UiO : Matematisk institutt

Det matematisk-naturvitenskapelige fakultet

STK-4051/9051 Computational Statistics Spring 2024

Instructor: Odd Kolbjørnsen, oddkol@math.uio.no

Exercise 1
Consider first the standard Weibull distribution with density function

$$f_0(x_0) = \alpha x_0^{\alpha-1} e^{-x_0^\alpha}$$

and cummulative distribution function

$$F_0(x_0) = 1 - e^{-x^\alpha}.$$

(a) Explain how the inversion method can be used to generate samples from f_0.

Exercise 1 (a) The inversion method is to generate $X = F_0^{-1}(U)$ where $U \sim \text{Uniform}[0,1]$. We have

$$\begin{aligned} 1 - e^{-x^\alpha} =& u \\ &\Updownarrow \\ x^\alpha =& -\log(1-u) \\ &\Updownarrow \\ x =& [-\log(1-u)]^{1/\alpha} \end{aligned}$$

Consider now the general Weibull distribution with density function

$$p(x) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^{\alpha}}$$

(b) Show that if x_0 has a standard Weibull density then $x = \beta x_0$ has a general Weibull density. Discuss how this result can be used to generate random variables from the general Weibull distribution.

(b) We have $x_0 = x/\beta$ giving that

$$\begin{aligned} f(x) =& f_0(x/\beta)/\beta \\ =& \alpha(x/\beta)^{\alpha-1} e^{-(x/\beta)^{\alpha}}/\beta \\ =& \frac{\alpha x^{\alpha-1}}{\beta^{\alpha}} e^{-(x/\beta)^{\alpha}} \end{aligned}$$

showing the result. We can then generate x by

$$x = \beta[-\log(1-u)]^{1/\alpha}$$

Assume now we want to generate two dependent random variables that have marginal distributions that are of the Weibull form. Direct specification of dependence for the Weibull distribution can be difficult, but can be greatly simplified through transformation (this is called a copula approach in the literature).

(c) Let $\Phi()$ be the cummulative distribution function for the standard Normal distribution. Show that if $y \sim N(0,1)$, then

$$x = F_0^{-1}(\Phi(y))$$

has a standard Weibull distribution.

$$\boxed{\begin{aligned}\Pr(X \le x) &= \Pr(F_0^{-1}(\Phi(Y)) \le x)\\ &= \Pr(\Phi(Y) \le F_0(x))\\ &= \Pr(Y \le \Phi^{-1}(F_0(x))\\ &= \Phi(\Phi^{-1}(F_0(x))) = F_0(x)\end{aligned}}$$

(d) Assume now that you are able to simulate $\boldsymbol{y} = (y_1, y_2)$ from a bivariate Normal distribution $N(\mathbf{0}, \boldsymbol{\Sigma})$ where $\Sigma_{1,1} = \Sigma_{2,2} = 1$ and $\Sigma_{1,2} = \Sigma_{2,1} = \rho$. Explain how you can use this to simulate two dependent Weibull distributed variables.

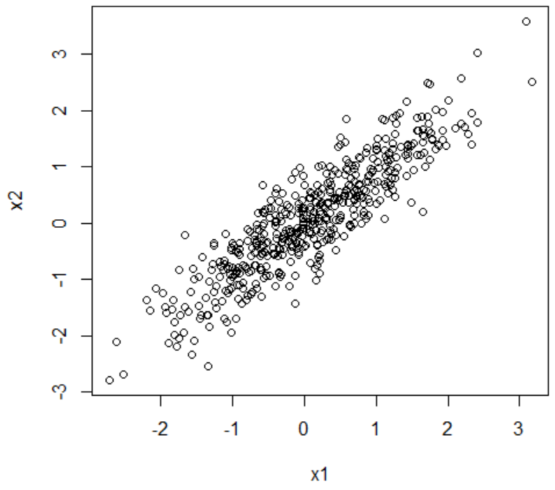
(d) We can then put

$$x_j = F_0^{-1}(\Phi(y_j)).$$

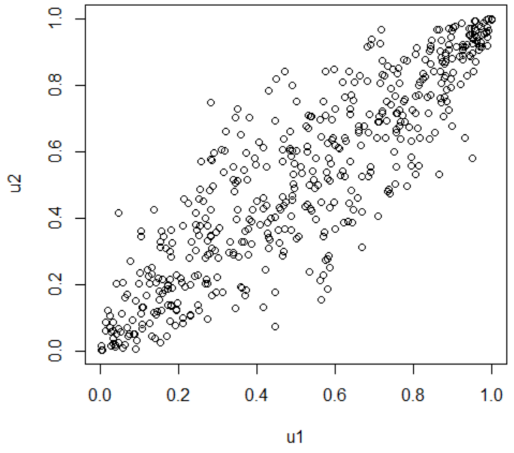
Since (y_1, y_2) are dependent, so will (x_1, x_2) be.

As transformations are monotone, the rank correlation will be preserved

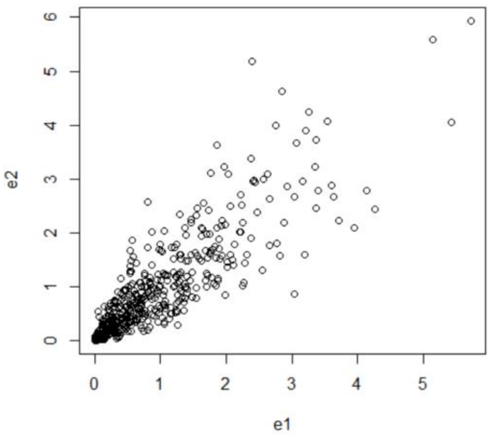
Normal



Uniform



Weibull(1,1)
=exponential



```
N=500

n1=rnorm(N,0,1)
n2=rnorm(N,0,1)

rho=0.9
x1=n1
x2=x1*rho+sqrt(1-rho^2)*n2

u1=pnorm(x1)
u2=pnorm(x2)

e1 =-log(u1)
e2 =-log(u2)

hist(e1)


plot(x1,x2)
plot(u1,u2)
plot(e1,e2)
```

Exercise 2

Consider the following algorithm, which we will call Barker's algorithm (after Barker (1965) who suggested it):
Given the current state $\boldsymbol{x}^{(t)}$:

- Draw $\boldsymbol{y}$ from the proposal distribution $K(\boldsymbol{x}^{(t)}, \boldsymbol{y})$ (or transition kernel).
- Draw $U \sim \text{Uniform}[0,1]$ and update

$$\boldsymbol{x}^{(t+1)} = \begin{cases} \boldsymbol{y}, & \text{if } U \leq r_B(\boldsymbol{x}^{(t)}, \boldsymbol{y}) \\ \boldsymbol{x}^{(t)} & \text{otherwise} \end{cases}$$

  where

$$r_B(\boldsymbol{x}, \boldsymbol{y}) = \frac{\pi(\boldsymbol{y})K(\boldsymbol{y}, \boldsymbol{x})}{\pi(\boldsymbol{x})K(\boldsymbol{x}, \boldsymbol{y}) + \pi(\boldsymbol{y})K(\boldsymbol{y}, \boldsymbol{x})}.$$

We will assume that $K(\boldsymbol{x}, \boldsymbol{y}) > 0$ for all $\boldsymbol{x}, \boldsymbol{y}$.

$$r_B(\boldsymbol{x},\boldsymbol{y}) = \frac{\pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})}{\pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y}) + \pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})}.$$

We will assume that $K(\boldsymbol{x},\boldsymbol{y}) > 0$ for all $\boldsymbol{x},\boldsymbol{y}$.

