(-i,i),
            ylab="walk",xlab="time")
    }
   Sys.sleep(1)
    lines((N[k]:N[k+1]),
          l*cumsum(X[1:N[k+1]])[N[k]:N[k+1]],col=2,lwd=2)
    Sys.sleep(1)
    for(i in seq(N[k],N[k+1],length=100))
    {
        plot((1 : N[k+1]),
            sqrt(i)*cumsum(X[ 1 : N[k+1]]),
            type="l",
            col=1,lwd=2,xlim=c(1,N[k+1]),ylim=c(-N[k+1],N[k+1]),
            ylab="walk",
            xlab="time")
    }
}
```

Forecasting

Non-stationary sequences

Observational series which desribe phenomena changing with time may be roughly classified in two broad categories, viz. evolutive and stationary In the former case, different sections of the time series are dissimilar in one or more respects. For instance, the sectional averages may be distinctly different, or some other structural property of the series may present variation. al state of the rest./

(Herman Wold, 1938 Uppsala)

Forecasting

The `forecast` package has a range of functions that can be used to forecast values of time series from historical data. In particular, the function `forecast` can be used to forecast values of time series using the Box-Jenkins method for ARIMA models,

```
fcast_bj <- forecast(mod, h=10)
fcast_bj
```

Explain the output after running the command above. Read the help file of `forecast` to see more information about its arguments.

Similarly, the functions `ses` and `holt` are used to forecast values using exponential smoothing and Holt's method respectively. Consider for example forecasts based on exponential smoothing. In the lecture notes we saw that the one step ahead forecast is

$$X_{n+1}^n = (1-\theta)X_n + \theta X_n^{n-1}.$$

```
fcast_es <- ses(LakeHuron, h=10)
fcast_es
```

Exponential smoothing takes the step ahead forecast to be the same as the 1-step ahead forecast. The ses function uses the optimal smoothing parameter to minimise the squared errors in the one-step ahead forecast. Note that $\theta \simeq 0$ is estimated as the optimum smoothing parameter for this dataset ($\alpha = 1-\theta$) in the notation used in the notes)

```
names(fcast_es)
fcast_es$model
```

This means that the 1-step ahead forecast is equal to the last value in the series.

The steps for obtaining forecasts using Holt's method are the same with those of exponential smoothing. Try to obtain forecasts using Holt's method for the Lake Huron data. Why are these so similar to the forecasts obtained using exponential smoothing?

Plotting forecasts using `forecast`

Plots of time series with forecasts are easily obtained using the `plot` function (method `plot.forecast*`). Below are three examples

```
## LakeHuron data
par(mfrow=c(1,2))
plot(fcast_bj)
plot(fcast_es)
```

```
## airmiles data: The revenue passenger miles flown by commercial
## airlines in the United States for each year from 1937 to 1960.
fcast <- holt(airmiles)
plot(fcast)
```

```
## USAccDeaths data: Time series of monthly totals of accidental
## deaths in the USA. Here a modification of Holt's method is used
## to account for the seasonal pattern.
deaths.fcast <- hw(USAccDeaths,h=48)
plot(deaths.fcast)
```

Exercise: Time series in the Environment

Consider the Mauna Loa CO2 time series and discuss appropriate methods for forecasting. How have the data been collected and what type of methods have been employed in the preprocessing? Write a short report about available sources of information that can be obtained from NOAA (I have not checked recently, the dataset might have been censored) and consider sketching time series analysis and methods that can be used to answer a hypothetical problem (e.g. precipitation, wind pressure, accumulation of risk)

```
library(lubridate)
url <- "ftp://aftp.cmdl.noaa.gov/products/trends/co2/co2_mm_mlo.txt"
if (! file.exists("co2new.dat") ||
    now() > file.mtime("co2new.dat") + weeks(4))
    download.file(url, "co2new.dat")
co2new <- read.table("co2new.dat")

names(co2new) <- c("year", "month", "decimal_data", "average",
                   "interpolated", "trend", "ndays")

co2new <- mutate(co2new, average = ifelse(average < -90, NA, average))