(a) Show that $\{\boldsymbol{x}_t\}$ is a Markov chain with invariant distribution $\pi(\boldsymbol{x})$.

Exercise 2 (a) We will show that the Markov chain satisfies the detailed balance criterion. We have for $\boldsymbol{x} \neq \boldsymbol{y}$

$$\begin{aligned}\pi(\boldsymbol{x})P(\boldsymbol{x},\boldsymbol{y}) =& \pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y})r_B(\boldsymbol{x},\boldsymbol{y})\\ =& \pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y})\frac{\pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})}{\pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y}) + \pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})}\\ =& \pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})\frac{\pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y})}{\pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y}) + \pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})}\\ =& \pi(\boldsymbol{y})P(\boldsymbol{y},|\boldsymbol{x}).\end{aligned}$$

(b) Explain how we can used the simulations $\{\boldsymbol{x}^{(t)}\}$ to estimate $E_\pi h = \int_{\boldsymbol{x}} h(\boldsymbol{x})\pi(\boldsymbol{x})d\boldsymbol{x}$.

What kind of properties of the Markov chain will influence on the precision of such an estimate?

(b) If we simulate $\{\boldsymbol{x}^t\}$ according to the described Markov chain, we have from general theory that we can estimate $\mu = E_\pi[h(\boldsymbol{x})]$ by

$$\hat{\mu} = \frac{1}{L}\sum_{t=D+1}^{D+L} h(\boldsymbol{x}^t)$$

where we discard the first D samples in order to minimized the bias due to that it can take some time until the samples are close enough to the target distribution. We further have

$$\begin{aligned}\text{Var}[\hat{\mu}] =& \frac{1}{L^2}[\sum_{t=D+1}^{D+L}\text{Var}[h(\boldsymbol{x}^t)] + 2\sum_{s=D+1}^{D+L-1}\sum_{t=s+1}^{D+L}\text{Cov}[h(\boldsymbol{x}^s), h(\boldsymbol{x}^t)]\\ \approx& \frac{\sigma_h^2}{L}[1 + 2\sum_{t=D+1}^{D+L-1}\rho(t-s)]\end{aligned}$$

showing the dependence on the correlation structure.

Assume A_1 and A_2 are two transition-kernels for Markov chains with the same stationary distribution π. Let v_1 be the variance of the estimate on $E_\pi h$ based on simulations using A_1 and v_2 the variance of the estimate of $E_\pi h$ using A_2.
Assume $A_1(\boldsymbol{x},\boldsymbol{y}) \geq A_2(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{y} \neq \boldsymbol{x}$. One can then show that $v_1 \leq v_2$ (this you do not have to prove).

(c) Let

$$r_M(\boldsymbol{x},\boldsymbol{y}) = \min\left\{1, \frac{\pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})}{\pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y})}\right\}.$$

Show that $r_M(\boldsymbol{x},\boldsymbol{y}) \geq r_B(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x},\boldsymbol{y}$.

(c) Assume $\pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x}) \geq \pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y})$. Then $r_M(\boldsymbol{x},\boldsymbol{y}) = 1 > r_B(\boldsymbol{x},\boldsymbol{y})$. Assume now $\pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x}) < \pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y})$. Then

$$r_M(\boldsymbol{x},\boldsymbol{y}) = \frac{\pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})}{\pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y})} \geq \frac{\pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})}{\pi(\boldsymbol{x})K(\boldsymbol{x},\boldsymbol{y}) + \pi(\boldsymbol{y})K(\boldsymbol{y},\boldsymbol{x})} = r_M(\boldsymbol{x},\boldsymbol{y})$$

showing the first result.

Show that $r_M(\boldsymbol{x},\boldsymbol{y}) \geq r_B(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x},\boldsymbol{y}$.

Use this to argue that the Metropolis-Hastings algorithm is more efficient than Barker's algorithm.

Based on the differences between these two algorithms, do you think this is a reasonable result?

Since we have proven that: $r_M(\boldsymbol{x},\boldsymbol{y}) \geq r_B(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x},\boldsymbol{y}$.

We have proven: $P_M(\boldsymbol{x},\boldsymbol{y}) \geq P_B(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{x} \neq \boldsymbol{y}$

$\boldsymbol{x} \neq \boldsymbol{y}$

$$P_M(\boldsymbol{x},\boldsymbol{y}) = K(\boldsymbol{x},\boldsymbol{y})r_M(\boldsymbol{x},\boldsymbol{y})$$

$$P_B(\boldsymbol{x},\boldsymbol{y}) = K(\boldsymbol{x},\boldsymbol{y})r_B(\boldsymbol{x},\boldsymbol{y})$$

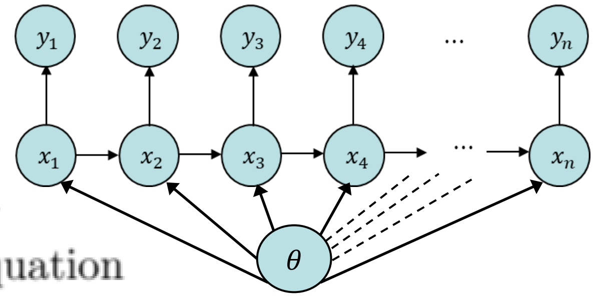
The result then apply:

Assume A_1 and A_2 are two transition-kernels for Markov chains with the same stationary distribution π. Let v_1 be the variance of the estimate on $E_\pi h$ based on simulations using A_1 and v_2 the variance of the estimate of $E_\pi h$ using A_2.
Assume $A_1(\boldsymbol{x},\boldsymbol{y}) \geq A_2(\boldsymbol{x},\boldsymbol{y})$ for all $\boldsymbol{y} \neq \boldsymbol{x}$. One can then show that $v_1 \leq v_2$ (this you do not have to prove).

Both algorithms are using the same proposals and both have the same invariant distribution. Since M-H give higher acceptance probabilities, the changes should happen more frequent and thereby give a more efficient algorithm.

Exercise 4

Consider the following state space model:



$$
\begin{aligned}
x_t &= \phi x_{t-1} + \varepsilon_t && \text{state equation} \\
y_t &\sim \text{Poisson}(\exp\{1 + x_t\}) && \text{observation equation}
\end{aligned}
$$

where x_0 and $\varepsilon_1, \varepsilon_2, \ldots$ are independent and standard normal distributed. We want to estimate ϕ based on observations $y_1, \ldots, y_T$. We will do this in a Bayesian way and assume we have a prior distribution $N(0, \sigma_\phi^2)$ on ϕ.

A possible way to estimate ϕ in such situations is to extend the state model to the following model:

$$
\begin{aligned}
\phi_t &= \phi_{t-1} && \text{state equation 1} \\
x_t &= \phi_{t-1} x_{t-1} + \varepsilon_t && \text{state equation 2} \\
y_t &\sim \text{Poisson}(\exp\{1 + x_t\}) && \text{observation equation}
\end{aligned}
$$

where $\phi_0 \sim N(0, \sigma_\phi^2)$. Non-linear filters try to compute the posterior distribution for (ϕ_t, x_t) based on $y_1, \ldots, y_t$. Since $\phi = \phi_T$, the posterior distribution for (ϕ_T, x_T) based on $y_1, \ldots, y_T$ gives us the posterior distribution for ϕ given $y_1, \ldots, y_T$.

Simulation methods for non-linear filters can therefore be used on the bivariate state vector (ϕ_t, x_t).

(a) Explain the general principles behind sequential importance sampling (SIS).

Discuss why resampling in general is important in connection to SIS algorithms.

Exercise 4 (a) The general idea is to simulate $(\phi, x_1, ..., x_t)$ by a proposal distribution $q_\phi(\phi)q_1(x_1|\phi)\prod_{i=2}^t q_i(x_i|x_{i-1},\phi)$ and then use the importance sampling technique to get importance weights

$$\begin{aligned}
w_t &= \frac{p(\phi, x_1, ..., x_t|y_1, ..., y_t)}{q(\phi, x_1, ..., x_t)} \\
&\propto \frac{p(\phi)p(x_1, ..., x_t|\phi)p(y_1, ..., y_t|x_1, ..., x_t, \phi)}{q(\phi, x_1, ..., x_t)} \\
&= \frac{p(\phi)p(x_1|\phi)\prod_{i=2}^t p(x_i|x_{i-1},\phi)\prod_{i=1}^t p(y_i|x_i)}{q_\phi(\phi)q_1(x_1|\phi)\prod_{i=2}^t q_i(x_i|x_{i-1},\phi)} \\
&\propto w_{t-1}\frac{p(x_t|x_{t-1},\phi)p(y_t|x_t)}{q(x_t|x_{t-1},\phi)}
\end{aligned}$$

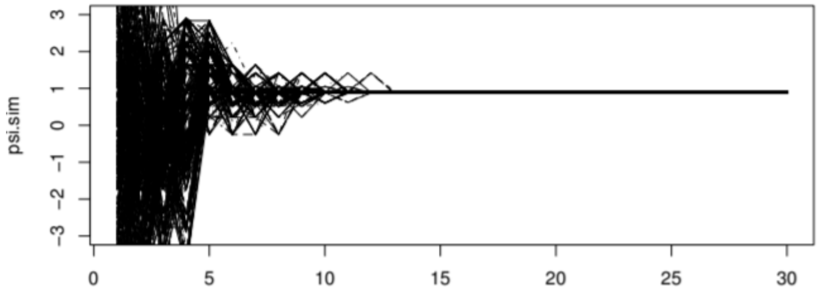
showing that the weights can be calculated recursively.

Due to that the variance of the weights will increase with t, a degeneracy problem occur. This can be fixed by performing resampling at each step (or when the efficient sample size is small).

(b) Simulations from the posterior distribution for (ϕ_t, x_t) based on $y_1, ..., y_t$ was performed through the following SIS algorithm:

- Draw $\tilde{x}_t^j$ from $N(\phi_{t-1}^j x_{t-1}^j, 1)$ for $j = 1, ..., M$
- Put $\tilde{\phi}_t^j = \phi_{t-1}^j$ for $j = 1, ..., M$.
- Calculate the weights $w_t^j = p(y_t|x_t = \tilde{x}_t^j)$ for $j = 1, ..., M$ and the normalized weights $q_t^j = w_t^j / \sum_{j'} w_t^{j'}$.
- Draw $(x_t^1, \phi_t^1), ..., (x_t^M, \phi_t^M)$ from $\{(\tilde{x}_t^1, \tilde{\phi}_t^1), ..., (\tilde{x}_t^M, \tilde{\phi}_t^M)\}$ with replacement and with probabilities $q_t^1, ..., q_t^M$.

The figure below shows simulations of ϕ_t for $t = 1, ..., T$ based on a SIS algorithm with resampling. Each curve corresponds to a sequence of simulated ϕ's, $\phi_1^j, ..., \phi_T^j$. The different simulations $\phi_t^j, j = 1, ..., M$ for a fixed t are (approximately) from the posterior distribution for ϕ_t given $y_1, ..., y_t$. Here $T = 30$ and $M = 50$.



Why do the number of different values of the simulated ϕ's decrease with t? What kind of problems do this make in the estimation of ϕ?

(b) The resampling step will result in that fewer and fewer unique values of ϕ will occur, in the end only one. This causes problems in estimation of ϕ due to that we then effectively only have one sample for describing the whole distribution of ϕ.

(c) A more efficient algorithm can be obtained by integrating out the unknown ϕ when simulating the x-process.

One can show (you do not have to do this) that

$$p(\phi|x_1,...,x_t,y_1,...,y_t) = N(\hat{\phi}_t, \sigma_t^2)$$

where

$$\hat{\phi}_t = \frac{\sigma_\phi^2 \sum_{i=2}^t x_i x_{i-1}}{1+\sigma_\phi^2 \sum_{i=2}^t x_{i-1}^2}, \quad \sigma_t^2 = \frac{\sigma_\phi^2}{1+\sigma_\phi^2 \sum_{i=2}^t x_{i-1}^2}$$

Use this to explain how you can simulate from the distribution $p(x_{t+1}|x_1,...,x_t,y_1,...,y_t)$.

(c) Assume you have a properly weighted sample $\{(x_t^i, \boldsymbol{S}_t^i, w_t^i), i = 1, ..., M\}$ with respect to $p(x_t, \boldsymbol{S}_t|y_1,...,y_t)$ where $\boldsymbol{S}_t^i$ are the sufficient statistics needed for calculating the distribution $p(\phi|x_1^i,...,x_t^i,y_1,...,y_t)$. The idea is then to update to a properly weighted sample $\{(x_{t+1}^i, \boldsymbol{S}_{t+1}^i, w_t^i), i = 1,...,M\}$ with respect to $p(x_{t+1}, \boldsymbol{S}_{t+1}|y_1,...,y_{t+1})$.