plot(co2new$average)
```

Seasonal-Trend decomposition

Seasonal-Trend decomposition using loess

Here, we will look at Seasonal-Trend Decomposition based on loess (STL), yet another useful time series decomposition method. The procedure was developed by Cleveland et al. and is based on the additive decomposition model where

$$Y_t = T_t + S_t + R_t$$

where T_t denotes the trend component, S_t the periodic component and R_t the remainder. STL consists of a sequence of smoothing operations each of which, with one exception, employs the same smoother: locally weighted regression (or loess). The method requires several parameters to be specified by the user (see Section 3 in the original paper).

For measurements x_i and \$y_i for $i = 1, \ldots, n$ on an independent variable and a dependent variable, respectively, the loess regression $\hat{g}(x)$, $x \in \mathcal{X}$, is a smoothing of y given x that is computed in the following way. Here $\mathcal{X}$ denotes the range of the independent variable. A positive integer $q \leq n$ is chosen. The values q of the x_i that are chosen that are closest to x are given a weight based on their distance from x. The neighbourhood weight for any x_i is

$$K_i(x) = W\left(\frac{x_t - x}{\lambda_q(x}\right)$$

where $\lambda_q(x)$ is the distance of the qth farthest x and $W(x) = (1 - |x|^3)^3_+$ is known as the tri-cube function. Here $x_+ = max(x, 0)$ which imples that $\{x \in \mathbb{R}, W(x) > 0\} = (-1, 1)$. In this way a local neighbourhood is specified by a *kernel function* K_i which assigns weights to points $x.$ in a region around x. The closest the points the largest the weight. The distance becomes 0 af the qth furthest point.

In general, we can define a local regression estimate of g as $g_{\hat{\theta}}$ where $\hat{\theta}$ minimises

$$\text{RSS}(f_\theta, x) = \sum_{i=1}^{n} K_i(x)(y_i - f_\theta(x_i))^2$$

and f_θ is some parametrised function such as a low order polynomial, for example!

- $f_\theta(x) = \theta_0$, leading to the Nadaraya-Watson estimate;
- $f_\theta(x) = \theta_0 + x\theta_1$, a local linear regression model.

And important point here is that the regression curve can be computed for any valye of x along the scale of the independent variable. In practice, this is performed by minimizing RSS on a dense grid of values in $\mathcal{X}$, i.e.

$$\left\{\min_{\theta(x)} \text{RSS}(f_{\theta(x)}, x) : x \in \text{ grid } \subset \mathcal{X}\right\}.$$

Hastie et al. Section 6.1.2 explain the choice of the order of the polynomial in the parametrised model f_θ Summarising

-Local linear fits can help bias dramatically at the boundaries at a modest cost in variance. Local quadratic fits do little at the boundaries for bias, but increase tha variance a lot; -Local quadratic fits tend to be most helpful in reducing bias due to curvature in the interior of the domain; -Asymptotic analysis suggest that local polynomials of odd degree dominate those of even degree. This is largely due to the fact that asymptotically, the mean square error is dominated by boundary effects.

STL consists of two recursive procedures: an inner loop nested inside an outer loop. In each of the passes through the inner loop, the seasonal and trend components are updated once; each complete run of the inner loop consists of

such passes. Each pass of the outer loop consists of the inner loop followed by a computation of robustness weights; the weights are used in the next run of the inner loop to reduce the influence of transient, aberrant behaviour on the trend and seasonal components.

R has a built-in function `stl` which implements the STL method. The additional argument `s.window` controls the smoothness of the seasonal component and can be set to `"periodic"` for ts objects with a periodic component. Below is an example of the STL decomposition to the `co2` data.

```
tsdec <- stl(co2,s.window="periodic")
plot(tsdec)
```

Further Topics

Weakly mixing process

The material in this section is taken from Leadbetter et al. (1983)

There are various ways in which the concept of an i.i.d. sequence can be generalised by allowing dependence, or allowing the X_t to have different distributions, or both. For a stationary sequence X , a commonly used restriction is that of strong mixing (Rosenblatt, 1956). A stationary time series X is said to satisfy the strong mixing assumption if there exists a mixing function $g(k)$, tending to zero as $k \to \infty$, and such that

$$|P(A \cup B) - P(A)P(B)| < g(k)$$

when $A \in \sigma(X_1, \dots, X_p)$, and $B \in \sigma(X_{p+k+1}, X_{p+k+2}, \dots)$, where $\sigma(X_i)$ represent the sigma-algebra generated by the random variables (X_i). In other words, if the sequence is mixing, any even A based on the past up to time p is "almost independent" from any event B based on the future starting at time $p + k + 1$ when k is large. The mixing condition is uniform, that is g does not depend on A or B.

The correlation between X_i and X_j is also a partial measurement of the dependency and thus the mixing, leading to another restriction of a similar type:

$$|cor(X_i, X_j)| < g(k),$$

where g goes to 0 at infinity. This restriction is particularly useful if X is a Gaussian Process.

Theorem (Normal comparison lemma): suppose $\mathbf{X} = (X_1, \dots, X_n)$ is a random vector that follows a standard multivariate normal distribution with covariance matrix $\Sigma^X = (\sigma^X_{ij})$ and $\mathbf{Y} = (\mathbf{Y_1}, \dots, \mathbf{Y_n})$ with covariance Σ^Y and let

$$\rho_{ij} = \max(|\sigma^X_{ij}|, |\sigma^Y_{ij}|),$$

and $x_1, \dots, x_n \in \mathbf{R}$, then:

$$P(X_j \le x_j,\ \forall j) - P(Y_j \le x_j,\ \forall j) \le \frac{1}{2\pi} \sum_{i \le i \le j \le n} (\sigma^X_{ij} - \sigma^Y_{ij})_+ (1 - \rho^2_{ij})^{-1/2} \exp\left(-\frac{(x_i^2 + x_j^2)/2}{1 + \rho_{ij}}\right).$$