We have

$$p(x_t, S_t|\boldsymbol{y}_{1:t-1}) = \int_{x_{t-1}} p(x_t, S_t|x_{t-1}, S_{t-1}) p(x_{t-1}, S_{t-1}|\boldsymbol{y}_{1:t-1}) dx_{t-1} dS_{t-1}$$
$$\approx \sum_{i=1}^N w_{t-1}^i p(x_t, S_t|x_{t-1}^i, S_{t-1}^i)$$
$$p(x_t, S_t|\boldsymbol{y}_{1:t}) \approx c \cdot \sum_{i=1}^N w_{t-1}^i p(x_t, S_t|x_{t-1}^i, S_{t-1}^i) p(y_t|x_t).$$

Simulation from $p(x_t, S_t|x_{t-1}^i, S_{t-1}^i)$ (possible proposal function)

(a) Simulate $\theta^i \sim p(\theta|x_{t-1}^i, S_{t-1}^i) = p(\theta|S_{t-1}^i)$.

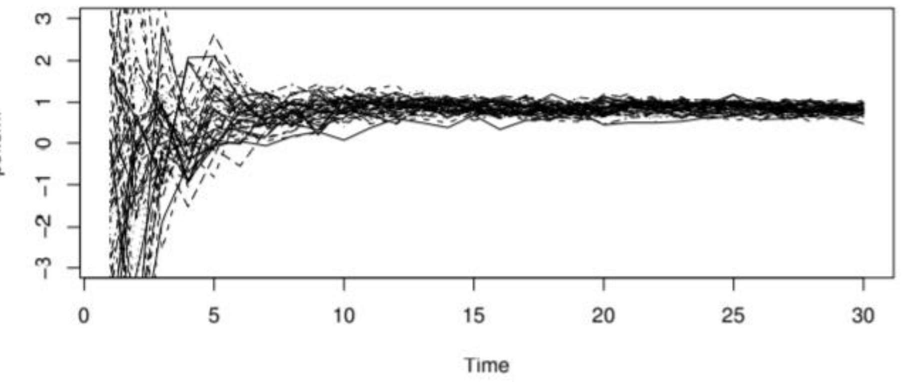
(b) Simulate $x_t^i \sim p(x_t|x_{t-1}^i, \theta^i)$.

(c) Update sufficient statistics

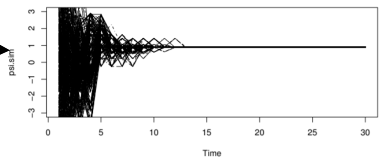
Consider now the SIS algorithm (with resampling) which at time t goes through the following steps:

- Draw $\tilde{x}_t^j$ from $p(x_t|x_1, ..., x_{t-1}, y_1, ..., y_{t-1})$ for $j = 1, ..., M$.
- Calculate the weights $w_t^j = p(y_t|x_t = \tilde{x}_t^j)$ for $j = 1, ..., M$ and the normalized weights $q_t^j = w_t^j / \sum_{j'} w_t^{j'}$.
- Draw $x_t^1, ..., x_t^M$ from $\{\tilde{x}_t^1, ..., \tilde{x}_t^M\}$ with replacement and the probabilities $q_t^1, ..., q_t^M$.
- Draw $\phi_t^j \sim p(\phi|x_1^j, ..., x_t^j, y_1, ., y_t)$

The figure below shows simulations of ϕ_t based on this algorithm.



Which advantages does this algorithm have compared to the one given in (b)?



In order to estimate the posterior expectation of ϕ, is it necessary to simulate the ϕ's at all? If not, explain how inferense on ϕ then can be performed. What is this technique called?

(d) By turning the simulation from the static parameter ϕ to the random variable $\boldsymbol{S}_t$, we reduce the degeneracy problem and obtain a more reliable description of the distribution for x_t and ϕ as well.

In order to estimate ϕ, we can use Rao-Blackwellization in that

$$E[\phi|y_1, ..., y_t] = E[E[\phi|\boldsymbol{S}_t]|y_1, ..., y_t] \approx \frac{1}{M}\sum_{j=1}^{M} E[\phi|S_t^j]$$

where we now have explicit solutions for the inner expectation.

Exercise 5 (Weight loss programme)
Venables and Ripley [1999] contain a dataset (originally from Dr. T Davies) describing weights (y_i) of obese patients after different number of days (x_i)

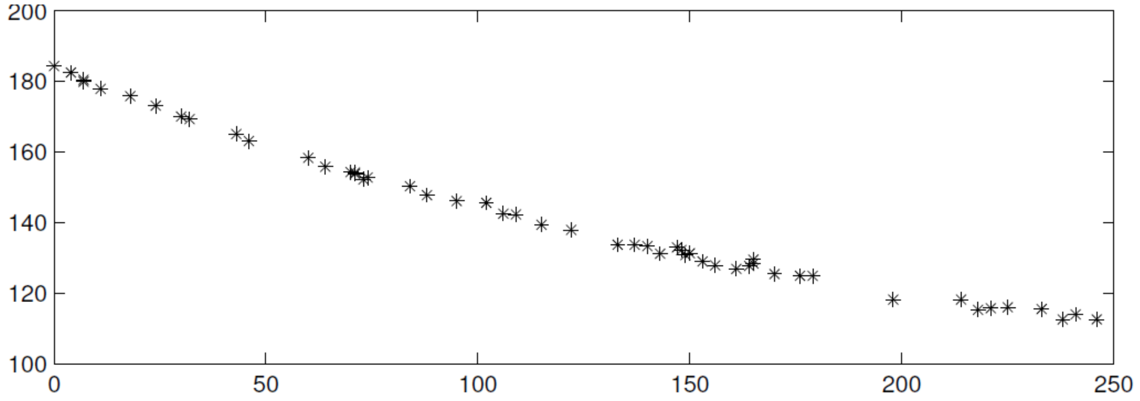


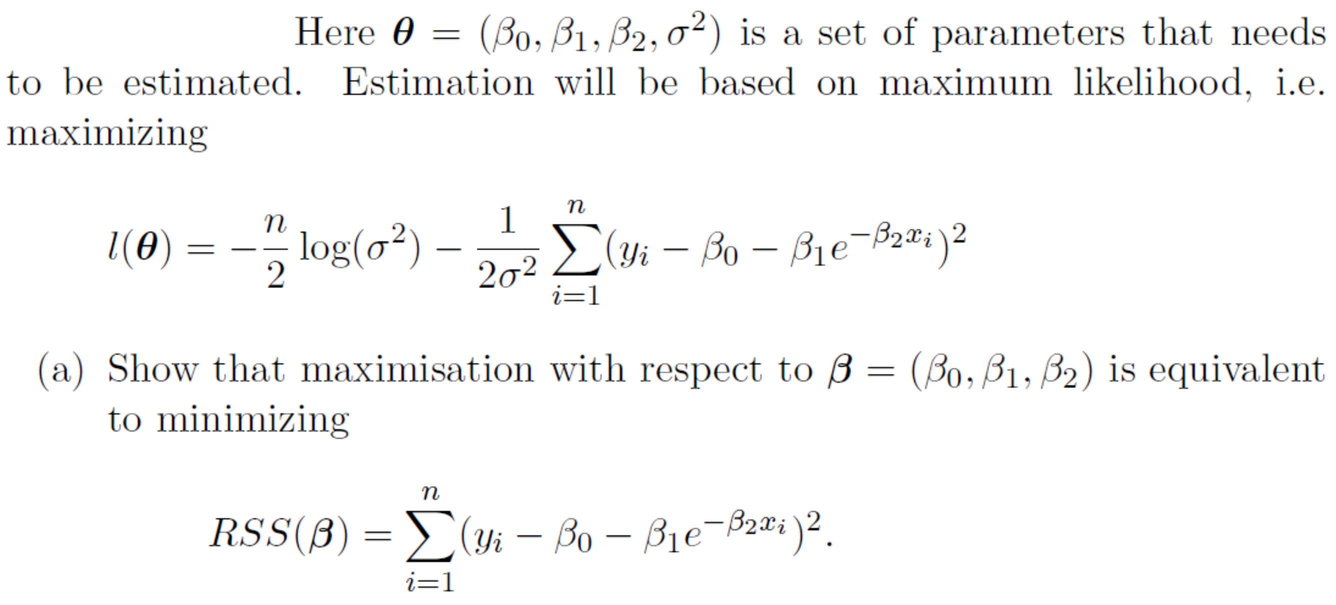
Figure 1: Weight loss from an obese patient

since start of a weight reduction programme. The data is plotted in Figure 1. Venables and Ripley [1999] suggests the following model for this dataset:

$$y_i = \beta_0 + \beta_1 e^{-\beta_2 x_i} + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$$

for $i = 1, ..., n$. Here $\boldsymbol{\theta} = (\beta_0, \beta_1, \beta_2, \sigma^2)$ is a set of parameters that needs to be estimated. Estimation will be based on maximum likelihood, i.e. maximizing

$$l(\boldsymbol{\theta}) = -\frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})^2$$

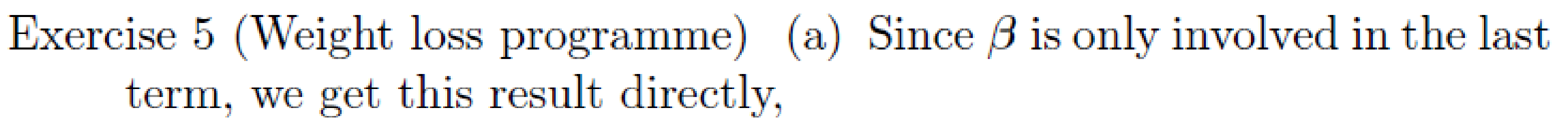


Here $\boldsymbol{\theta} = (\beta_0, \beta_1, \beta_2, \sigma^2)$ is a set of parameters that needs to be estimated. Estimation will be based on maximum likelihood, i.e. maximizing

$$l(\boldsymbol{\theta}) = -\frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})^2$$

(a) Show that maximisation with respect to $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$ is equivalent to minimizing

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})^2.$$



Exercise 5 (Weight loss programme) (a) Since $\boldsymbol{\beta}$ is only involved in the last term, we get this result directly,

When you are asked to show something use:

- Arguments
- Computations

If you are not able to do so you might get points if you can provide insight to the problem

$$l(\boldsymbol{\theta}) = -\frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})^2$$

What is the optimal value for σ^2 given β?

We have that

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})^2$$

Putting this to zero, we obtain

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})^2.$$

One can also show that the second derivative becomes positive, showing that it is a maximum point. This shows that for given $\hat{\beta}$ we have an explicit solution for $\hat{\sigma}^2$.

(b) Describe Newton's method and perform the calculations needed to implement this algorithm.

(b) Assume one wants to minimize $g(\boldsymbol{\theta})$. Newton's method:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta} - [g''(\boldsymbol{\theta}^t)]^{-1} g'(\boldsymbol{\theta}^t).$$

In this case $\boldsymbol{\theta} = \beta$ and $g(\beta) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})^2$. We have

$g'(\boldsymbol{\theta}^t)$.

$$\frac{\partial g(\beta)}{\partial \beta_0} = -2\sum_{i=1}^n (y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})$$

$$\frac{\partial g(\beta)}{\partial \beta_1} = -2\sum_{i=1}^n (y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i}) e^{-\beta_2 x_i}$$

$$\frac{\partial g(\beta)}{\partial \beta_2} = 2\sum_{i=1}^n (y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i}) \beta_1 e^{-\beta_2 x_i} x_i$$

(b) Describe Newton's method and perform the calculations needed to implement this algorithm.

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta} - [g''(\boldsymbol{\theta}^t)]^{-1} g'(\boldsymbol{\theta}^t).$$

$g''(\boldsymbol{\theta}^t)$

$$\boxed{\frac{\partial g(\beta)}{\partial \beta_0} = -2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})}$$

$$\frac{\partial^2 g(\beta)}{\partial \beta_0 \partial \beta_0} = 2n$$

$$\frac{\partial^2 g(\beta)}{\partial \beta_0 \partial \beta_1} = 2\sum_{i=1}^{n} e^{-\beta_2 x_i}$$

$$\frac{\partial^2 g(\beta)}{\partial \beta_0 \partial \beta_2} = 2\beta_1 \sum_{i=1}^{n} e^{-\beta_2 x_i} x_i$$

(b) Describe Newton's method and perform the calculations needed to implement this algorithm.

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta} - [g''(\boldsymbol{\theta}^t)]^{-1} g'(\boldsymbol{\theta}^t).$$

$g''(\boldsymbol{\theta}^t)$

$$\boxed{\begin{aligned}\frac{\partial g(\beta)}{\partial \beta_1} &= -2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})e^{-\beta_2 x_i}\\ \frac{\partial g(\beta)}{\partial \beta_2} &= 2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})\beta_1 e^{-\beta_2 x_i} x_i\end{aligned}}$$

$$\frac{\partial^2 g(\beta)}{\partial \beta_1 \partial \beta_1} = 2\sum_{i=1}^{n} e^{-2\beta_2 x_i}$$

$$\frac{\partial^2 g(\beta)}{\partial \beta_1 \partial \beta_2} = 2\sum_{i=1}^{n}(y_i - \beta_0 - 2\beta_1 e^{-\beta_2 x_i})e^{-\beta_2 x_i} x_i$$

$$\frac{\partial^2 g(\beta)}{\partial \beta_2 \partial \beta_2} = -2\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})\beta_1 e^{-\beta_2 x_i} x_i^2 + \sum_{i=1}^{n} \beta_1^2 e^{-2\beta_2 x_i} x_i^2$$

For reference, the following results where obtained using Newton's method in this case. Note however that for small perturbations of these starting values, numerical problems occured.

Iteration s	$\beta_0^{(s)}$	$\beta_1^{(s)}$	$\beta_2^{(s)}$	$(\sigma^2)^{(s)}$	$l^{(s)})$
0	90.000	95.000	0.0050000	209.386	-143.259
1	84.339	100.551	0.0051991	0.72866	-69.670
2	76.350	107.131	0.0044158	1.47380	-78.827
3	76.801	106.841	0.0045417	0.65652	-68.314
4	81.664	102.393	0.0048807	0.60634	-67.281
5	81.399	102.662	0.0048866	0.56958	-66.468
6	81.374	102.684	0.0048844	0.56958	-66.468
7	81.374	102.684	0.0048844	0.56958	-66.468

(c) A run with Fisher's scoring algorithm with the same starting values as above gave the following results:

Iteration s	$\beta_0^{(s)}$	$\beta_1^{(s)}$	$\beta_2^{(s)}$	$(\sigma^2)^{(s)}$	$l(\boldsymbol{\beta}^{(s)}, (\sigma^2)^{(s)})$
0	90.000	95.00	0.0050000	209.39	-143.259
1	81.400	102.66	0.0048760	0.5758	-66.609
2	81.374	102.68	0.0048844	0.5696	-66.468
3	81.374	102.68	0.0048844	0.5696	-66.468
4	81.374	102.68	0.0048844	0.5696	-66.468

Give a general description of this algorithm and discuss its benefits compared to Newton's method

For the Fisher scoring algorithm, we replace the matrix of second derivatives with their expectation. This guarantees that the matrix becomes positive (semi-)definite
Since this is the variance of the scoring function.

Which in turn stabilize the solution.

(d) Show that given β_2, the maximum values for all the other parameters can be found analytically.

What benefits do this result have with respect to optimisation?

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 e^{-\beta_2 x_i})^2.$$

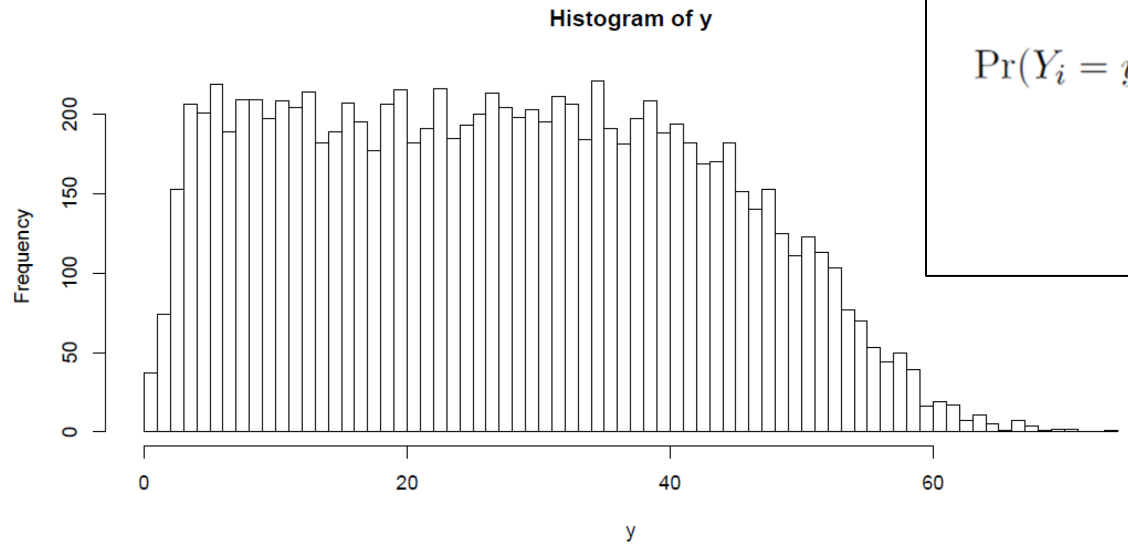
(d) For given β_2, we can define $z_i = e^{-\beta_2 x_i}$ and we then have an ordinary linear regression model with z_i as explanatory variable. We can then use the general results from linear regression.

This means that we can reduce the optimization down to just one variable, simplifying the problem significantly.

Exercise 6
Consider a mixture model where

$$\Pr(C_i = k) = \pi_k$$
$$\Pr(y_i = y | C_i = k) = \frac{\lambda_k^{y_i} e^{-\lambda_k}}{y_i!}$$

Our aim is to obtain maximum likelihood estimates of $\boldsymbol{\theta} = \{(\pi_k, \lambda_k), k = 1, ..., K\}$ based on observations $\boldsymbol{y} = (y_1, ..., y_n)$.
The histogram below shows a simulated dataset with $K = 10$ classes and $n = 10\,000$.



$$\Pr(Y_i = y) = \sum_{k=1}^{K} \Pr(C_i = k) \Pr(Y_i = y | C_i = k)$$
$$= \sum_{k=1}^{K} \pi_k \frac{\lambda_k^{y} e^{-\lambda_k}}{y!}$$

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{K} [\sum_{k=1}^{K} \pi_k \frac{\lambda_k^{y_i} e^{-\lambda_k}}{y_i!}]$$

(a) Write down the likelihood function based on the observations $\boldsymbol{y}$.

A call to a general optimiser using the Nelder-Mead algorithm gave the following estimates:

k	1	2	3	4	5	6	7	8	9	10
$\hat{\pi}_k$	0.107	0.000	0.146	0.164	0.049	0.126	0.004	0.202	0.183	0.020
$\hat{\lambda}_k$	5.18	8.26	11.32	19.26	27.75	28.01	28.27	37.17	47.85	48.83

with a log-likelihood value equal to 40573.98, obtained after 502 function calls.

Describe short the main features of the Nelder-Mead algorithm.

Nelder-Mead: With $\boldsymbol{\theta}$ p-dimensional, we start with $p+1$ values of $\boldsymbol{\theta}$. These $p+1$ values are dynamically altered by changing the worst value with a better one, defined through a search line going through the worst value and the average of the other values. The worst value is then updated to a better (best?) value along this line. The algorithm is performing these steps iteratively until some stopping criterion is achieved. This method does not need the derivatives.

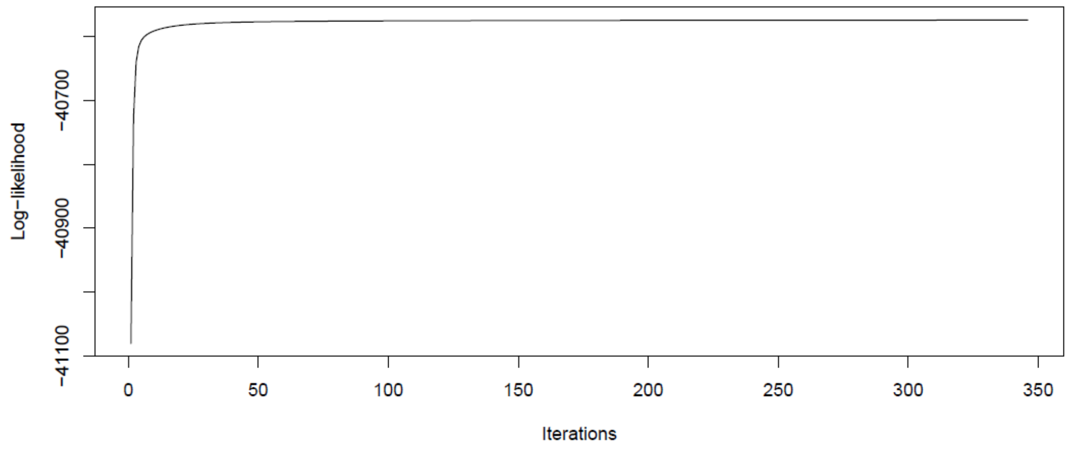
(b) An alternative to a direct optimiser is to use the EM algorithm where we treat $\{c_i\}$ as missing variables. Derive the updating equations for the parameters involved in this case.

The plot below shows the log-likelihood values at different iterations based on the EM algorithm.

Further, the final estimates obtained in this case is given in the table below, obtained after 346 iterations with a final log-likelihood value of 40573.98.

Explain the result in the figure with respect to properties of the EM algorithm.

k	1	2	3	4	5	6	7	8	9	10
$\hat{\pi}_k$	0.110	0.136	0.097	0.107	0.103	0.025	0.149	0.074	0.119	0.081
$\hat{\lambda}_k$	5.24	11.31	17.56	22.30	27.80	31.13	35.04	40.76	46.68	49.32



(b) We have that the complete likelihood is given by

$$\begin{aligned}
L_c(\boldsymbol{\theta}) &= \prod_{i=1}^{n} \pi_{c_i} \frac{\lambda_{c_i}^{y_i} e^{-\lambda_{c_i}}}{y_i!} \\
\ell_c(\boldsymbol{\theta}) &= \log L_c(\boldsymbol{\theta}) \\
&= \sum_{i=1}^{n} [\log \pi_{c_i} + y_i \log \lambda_{c_i} - \lambda_{c_i} - \log y_i!] \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} I(c_i = k)[\log \pi_k + y_i \log \lambda_k - \lambda_k - \log y_i!] \\
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) &= E[\ell_c(\boldsymbol{\theta}) | \boldsymbol{\theta}^t] \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} \Pr(C_i = k | \boldsymbol{\theta}^t)[\log \pi_k + y_i \log \lambda_k - \lambda_k - \log y_i!]
\end{aligned}$$

$$Q_{lagr}(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) + \delta(\sum_{k=1} \pi_k - 1)$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{K}\Pr(C_i = k|\boldsymbol{\theta}^t)[\log\pi_k + y_i\log\lambda_k - \lambda_k - \log y_i!] \; + \delta(\sum_{k=1}\pi_k - 1)$$

$$\frac{\partial}{\partial\pi_k}Q_{lagr}(\boldsymbol{\theta},\boldsymbol{\theta}^t) = \sum_{i=1}^{n}\Pr(C_i = k|\boldsymbol{\theta}^t)\frac{1}{\pi_k} - \delta$$

giving

$$\begin{aligned}\pi_k^t =&\delta^{-1}\sum_{i=1}^{n}\Pr(C_i = k|\boldsymbol{\theta}^t)\\ =&\frac{1}{n}\sum_{i=1}^{n}\Pr(C_i = k|\boldsymbol{\theta}^t)\end{aligned}$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{K} \Pr(C_i = k|\boldsymbol{\theta}^t)[\log \pi_k + y_i \log \lambda_k - \lambda_k - \log y_i!] \ + \delta(\sum_{k=1} \pi_k - 1)$$

$$\frac{\partial}{\partial \lambda_k} Q_{lagr}(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = \sum_{i=1}^{n} \Pr(C_i = k|\boldsymbol{\theta}^t)[\frac{y_i}{\lambda_k} - 1]$$

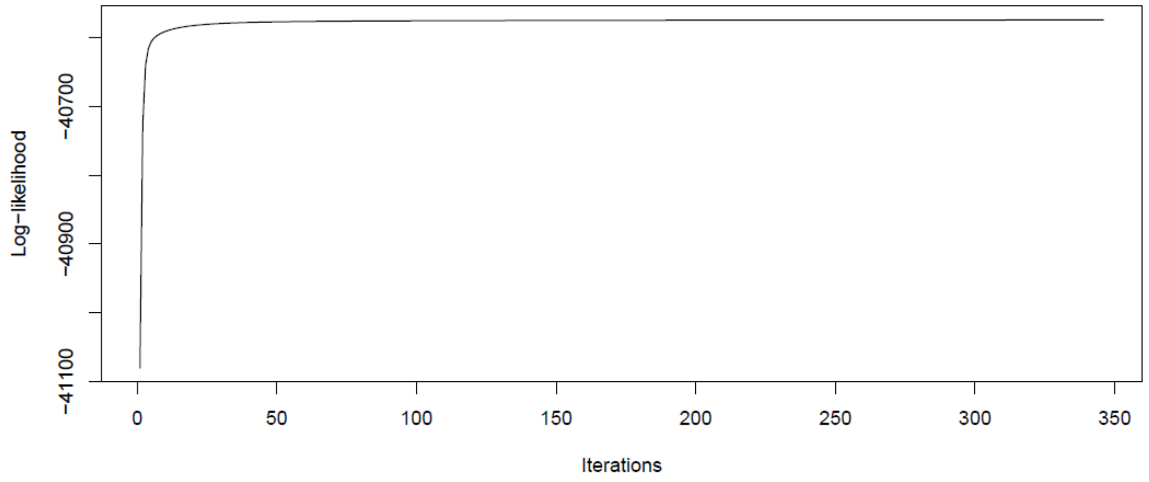
giving

$$\lambda_k^{t+1} = \frac{\sum_{i=1}^{n} \Pr(C_i = k|\boldsymbol{\theta}^t) y_i}{\sum_{i=1}^{n} \Pr(C_i = k|\boldsymbol{\theta}^t)}$$

where the probabilities $\Pr(C_i = k|\boldsymbol{\theta}^t)$ are based on the parameter values from the previous iteration.

Explain the result in the figure with respect to properties of the EM algorithm.

k	1	2	3	4	5	6	7	8	9	10
$\hat{\pi}_k$	0.110	0.136	0.097	0.107	0.103	0.025	0.149	0.074	0.119	0.081
$\hat{\lambda}_k$	5.24	11.31	17.56	22.30	27.80	31.13	35.04	40.76	46.68	49.32



The EM-algorithm has the property that the (log-)likelihood values will never decrease from one iteration to another, which the plot demonstrate.

(c) Comparing the results from the two algorithms, the estimates appears to be quite different. However, the log-likelihood values are quite similar. Try to give an explanation on this.

A call to a general optimiser using the Nelder-Mead algorithm gave the following estimates:

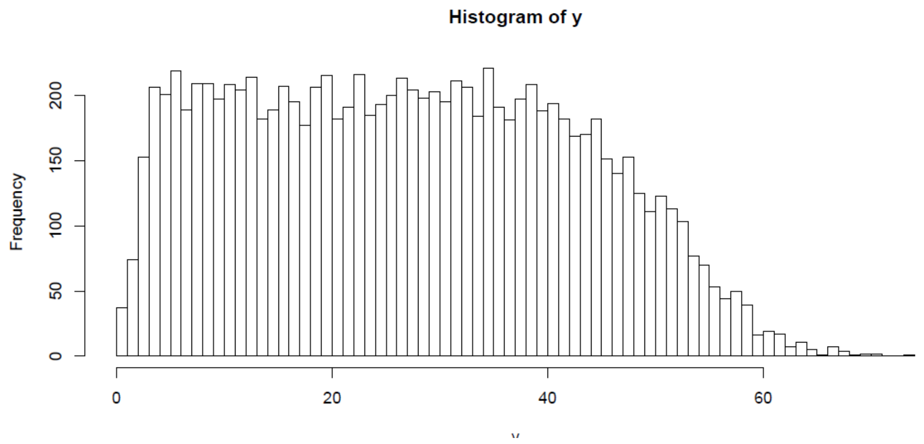
k	1	2	3	4	5	6	7	8	9	10
$\hat{\pi}_k$	0.107	0.000	0.146	0.164	0.049	0.126	0.004	0.202	0.183	0.020
$\hat{\lambda}_k$	5.18	8.26	11.32	19.26	27.75	28.01	28.27	37.17	47.85	48.83

log-likelihood 40573.98,

k	1	2	3	4	5	6	7	8	9	10
$\hat{\pi}_k$	0.110	0.136	0.097	0.107	0.103	0.025	0.149	0.074	0.119	0.081
$\hat{\lambda}_k$	5.24	11.31	17.56	22.30	27.80	31.13	35.04	40.76	46.68	49.32

log-likelihood 40573.98.

A problem in this case is that the different classes are difficult to distinguish from the data, making several configurations of mixtures of Poisson data possible.



Exercise 7

Cortez et al. [2009] considers a dataset of red wine quality of the Portuguese "Vinho Verde" wine. The following variables are measured:
Input variables (based on physicochemical tests): x_1 - fixed acidity, x_2 - volatile acidity, x_3, - citric acid, x_4 - residual sugar, x_5 - chlorides, x_6 - free sulfur dioxide, x_7 - total sulfur dioxide, x_8 - density, x_9 - pH, x_{10} - sulphates, x_{11} - alcohol.
Output variable (based on sensory data): y - quality (score between 0 and 10).
A simple model for the output quality is

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i$$

with $p = 11$ and $i = 1, ..., n = 1599$. This model is including all the variables as linear terms. In practice however, we would like to perform some kind of model selection. One way of describing possible submodels is

$$y_i = \beta_0 + \sum_{j=1}^{p} \gamma_j \beta_j x_{ij} + \varepsilon_i$$

where $\gamma_j = 1$ if the covariate is to be included into the model and 0 otherwise. Define $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_p)$. Our aim will be to minimise $J(\boldsymbol{\gamma})$ where

$$J(\boldsymbol{\gamma}) = -2 * \ell(\gamma) + 2 * (2 + \sum_{j=1}^{p} \gamma_j)$$

and $\ell(\gamma)$ is the log-likelihood value obtained by selecting the optimal $\boldsymbol{\beta}$ values for the given model.

(a) The figure below shows the results from two different versions of simulated annealing, the first changing one γ_j at a time, the second also allowing for two changes at a time. For the first version, one component is selected at random at each iteration. For the second version, first a random selection on whether to change one or two variables is made, thereafter the components to change are selected at random.

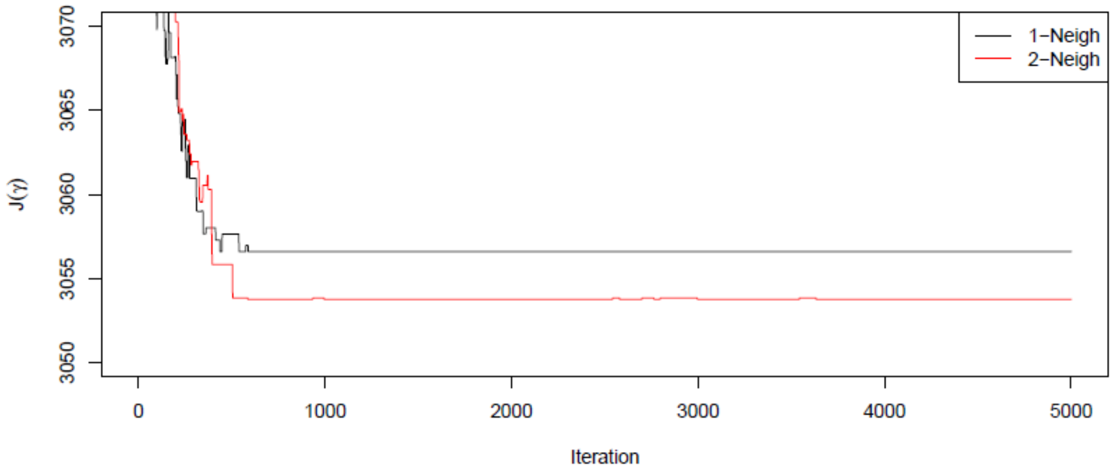
Give a short description of the simulated annealing algorithm. Discuss in particular the use of neighborhoods and relate that to the figure below.

Exercise 7 (a) In simulated annealing, first a neighborhood structure is chosen defining possible changes at each iteration. Thereafter a possible proposal $\boldsymbol{\gamma}^*$ is drawn from the neighborhood of the current value $\boldsymbol{\gamma}^t$. The proposal is then accepted with a probability

$$\min[1, \exp\{[J(\boldsymbol{\gamma}^t) - J(\boldsymbol{\gamma}^*)]/\tau_t\}$$

where τ_t is the temperature at iteration t. In order to guarantee convergence to the global maximum τ_t chould convergence to zero as $c/\log(1+t)$ where c is the depth (the smalles increase needed to escape from a local minimum). In practice this leads to much too slow convergence and a faster decrease is typically used.

Give a short description of the simulated annealing algorithm. Discuss in particular the use of neighborhoods and relate that to the figure below.



- In the figure a too fast cooling schedule has been used, the 1-Neigh is stuck before the 2-Neigh since this has less flexebility

(b) Assume that the temperature is selected to be very large. What kind of algorithm does then appear?

On the other hand, if the temperature is kept fixed, what kind of algorithm do we then obtain?

$$\min[1, \exp\{[J(\gamma^t) - J(\gamma^*)]/\tau_t\}$$

(b) If the temperature is chosen to be very large, just random changes are made, not using $J(\gamma)$ at all.

If a a very low temperature is chosen, changes are only made if a better proposal is found, corresponding to a greedy algorithm.

If the temperature is fixed, we obtain a Metropolis-Hastings algorithm with $J(\gamma)/\tau$ as target distribution. For $\tau = 1$ this then corresponds to a Bayesian posterior with the penalty term serving as a prior.